

Washington University in St. Louis
Washington University Open Scholarship

All Theses and Dissertations (ETDs)

Spring 3-19-2013

Learning with Single View Co-training and Marginalized Dropout

Minmin Chen

Washington University in St. Louis

Follow this and additional works at: <https://openscholarship.wustl.edu/etd>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Chen, Minmin, "Learning with Single View Co-training and Marginalized Dropout" (2013). *All Theses and Dissertations (ETDs)*. 1078.
<https://openscholarship.wustl.edu/etd/1078>

This Dissertation is brought to you for free and open access by Washington University Open Scholarship. It has been accepted for inclusion in All Theses and Dissertations (ETDs) by an authorized administrator of Washington University Open Scholarship. For more information, please contact digital@wumail.wustl.edu.

Washington University in St. Louis
School of Engineering and Applied Science
Department of Computer Science and Engineering

Dissertation Examination Committee:
Kilian Q. Weinberger, Chair
John Blitzer
John Cunningham
Tao Ju
Robert Pless
Bill Smart

Learning with Single View Co-training and Marginalized Dropout
by
Minmin Chen

A dissertation presented to the Graduate School of Arts and Sciences
of Washington University in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

May 2013
Saint Louis, Missouri

copyright by
Minmin Chen
2013

Contents

List of Figures	iv
List of Tables	vi
Acknowledgments	vii
Abstract	x
1 Introduction	1
1.1 Supervised, Unsupervised and Semi-Supervised Learning	3
1.1.1 Supervised Learning	3
1.1.2 Unsupervised Learning	5
1.1.3 Semi-Supervised Learning	5
1.2 Motivation and Overview	6
1.2.1 Learning with Weak Supervision	7
1.2.2 Domain Adaptation	10
1.2.3 Learning from Multiple Domains	13
1.2.4 Learning from Corrupted Data	14
1.2.5 Learning with Partial Supervision	15
1.2.6 Discussion	17
1.3 Optimization Background	18
1.3.1 Gradient-based Method for Unconstrained Optimization	18
1.3.2 Augmented Lagrangian Method for Constrained Optimization	23
1.3.3 Least-Squares Problem	24
1.3.4 Block Coordinate descent (Alternating Optimization)	26
2 Single View Co-training	28
2.1 Pseudo Multi-view Co-training	28
2.1.1 Introduction	29
2.1.2 Recap of Learning with Weak Supervision	31
2.1.3 Background and Related Work	32
2.1.4 Pseudo Multi-view Co-training	35
2.1.5 Extension	40
2.1.6 Experimental Results	42
2.1.7 Conclusion	50

2.2	Co-training for Domain Adaptation	50
2.2.1	Introduction	51
2.2.2	Recap of Domain Adaptation	52
2.2.3	Background and Related Work	54
2.2.4	Self-training	57
2.2.5	Feature Selection	58
2.2.6	Co-training for Domain Adaptation	60
2.2.7	Experimental Results	63
2.2.8	Conclusion	69
3	Marginalized Dropout	71
3.1	Marginalized Corrupted Feature Learning	72
3.1.1	Introduction	72
3.1.2	Recap of Learning from Multiple Domains	74
3.1.3	Background and Related Work	75
3.1.4	marginalized Stacked Denoising Autoencoders	78
3.1.5	Extension to High Dimensional Data	82
3.1.6	Experimental Results	83
3.1.7	Conclusion	90
3.2	Marginalized Corrupted Regularization	90
3.2.1	Introduction	91
3.2.2	Recap of Learning from Corrupted Data	94
3.2.3	Background and Related work	95
3.2.4	Learning with Marginalized Corrupted Features	97
3.2.5	Experiments	108
3.2.6	Conclusion	116
3.3	Marginalized Corrupted Labels	117
3.3.1	Recap of Learning with Partial Supervision	118
3.3.2	Introduction	119
3.3.3	Related Work	121
3.3.4	Method	124
3.3.5	Experimental Results	130
3.3.6	Conclusion	136
4	Conclusion and Future Directions	137
	References	142

List of Figures

1.1	Sample images from the MNIST handwritten digit dataset.	2
1.2	Sample images from the Caltech-256 object recognition benchmark set [73] and Bing top ranked results [11].	9
1.3	Sample review from the book (left) and kitchen appliance (right) domain of the Amazon review benchmark set [19].	11
1.4	Examples of applying (a) feature dropout (blank-out noise); (b) Bit-swap noise; (c) Poisson noise to create examples that look like “real” ones.	15
1.5	Sample image with <i>partial</i> tags.	16
1.6	Comparison of conjugate gradient descent (red) and steepest descent with line minimization in terms of convergence rate.	21
2.1	Venn diagram on learning with weak supervision.	31
2.2	Conditionally-independent assumption of Co-training.	33
2.3	Sample images from the paired handwritten digit dataset. Upper row: positive examples; lower row: negative examples.	43
2.4	The heatmap of two randomly initialized weight vectors \mathbf{u}, \mathbf{v} (upper row) and $\mathbf{u}^*, \mathbf{v}^*$ learned with PMD (lower row) on the input space.	44
2.5	The heatmaps of \mathbf{u}, \mathbf{v} in 20 PMC iterations (1 st iteration on top).	45
2.6	The progress made by the two classifiers $h_{\mathbf{u}}, h_{\mathbf{v}}$ in different PMC iterations.	46
2.7	Refining image search ranking with Multi-class PMC.	47
2.8	Comparison of PMC and existing works on recognition accuracy obtained with 300 web images and a varying number of Caltech-256 training examples.	48
2.9	Venn diagram on domain adaptation.	53
2.10	Comparison of self-training, SEDA and CODA on the relative test-error reduction over logistic regression, averaged across all 12 domain adaptation tasks.	66
2.11	Comparison of CODA and existing work on the relative test-error reduction over logistic regression, averaged across all 12 domain adaptation tasks.	67
2.13	The ratio of the average number of used features between source and target inputs (2.22), tracked throughout the CODA optimization. The three plots show the same statistic at different amounts of target labels.	67
2.12	Comparison of CODA and existing work on the absolute test error under varying amounts of labeled target data on 12 adaptation tasks.	70
3.1	Venn diagram on learning from multiple source domains.	74

3.2	Comparison of mSDA and existing works across all twelve domain adaptation task in the small Amazon review dataset.	87
3.3	Transfer ratio and training times on the small (<i>left</i>) and full (<i>right</i>) Amazon Benchmark data. Results are averaged across the twelve and 380 domain adaptation tasks in the respective data sets (5,000 features).	87
3.4	<i>Left</i> : Transfer ratio as a function of the input dimensionality (terms are picked in decreasing order of their frequency). <i>Right</i> : Besides domain adaptation, mSDA <i>also</i> helps in domain <i>recognition</i> tasks.	88
3.5	Venn diagram on learning from corrupted data.	94
3.6	Graphical model interpretation of marginalized corrupted features.	106
3.7	Classification errors of MCF predictors using blankout corruption – as a function of the blankout corruption level q – on all data sets for l_2 -regularized quadratic, exponential, and quadratic loss functions. The case of MCF with blankout corruption and $q = 0$ corresponds to a standard l_2 -regularized classifier. Figure best viewed in color.	110
3.8	The performance of standard and MCF classifiers with blankout and Poisson corruption models as a function of training set size on the Dmoz and Reuters data sets. Both the standard and MCF predictors employ l_2 -regularization. Figure best viewed in color.	112
3.9	Comparison between MCF and explicitly adding corrupted examples to the training set (for quadratic loss) on the Amazon (books) data using blankout corruption. Training with MCF is equivalent to using infinitely many corrupted copies of the training data.	113
3.10	Evaluation on the “Nightmare at test-time” scenario. Classification errors of standard and MCF predictors with a blankout corruption model – trained using three different losses – and of FDROP [67] on the MNIST data set using the “nightmare at test time” scenario. Classification errors are represented on the y -axis, whereas the amount of features that are deleted out at test time is represented on the x -axis. The images of the digit illustrate the amount of feature deletions applied on the digit images that are used as test data. Figure best viewed in color.	115
3.11	Venn diagram on learning with partial supervision.	119
3.12	Sample images with associated tags from the Espgame dataset.	120
3.13	Schematic illustration of FastTag. During training two linear mappings \mathbf{B} and \mathbf{W} are learned and co-regularized to predict similar results. At testing time, a simple linear mapping $\mathbf{x} \rightarrow \mathbf{W}\mathbf{x}$ predicts tags from image features.	124
3.14	Predicted keywords using FastTag for sample images in the Espgame dataset.	129
3.15	Comparison of FastTag and existing work in terms of F1 score vs. training time on the Corel5K, Espgame and IAPRTC-12 dataset.	132
3.16	Comparison of FastTag and existing work at different levels of tag sparsity.	134

List of Tables

2.1	Comparison of PMC and existing work on the paired handwritten digit dataset.	43
2.2	Statistics of the Amazon review dataset [19].	64
3.1	Statistics of the large and small set of the Amazon review dataset [19]. . . .	84
3.2	The probability density function (PDF), mean and variance of corrupting distributions of interest.	101
3.3	Moment-generating functions (MGFs) of corrupting distributions of interest.	103
3.4	Comparison of MCF and l_2 -regularization on classification errors obtained on the CIFAR-10 data set with simple spatial-pyramid bag-of-visual-word features.	114
3.5	Comparison of FastTag and existing work on the Corel5K dataset.	131
3.6	Comparison of FastTag and existing work on the Espgame and IAPRTC-12 dataset.	132

Acknowledgments

My utmost gratitude goes to my advisor Kilian Q. Weinberger. Without him, none of the work described in this dissertation would have been possible. Kilian has been an amazing advisor, who introduced me into the field, inspired me with invaluable insights and guided me through the entire course to finish this dissertation. He always makes himself available to discuss ideas and to provide extremely helpful feedbacks. His enthusiasm for research and encouragement have kept me continuing my work.

I would like to thank John Blitzer, Olivier Chapelle, Laurens van der Maaten and Fei Sha for their invaluable input and contributions to these works. John also kindly served as my external committee member and provided me with very useful comments. Every meeting with Olivier and Fei brings me new knowledge about the field. Laurens has led the effort for the work I presented in Section 3.2. I would also like to thank my other committee members John Cunningham, Tao Ju, Robert Pless and Bill Smart for their helpful suggestions.

I would like to thank Yixin Chen for his support during the first few years of my PhD study. Further, I would like to thank Jian-Tao Sun, Prabhakar Krishnamurthy and Alice Zheng for providing me with great opportunities to intern at Microsoft Research Asia, Yahoo! Labs and Microsoft Research.

I would like to thank my group members, Eddie Xu, Stephen Tyree, Dor Kedem and Matt Kusner for collaborating on projects, bouncing off ideas, and generously taking time to help with my writings and talks. I will miss the time we spent together working toward the same deadline. I would also like to thank all my friends in St. Louis, who have made the past six years some of the most memorable years in my life.

I would like to thank my parents and grandparents for their unconditional love and support, having faith in me when I have doubts. I am also very thankful to my uncle who first encouraged me to pursue postgraduate study.

With great love, I thank my husband Zijian Guo for sharing with me all the joys and supporting me through the rough time whole-heartedly.

Minmin Chen

Washington University in Saint Louis
May 2013

Dedicated to my parents.

ABSTRACT OF THE DISSERTATION

Learning with Single View Co-training and Marginalized Dropout

by

Minmin Chen

Doctor of Philosophy in Computer Science

Washington University in St. Louis, May 2013

Research Advisor: Kilian Q. Weinberger

The generalization properties of most existing machine learning techniques are predicated on the assumptions that 1) a sufficiently large quantity of training data is available; 2) the training and testing data come from some common distribution. Although these assumptions are often met in practice, there are also many scenarios in which training data from the relevant distribution is insufficient. We focus on making use of additional data, which is readily available or can be obtained easily but comes from a different distribution than the testing data, to aid learning.

We present five learning scenarios, depending on how the distribution we used to sample the additional training data differs from the testing distribution: 1) learning with weak supervision; 2) domain adaptation; 3) learning from multiple domains; 4) learning from corrupted data; 5) learning with partial supervision.

We introduce two strategies and manifest them in five ways to cope with the difference between the training and testing distribution. The first strategy, which gives rise to *Pseudo*

Multi-view Co-training (PMC) and *Co-training for Domain Adaptation* (CODA), is inspired by the co-training [23] algorithm for multi-view data. PMC generalizes co-training to the more common single view data and allows us to learn from weakly labeled data retrieved free from the web. CODA integrates PMC with an another feature selection component to address the feature incompatibility between domains for domain adaptation. PMC and CODA are evaluated on a variety of real datasets, and both yield record performance.

The second strategy marginalized dropout leads to *marginalized Stacked Denoising Autoencoders* (mSDA), *Marginalized Corrupted Features* (MCF) and *FastTag* (FastTag). mSDA diminishes the difference between distributions associated with different domains by learning a new representation through marginalized corruption and reconstruction. MCF learns from a known distribution which is created by corrupting a small set of training data, and improves robustness of learned classifiers by training on “infinitely” many data sampled from the distribution. FastTag applies marginalized dropout to the output of partially labeled data to recover missing labels for multi-label tasks. These three algorithms not only achieve the state-of-art performance in various tasks, but also deliver orders of magnitude speed up at training and testing comparing to competing algorithms.

Chapter 1

Introduction

Machine learning is a relatively new branch of artificial intelligence, which aims at getting computers to act based on past experience instead of being explicitly programmed. Tom M. Mitchell provided a widely quoted definition of learning from experience as follows:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E . [111]

Experience is commonly embodied in the (large amount of) external data inputted to the computer program. Considering the task of spam filtering, where the goal is to distinguish between spam and non-spam messages. To build such a machine, a large amount of email messages are often provided to learn patterns or regularities that can differentiate spam and non-spam messages. After learning, these patterns or regularities are then used to classify new email messages into spam or non-spam folders.

Figure 1.1 illustrates another example of recognizing handwritten digits. The task is non-trivial as handwriting varies greatly. To enable learning, we are provided with a data corpus of handwritten digits. Each input in the corpus is a 28×28 pixel image, which can be represented as a vector of 784 real numbers. The goal is to build a recognizer so that whenever a new image of handwritten digit comes in, we will be able to produce the identity of the digit $0, \dots, 9$ as the output.



Figure 1.1: Sample images from the MNIST handwritten digit dataset.

In summary, the goal of machine learning is to automatically extract statistical regularities and patterns presented in a data corpus, and later to act on unseen inputs with the use of these regularities and patterns.

Note that, there is another class of important techniques in machine learning, *reinforcement learning*, which has a different setup. It deals with the problem of finding suitable actions to take in a given situation in order to maximize a reward, and studies the tradeoff between exploration and exploitation. Instead of being provided with a data corpus beforehand, the learner receives feedback (reward) while acting. However, we are not going to discuss it in this thesis and would like to refer interested readers to [90, 149].

To measure the performance of different learning algorithms, the data corpus is often divided into two subsets. One is called the training data, on which the learning takes places. The remaining one is the testing data, on which the prediction and measure of quality are performed.

Note that, a *feature extraction* stage that preprocess the original raw input into an input vector is often required before any learning or testing can happen. For example, in the case of handwritten digit recognition, typically each image is translated and rescaled so that the digit is centered in a box of fixed size (28×28 in the example). The gray scale at each pixel of the box is then extracted to form an input vector. The space in which the input vectors lie in is commonly referred to as the feature (or instance) space. Note that the preprocessing of the training and testing inputs has to follow the same protocol. For almost all practical

applications, the feature extraction stage is equally important as the learning method itself. A good feature extraction stage reduces the noisy variations in the raw input and leads to easier pattern recognition in the learning phase. However, we will focus on the learning methods in this dissertation.

1.1 Supervised, Unsupervised and Semi-Supervised Learning

Based upon the types of information that are included in the training data, one can roughly divide different learning methods into three camps: supervised, unsupervised and semi-supervised learning.

1.1.1 Supervised Learning

Let $\mathcal{X} \in \mathcal{R}^d$ denote the feature space of dimension d , and \mathcal{Y} the label (or output) space. In supervised learning, the training data is assumed to be sampled i.i.d. (independently and identically distributed) from some joint distribution \mathcal{D} on $\mathcal{X} \times \mathcal{Y}$ and comes in pairs, *i.e.*, $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\} \subset \mathcal{X} \times \mathcal{Y}$. Here \mathbf{x}_i is the input vector and y_i is its associated label (or output). In the example of handwritten digits, the input vector \mathbf{x}_i corresponds to the 784 pixel values extracted from the 28×28 grid, and the label y_i corresponds to the identity of the digit, ranging from 0 to 9. It is assumed that there is an underlying target function f which generates the labels from the inputs, *i.e.*, $y_i = f(\mathbf{x}_i)$. However, the target function f is usually unknown, and needs to be inferred from the training data. We can further divide the tasks into a *classification* or a *regression* one based on the labels. In the handwritten digit recognition, or the spam filtering task we introduced, the goal is to categorize each input vector into an output from a discrete and finite set. In these cases, we have a classification problem. Examples of classification algorithms include k -nearest neighbors [47] and support vector machines [156, 135]. When the labels take continuous values, it is a regression task. Examples of regression algorithms include linear regression, kernel regression [64] and gaussian process [125]. Note that there are more complicated tasks

where structural dependencies exist between the inputs \mathbf{x}_i 's and the outputs y_i 's. In these cases more specialized algorithms will be required.

Generalization. The output of these learning algorithms can be expressed as a function $h(\mathbf{x})$ which takes an input vector \mathbf{x} and outputs a prediction $\hat{y} = h(\mathbf{x})$. In the case of supervised learning, the evaluation of learning quality is well-defined since we can compare the prediction \hat{y} to the ground truth y . Let us define a binary indicator function

$$[h(\mathbf{x}) \neq y] = \begin{cases} 0, & h(\mathbf{x}) = y \\ 1, & h(\mathbf{x}) \neq y \end{cases}$$

We can then compute the error of a learner on the training set (the empirical risk) as

$$\epsilon_D(h) = \frac{1}{n} \sum_{i=1}^n [h(\mathbf{x}_i) \neq y_i] \quad (1.1)$$

But we are not really interested in the error on the training set, instead, we want to find a learner which generalizes well to unseen test data. The generalization risk can be expressed as the expected error under the underlying distribution \mathcal{D} ,

$$\epsilon(h) = \mathbf{E}_{(\mathbf{x}, y) \sim \mathcal{D}} [h(\mathbf{x}) \neq y] \quad (1.2)$$

However, since the distribution \mathcal{D} is unknown, we can not explicitly compute it. Fortunately, we can relate the generalization risk of any hypothesis h from a hypothesis class \mathcal{H} to its empirical risk with the help of Vapnik-Chervonenkis theory [156]. For any $h \in \mathcal{H}$, trained on a set of training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ sampled i.i.d. from \mathcal{D} , with probability $1 - \delta$,

$$\epsilon(h) \leq \epsilon_D(h) + 2\sqrt{2 \frac{|\mathcal{H}| \log \frac{2en}{|\mathcal{H}|} + \log \frac{2}{\delta}}{n}} \quad (1.3)$$

where $|\mathcal{H}|$ denotes the Vapnik-Chervonenkis dimension [156] of the hypothesis class. We can see that the gap between the generalization and empirical error grows inversely with the size of the training data.

In other words, the generalization performance of supervised learning methods are predicated on the condition that a sufficiently large quantity of training data is available for learning.

1.1.2 Unsupervised Learning

Although extracting the input vectors can often be done very conveniently, acquiring labels turns out to be very expensive for a many tasks. For instance, it takes 2 years to annotate 4,000 sentences in the Penn Chinese Treebank. In annotating phone conversation, 400 hours of annotation time is required for one hour of speech.

Unsupervised learning offers possibilities to circumvent the costly manual annotation. In unsupervised learning, the training data $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ is still assumed to be drawn i.i.d. from some underlying distribution \mathcal{D} , however, the labels of these inputs are unknown. Common unsupervised learning tasks include: 1) *dimensionality reduction*; The goal is to project the inputs to a lower-dimensional space where a more meaningful distance interpretation or visualization is possible. Examples of dimensionality reduction algorithms include principal component analysis [89], Isomap [151], locally linear embedding [130], Laplacian eigenmap [8], and maximum variance unfolding [163]. 2) *clustering*; The goal is to discover similarity between examples and group the inputs into clusters. Examples of clustering algorithms include k -means, and spectral clustering [114]. 3) *density estimation*; The goal is to infer the underlying distribution that have generated the training data X . Examples of density estimation algorithms include mixture models [107].

Since the inputs given to the learner are unlabeled, generally there is no error or reward signal that can be used to evaluate a potential solution in unsupervised learning. Though for some tasks, such as clustering, it is possible to measure the quality of the solution if some ground truth about the underlying clusters is available.

1.1.3 Semi-Supervised Learning

Semi-supervised learning is halfway between supervised and unsupervised learning. It does not seek to completely eliminate the needs for labels, but resorts to unlabeled data to minimize the amount of labeling effort required to achieve good performance.

In semi-supervised learning, the training data contains a relatively small set of input-output pairs drawn from some joint distribution \mathcal{D} , $L = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$, as well

as another set of inputs $U = \{\mathbf{x}_{n+1}, \dots, \mathbf{x}_m\}$ drawn i.i.d. from the same distribution. The labels of the inputs in U are unknown though. The size of the labeled data L is usually much smaller than the unlabeled data U , that is, $n \ll m$. The goal is to find a hypothesis $h \in \mathcal{H}$ using both the labeled inputs from L and the unlabeled inputs from U , so that it accurately predicts the labels of test inputs sampled from the same distribution.

Semi-supervised learning algorithms can be roughly divided into four classes based on the different assumptions made in these algorithms: 1) *generative models*; These models are based on the smoothness assumption of semi-supervised learning. Examples of algorithms in this class include EM algorithms on mixture models [117]. 2) *low density separation models*; These models follow the assumption that the decision boundary should lie in a low-density region. Examples of algorithms in this class include Transductive SVM [88, 38, 143], entropy or information regularization models [150, 71]. 3) *manifold-based methods*; These models follow the assumption that high-dimensional data lie (roughly) on a low-dimensional manifold. Dimensionality reduction algorithms [89, 151, 130, 8, 163] are performed on all the data, labeled or unlabeled, to find a new representation, and supervised learning is then performed on the new representation using labeled data only. 4) *graph-based models*; These models are inspired by the cluster assumption: points within the same cluster should be of the same class. Algorithms in this class penalize non-smoothness along the edges of a weighted graph and propagate labels along the edges. Examples of algorithms in this class include graph mincut [24], gaussian random fields and harmonic function method [175], graph kernels [39, 144]. Self-training [168] and co-training [23] are another two prominent semi-supervised learning algorithms. A complete discussion on these work is beyond the scope of this thesis. We refer the readers to [36, 174] for a detailed study on this topic.

1.2 Motivation and Overview

Machine learning has become the tool of choice for many applications. However, successful applications of machine learning often rely on the existence of large quantities of labeled data, which can be difficult to obtain. In recent years, the research community has delved into crowdsourcing as a potential platform for acquiring labels cheaply and at scale [81]. However,

crowdsourced labels are often very noisy and inconsistent, and obtaining high-quality labels can still be expensive, especially for large-scale datasets.

Unsupervised and semi-supervised learning approaches alleviate the problem. However, most of the algorithms we reviewed in section 1.1.2 and 1.1.3 still assume that 1) the training data, labeled or unlabeled, comes from the same distribution as the testing data 2) the labels are clean and complete. As technology enables data collection at ever increasing scale and speed, nowadays we have access to an enormous amount of data. Unfortunately, not all of these data, in fact, a majority part of these data does not necessarily follow the same distribution as the testing data.

In this dissertation, we focus on making use of additional external data, which comes from a slightly different distribution than the test data, to aid learning. We are concerned with five learning scenarios that differ from classical supervised, unsupervised, or semi-supervised learning, *i.e.* 1) learning with weak supervision; 2) domain adaptation; 3) learning from multiple domains; 4) learning from dropout distribution; 5) learning with partial supervision. In Chapter 2, we are going to cover the first two scenarios with a common strategy – single view co-training. In Chapter 3, we are going to elaborate our second strategy – marginalized dropout to learn in the remaining three scenarios.

1.2.1 Learning with Weak Supervision

Nowadays, it is possible to obtain large quantities of data for almost any topic or class description with very little human intervention (*e.g.* through automated image search or wikipedia lookups). For example, we can retrieve hundreds of thousands of images of different objects by searching on any of the search engines using the class description of these objects as queries. Typing the keywords “eiffel tower” into Bing returns thousands of images semantically applicable to the concept of eiffel tower. However, not all of the retrieved results are relevant to the query concept. As images are mostly indexed by surrounding texts, the fraction of returned results which are *visually* relevant to the query concept is usually small. As a result, the distribution of the retrieved images are very different from that of the training/testing data carefully curated by human annotators.

Figure 1.2 shows sample images from the Caltech256 object recognition benchmark dataset [73] and the top ranked results [11] returned by BingTM image search using the class names as queries. Each row shows images from one particular object class.

Caltech256 contains over 30,000 images of 256 object categories handpicked and annotated by human annotators. Candidate images are collected from the web following the same protocol we just introduced, *i.e.*, through automated image search. What is different in this case is that a manual cleanup process is implemented afterwards. Annotators are explicitly asked to rule out images that are 1) very cluttered; 2) line drawings; 3) abstract or artistic; 4) not largely occupied by the object. As a result, images within each category are fairly consistent both visually and semantically.

In contrast, the web images appear to be less homogeneous visually due to polysemy, caricaturization, as well as variations in viewpoints. One can observe that: 1) even the top ranked results returned still contain outliers (*i.e.*, images unrelated to the object category *visually*); For instance, the naked women in the “eiffel tower” category, and the cliff in the “hawkbill” category. They are not completely irrelevant to the query concept itself as a word often has multiple meanings. The image is returned probably because the caption or the text surrounding the image describes it as “a women posing like the eiffel tower”. To some extent, it is desirable for a search engine to be able to return a diverse set of results. But the diversity also poses a great challenge for learning. 2) even relevant results in the Bing group are still of noticeable difference to the training images from Caltech256; First, while the Caltech256 benchmark set consists of only real photographs, the Bing retrieved results include cartoon drawings for almost all the seven categories. Second, the hand collected images contain only object of interest, while the Bing counterpart often includes extraneous items, such as people or faces. These image can distract the learning as “people” and “faces” are also valid object categories in the Caltech256 set. 3) Comparing to the human curated images, where the object of interest is in the center, the web images are often taken from very different shooting distances or angles, causing the images to appear less common visually.

In summary, the images returned by Bing search are semantically and visually less coherent than the images from the Caltech256 benchmark set. One can regard the Bing group as a noisy superset of the Caltech256 training data. It contains images which are closely related

to the object category, but also images which are only remotely connected, or even irrelevant. In this setting, we aim at utilizing the noisy set to assist learning.

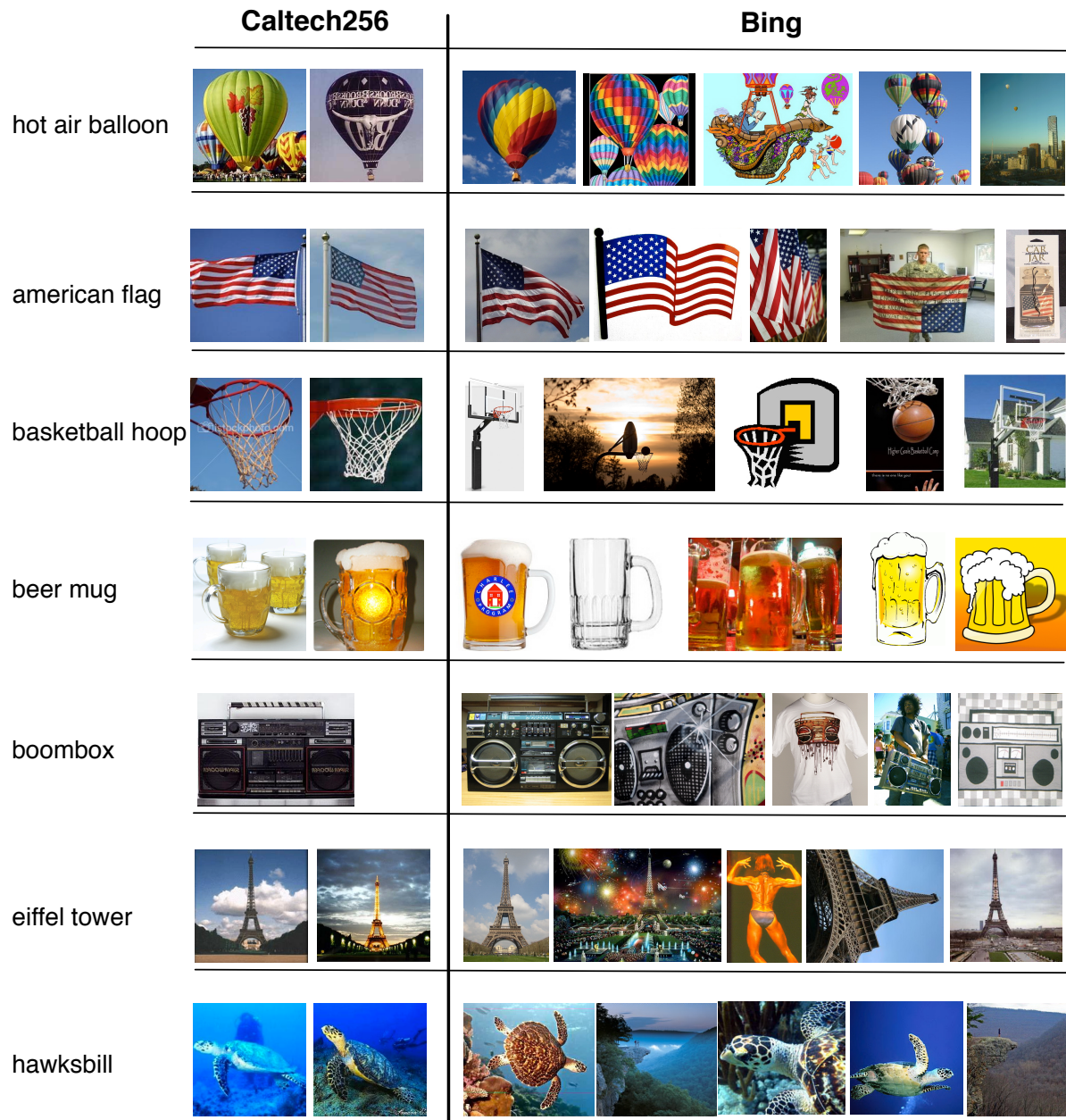


Figure 1.2: Sample images from the Caltech-256 object recognition benchmark set [73] and Bing top ranked results [11].

Notation. We define the setup of learning with weak supervision as follows: let $L = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$ be a small set of input-output pairs drawn from some joint distribution \mathcal{D} . Let $L' = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}, m \gg n$ be another set of labeled inputs drawn from a different distribution \mathcal{D}' . The support of \mathcal{D}' is a *superset* of the support of the testing distribution \mathcal{D} . The goal is to find a hypothesis $h \in \mathcal{H}$ using both the labeled data from L and L' to accurately predicts the labels of the testing data drawn from the distribution \mathcal{D} .

Pseudo Multi-view Co-training (PMC) [42]. Section 2.1 describes Pseudo Multi-view Co-training (PMC), an new framework we developed to learn from weak supervision. It is a variant of the co-training [23] algorithm for semi-supervised learning. Similar to Co-training, it cherry-picks the examples which are close to the training data from the noisy labeled set L' to expand the labeled set L . Different to co-training, which is designed to work with multi-view data, PMC works on the more common single view data. The idea is to automatically divide the features of a single view data set into two mutually exclusive subsets – thereby creating a pseudo-multi-view representation for co-training. Inspired by a theoretical analysis of Balcan et al. [5], the feature division is learned explicitly to satisfy the necessary conditions to enable successful co-training. PMC successfully exploits noisy web search results to improve on the challenging Caltech-256 object recognition task.

1.2.2 Domain Adaptation

The second learning scenario we are interested arises when we have plenty of training data available from some *source* domain, but are ultimately interested in learning a model that generalizes well to a related *target* domain. Re-collecting training data for the *target* domain is costly, and one often wishes to leverage the data from the *source* domain, and adapt the learner trained on the source inputs to the new *target* domain. The challenge is that the data distribution of the *target* domain often differs from that of the *source* domain.

The needs to adapt between domains surface everyday as personalization becomes prevalent, not only in web applications such as personalized news or ads, but also in education, healthcare and television, etc. As an example, spam filters can be trained on some public collection of spam and non-spam emails. However, when applied to an individual user’s



★★★★★ **Stunning, epic, extraordinary debut novel**, October 9, 2003
By [Peggy Vincent "author and reader"](#)
This review is from: [The Kite Runner \(Hardcover\)](#)
I read 2-3 books a week, and this is without doubt my **favorite** of this year. A **beautiful** novel by Afghan-American Khaled Hosseini that rans among the **best-written** and provocative stories of the year so far. This unusually **eloquent** story is also about the fragile relationship ...

★★★★★ **Plusses and Minuses**, July 18, 2005
By [Elaine Corwin "buttercup21"](#)
[Amazon Verified Purchase](#) ([What's this?](#))
This review is from: [Cuisinart DCC-1200 Brew Central 12-Cup Programmable Coffeemaker](#)
This unit makes the **best** coffee I've had in a home. It is my **favorite** unit. It makes **excellent** and HOT coffee. The carafe is **solidly constructed** and fits securely in squared body. The timer is **easy to program**, as is the clock ...

Figure 1.3: Sample review from the book (left) and kitchen appliance (right) domain of the Amazon review benchmark set [19].

inbox, we may want to “personalize” the spam filter, *i.e.*, adapt it to fit the user’s own distribution of emails in order to achieve better performance. Shift of domain is common in natural language processing as well. In general, labeled data for tasks like part-of-speech tagging, parsing, or information extraction are drawn from a limited set of document types and genres in a given language because of availability, cost, and project goals. However, applications for the trained systems often involve somewhat different document types and genres [52]. For example, most of the labeling were performed on text from news articles (in particular, the Wall Street Journal), which uses a very specialized set of languages. Labeled data from this domain can be a poor match to be used as training examples for other domains, such as biomedical texts, mails or meeting transcripts, etc. Another example arises in the vision community. The changes in camera, image resolution, lighting, background, viewpoint, and post-processing cause substantial distribution shift for vision data [132]. For example, there are a large number of readily categorized inventory images online (from Amazon, Ebay, etc.), which can be useful for object recognition tasks. However, inventory images are mostly captured in a much more controlled manner, and look very different from the ones taken in real world surroundings. As a result, directly porting a learner trained on the *source* domain (inventory images) to the *target* domain (real life images) often leads to degenerated performance.

If the data distribution from the *source* and *target* domain differs vastly, then it is hopeless to generalize from one to another. Nevertheless, adaptation is possible when the two domains are related, *e.g.*, the two distributions share supports, or the supports of the two distributions overlap. Figure 1.3 shows two sample reviews from the Amazon review benchmark set [19], one from the “books” domain (left), and the other from the “kitchen appliances” domain (right). We highlight the sentiment words that are shared across domains in green, and the ones that are specific to different domains in blue (source) and red (target) respectively. One can see that very different vocabularies are used to express the same kind of opinion (in this case, positive sentiment) about different products. People use “well-written” and “eloquent” when recommending books. On the other hand, “solidly constructed” and “easy to program” are used as strong indicators of positive view about coffee makers. Fortunately, there are also words shared across both products, like “best”, “favorite”. These features are invariant across different domains, and will be used to bridge the two distributions.

Notation. We formalize the setup in domain adaptation as follows: denote as $L_S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{n_S}, y_{n_S})\} \subseteq \mathcal{X} \times \mathcal{Y}$ the set of labeled data drawn from some source distribution \mathcal{D}_S , $L_T = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{n_T}, y_{n_T})\} \subseteq \mathcal{X} \times \mathcal{Y}$ a small set of labeled data sampled from the target distribution \mathcal{D}_T . Let $U_T = \{\mathbf{x}_{n_T+1}, \dots, \mathbf{x}_{m_T}\}$, $m_T \gg n_T$ be another set of unlabeled data drawn i.i.d. from the same target distribution. Note that, L_T could be empty in some cases, *i.e.*, we only have access to unlabeled data from the target domain. The goal is to find a hypothesis $h \in \mathcal{H}$ using both the labeled data from L_S , the labeled and data L_T and unlabeled data U_T from the target domain, so that it accurately predicts the labels of the testing data drawn from the *target* distribution \mathcal{D}_T .

Co-training for Domain Adaptation (CODA) [41]. Section 2.2 describes Co-training for Domain Adaptation (CODA), an algorithm we developed to adapt from a single *source* domain. CODA is a variant of the Pseudo Multi-view Co-training (PMC) algorithm we introduced for learning with weak supervision. It slowly adapts the training set from the source distribution to the target, both data-wise and feature-wise. To shift the distribution of the training data from source to target, CODA gradually add inputs from the target domain to the training set using co-training. Meanwhile, it includes a feature selection component to shift the features used by the model from source-heavy to target-heavy. Combining these two strategies, CODA significantly outperforms existing algorithms on a standard domain adaptation benchmark set [19].

1.2.3 Learning from Multiple Domains

Most existing work on domain adaptation has been focused on the one source and one target scenario. Here we generalize to the cases when there are multiple source domains, each associated with a different distribution. Glorot et al. [69] demonstrated that learning a shared representation using data from all the available domains leads to better performance than learning on a single source and target domain.

Notation. We formalize the setup in learning with multiple domains as follows: Let K denote the number of source domains available. Each source domain is associated with a distribution $\mathcal{D}_{S_j}, j = 1, \dots, K$, from which we can sample a set of labeled inputs $L_{S_j} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{n_{S_j}}, y_{n_{S_j}})\}$ and a set of unlabeled inputs $U_{S_j} = \{\mathbf{x}_1, \dots, \mathbf{x}_{m_{S_j}}\}$. Note that, it is possible that we only have access to unlabeled data for some source domains. Same as before, denote as $L_T = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{n_T}, y_{n_T})\} \subseteq \mathcal{X} \times \mathcal{Y}$ a small set of labeled data sampled from the target distribution \mathcal{D}_T . Let $U_T = \{\mathbf{x}_{n_T+1}, \dots, \mathbf{x}_{m_T}\}, m_T \gg n_T$ be another set of unlabeled data drawn i.i.d. from the same target distribution. Note that, L_T could be empty in some cases. Again, the goal is to find a hypothesis $h \in \mathcal{H}$ that works well for the target domain.

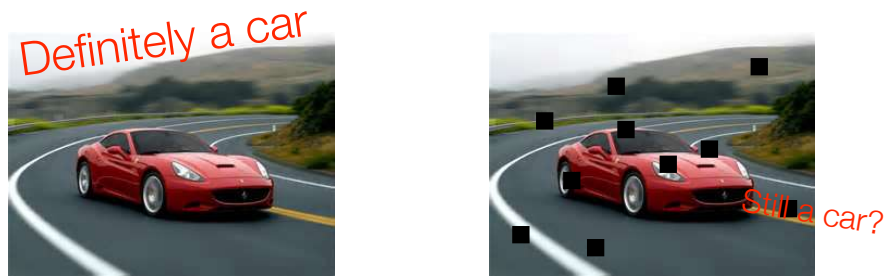
marginalized Stacked Denoising Autoencoders (mSDA) [43]. Section 3.1 describes marginalized Stacked Denoising Autoencoders (mSDA), an algorithm we developed to learn from multiple source domains. mSDA is inspired by the Stacked Denoising Autoencoders (SDAs) algorithm Glorot et al. [69] adopted for domain adaptation. Similar to SDAs, mSDAs learn a general data representation through corruption and reconstruction using all the unlabeled data from different domains, and then train a classifier using labeled data on the new representation. Different to SDAs, which employ a nonlinear encoders and decoders to learn the representation, mSDAs use single-layer *linear* denoising autoencoders as the basic building blocks. Our new formulation has some very desirable optimization properties, such as layer-wise closed form solution and noise marginalization. mSDAs match the record accuracy attained by SDAs while reducing the training time by several orders of magnitude.

1.2.4 Learning from Corrupted Data

Again here we focus on the scenario when the training data from the relevant distribution is insufficient. However, in this case, we do not assume there is data from a different distribution that is readily available or can be easily obtained. Training on the small set of training data often results in degenerated performance on unseen testing data although the learner has low empirical risk, which is commonly referred to as overfitting. The standard approach to deal with overfitting is through regularization or learning with priors. However, these approaches can be unintuitive for practitioners (such as biologists) as prior knowledge on the model parameters is required to select a proper regularizer or prior. Instead, we explore using feature dropout (or data corruption) to create a distribution with known density function and sample from it more training data to improve the robustness of the learned classifiers.

Figure 1.4 shows some of the corrupting distributions we used to create the additional training examples. Feature dropout randomly removes some active features (in this case, pixels or image patches of an image) from each example. Bit-swap noise randomly replaces some active features with another ones, which is extremely useful for text document. Imagine people switch between synonyms, such as replacing “president” with “obama” in the example. Poisson corruption is also useful for count vectors. As shown in the figure, applying these corruption does not change the output, but produces additional training examples that can capture some of the distribution variation which can not be captured in the original small set of training examples.

Marginalized Corrupted Feature (MCF) [155]. Section 3.2 describes Marginalized Corrupted Features (MCF) regularization, an new learning framework we developed to improve generalization when no additional external data is available. MCF is inspired by the marginalized corruption employed in mSDA. It extends the training set with infinitely many artificially generated training examples that are obtained by corrupting the original training data. In other words, instead of approximating the exact statistics of the testing distribution \mathcal{D} with finite data, MCF learns from a slightly modified data distribution \mathcal{D}' with infinite training data. MCF is practical and efficient for a range of predictors and corruption models. We show empirically on a variety of data sets that MCF classifiers can be trained efficiently, may generalize substantially better to test data, and are also more robust to feature deletion at test time. In contrast to regularization or learning with prior method, learning with MCF



(a) object recognition

president	1	0
Obama	0	1
important	1	Still politics?
game-changing	0	
...	...	
bill	1	0
law	0	1

Politics

(b) text classification

keep	0	
amazing	5	4
ideas	2	Still positive?
value	0	
poor	0	
...	...	
average	1	

Positive review

(c) sentiment analysis

Figure 1.4: Examples of applying (a) feature dropout (blank-out noise); (b) Bit-swap noise; (c) Poisson noise to create examples that look like “real” ones.

is more intuitive as the corrupting is applied to the data itself, and we provides some simple guideline for choosing corrupting distributions.

1.2.5 Learning with Partial Supervision

The last learning scenario we are going to explore is to learn with partial supervision. Acquiring clean and complete labels can be time consuming and require domain expertise that few possess. Lately, people have been turning to crowdsourcing as a tool for obtaining labels at scale. However, crowd sourced labels are noisy and notoriously incomplete.



Figure 1.5: Sample image with *partial* tags.

We take automatic image annotation [17, 49, 57, 75, 85, 94, 104] as an example. Given an image, the goal is to annotate it with the complete list of tags that describe all visual features present in the image. Note that it is easy to tag an image with a few of the most prominent visual features, but to obtain the complete list can be quite difficult. The ESP game [162] takes the novel approach of allowing free-form input from the user, which is quick to do, but incentivizes pairs of labelers to match their answers. This results in tag sets with high precision, but without guarantees for high recall; that is, each image may be tagged with only a small set of tags that describe the most obvious visual features.

Figure 1.5 shows an image with a partial list of tags, which prescribes the dominant objects in the image, *i.e.*, snow, lake and feet. Our goal is to infer the full list of relevant tags, which describes all the visual features present in the image, *e.g.*, mountain, snow, sky, lake and water, etc. To be able to tag an image with the full list of tags is important for indexing in searchable image databases such as Flickr, Picassa or Facebook. The full list of tags could also serve as a starting point for image caption.

Notation. We formalize the setup of automatic image annotation with partial tags as follows: let $\mathcal{T} = \{\omega_1, \dots, \omega_T\}$ denote the dictionary of T possible annotation tags. Let $L = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\} \subset \mathcal{R}^d \times \{0, 1\}^T$ denote the set of training data, where each vector $\mathbf{x}_i \in \mathcal{R}^d$ represents the features extracted from the i -th image and each \mathbf{y}_i prescribes the small *partial* subset of tags that are appropriate for the i -th image. Our goal is to learn a linear function $\mathbf{W} : \mathcal{R}^d \rightarrow \mathcal{T}$, which maps a test image \mathbf{x}_i to its *complete* tag set.

FastTag. Section 3.3 describes FastTag, an algorithm we developed for image annotation using incomplete user tags. It employs a simple yet effective strategy to cope with overly

sparse supervision. FastTag learn two linear mappings simultaneously, one to recover the missing labels, and the other one to project the inputs to the predicted labels. These two mappings are co-regularized in a jointly convex loss function, which can be efficiently optimized with closed form updates. The simplicity of the framework allows us to incorporate a variety of image descriptors cheaply and to scale to datasets of a large number of images. FastTag matches the current state-of-the-art in tagging quality, yet reduces the training and testing times by several orders of magnitude on several standard real-world benchmark data sets.

1.2.6 Discussion

Note that there are several other popular learning paradigms that share similar goals as the ones we introduced. For example, active learning [138] seeks to reduce the overall amount of labels required by allowing the learning algorithm to interactively query for new labels. It is hoped that with a smart query strategy, the total number of examples required to learn a concept can be much lower than that would have been required in normal supervised learning.

Multiple instance learning (MIL), originally introduced by Dietterich et al. [55], is highly related to learning with weak or partial supervision. Rather than requiring each instance to be labeled as positive or negative, MIL algorithms take a set of bags that are labeled positive or negative as inputs. Labeling effort is reduced in this case as each bag contains many instances and a single label is assigned to one bag of instances. A bag is labeled negative if all the instances in it are negative, or positive if there is at least one positive instance in the bag. Given a collection of labeled bags, MIL algorithms aim at either 1) inducing a concept that will label individual instances correctly; or 2) labeling bags without inducing the concept.

Multi-task learning (MTL) [30, 59, 58, 165, 37] is a learning framework closely related to domain adaptation. Different from domain adaptation, usually there is only a single distribution on the observations in multi-task learning. Instead, the target functions $f_i(\mathbf{x}), i = 1, \dots, K$ differ across tasks, where K is the number of tasks. The goal of MTL

is to improve the performance of learning algorithms on all the tasks by learning classifiers for multiple tasks jointly. It works particularly well when the tasks are related and under-sampled.

1.3 Optimization Background

Optimization plays a very important role in machine learning. Most of the works we are going to include in this dissertation follows a simple scheme, that is formulating the objective as an optimization problem, and then solving with an optimization solver. In this section, we are going to briefly review some of the general optimization techniques used in these works. We will start with the basic first order gradient based method for unconstrained optimization problems, followed with a discussion on the augmented Lagrangian method we employed to solve problems with constraints. Further, we will review the analytical solution for least square problems, and the alternating optimization (block coordinate descent) method we applied to transform an optimization problem into two least square problems so that each one of them can be solved analytically.

1.3.1 Gradient-based Method for Unconstrained Optimization

First, we are going to introduce several gradient-based optimization techniques for unconstrained problems:

$$\min_{\mathbf{x} \in \mathcal{R}^n} f(\mathbf{x}) \tag{1.4}$$

For most of the analysis, we assumed that f is a continuously differentiable function. Note that in our algorithms, there are a few optimization problems which have non-differentiable objective functions, they are either handled with a differentiable approximation or by resorting to sub-gradient methods [26].

We will start with a brief discussion of the necessary and sufficient conditions for an optimal solution, which offers important insight on several of the optimization solvers we are going to introduce.

Proposition 1.1. (Necessary Optimality Condition) Let \mathbf{x}^* be an unconstrained local minimum of $f : \mathcal{R}^n \rightarrow \mathcal{R}$, and assume that f is continuously differentiable in an open set S containing \mathbf{x}^* , then

$$\nabla f(\mathbf{x}^*) = 0 \tag{1.5}$$

If in addition f is twice continuously differentiable within S , then

$$\nabla^2 f(\mathbf{x}^*) : \text{positive semidefinite.} \tag{1.6}$$

Proposition 1.2. (Sufficient Optimality Condition) Let $f : \mathcal{R}^n \rightarrow \mathcal{R}$ be twice continuously differentiable in an open set S . Suppose $\mathbf{x}^* \in S$ satisfies the conditions that

$$\nabla f(\mathbf{x}^*) = 0, \quad \nabla^2 f(\mathbf{x}^*) : \text{positive semidefinite.} \tag{1.7}$$

Then, \mathbf{x}^* is a strict unconstrained local minimum of f .

We are going to skip the proofs, and refer interested readers to [13].

The first class of methods we are going to review follow a simple iterative descent idea. In other words, they start from some initial point \mathbf{x}_0 and successively generate $\mathbf{x}_1, \mathbf{x}_2, \dots$ as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k, \quad k = 0, 1, \dots \tag{1.8}$$

such that the objective function f is decreased at each iteration. The process stops when the optimality conditions are satisfied. Here the scalar α_k is commonly referred to as the stepsize and $\mathbf{d}_k \in \mathcal{R}^n$ as the descent direction.

Descent direction. The class of methods is widely referred to as gradient based method as the descent direction is chosen to be of the opposite direction as the gradient so that the loss can decrease at each iteration. In other words,

$$\nabla f(\mathbf{x}_k)^\top \mathbf{d}_k < 0. \tag{1.9}$$

There are many directions that satisfy the condition. Two widely used descent directions lead to the two classes of popular gradient based methods, the steepest descent method,

and the newton's method. For steepest descent methods, the descent direction is the right opposite of the gradient direction, that is,

$$\mathbf{d}_k = -\nabla f(\mathbf{x}_k) \Rightarrow \mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) \quad (1.10)$$

For newton's methods, the descent direction is set so that gradient of the second order taylor expansion of $f(\mathbf{x}_{k+1})$ reaches zero, *i.e.*,

$$\mathbf{d}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k) \Rightarrow \mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k) \quad (1.11)$$

Stepsize. Many different strategies have been proposed to set the stepsize as well. For example, the constant stepsize method simply sets $\alpha_k = s$, where s is a small constant. Though a simple approach, it runs into problems when the constant is either too large (leading to divergence) or too small (slow convergence). The diminishing stepsize method starts with a relatively large step size and decreases the stepsize at each iteration. Some special care has to be taken to ensure sufficient decrease in the loss at each iteration in this case. The exact minimization or limited minimization rules choose the stepsize by solving another optimization problem to minimize the loss $f(\mathbf{x}_{K+1})$ w.r.t the step size α_k . Several line search rules were also proposed, such as the Armijo rule, which finds the largest step size possible with a sufficient decrease in the loss by successive stepsize reduction.

Convergence. If the descent direction is gradient related, then convergence to a stationary point (zero derivative) is guaranteed for methods using the exact minimization, limited minimization or the Armijo line search rule for general function classes. For constant or diminishing stepsize method, Liptchitz continuity needs to be further enforced on the objective function to guarantee convergence.

Conjugate Gradient Method

In this section, we are going to explain the Polack-Ribiere flavor of conjugate gradient method [122] in more detail. It is employed in a optimization package ¹ used in several of our algorithms. Conjugate gradient methods is a a class of optimization techniques that

¹<http://www.gatsby.ucl.ac.uk/~edward/code/minimize/minimize.m>

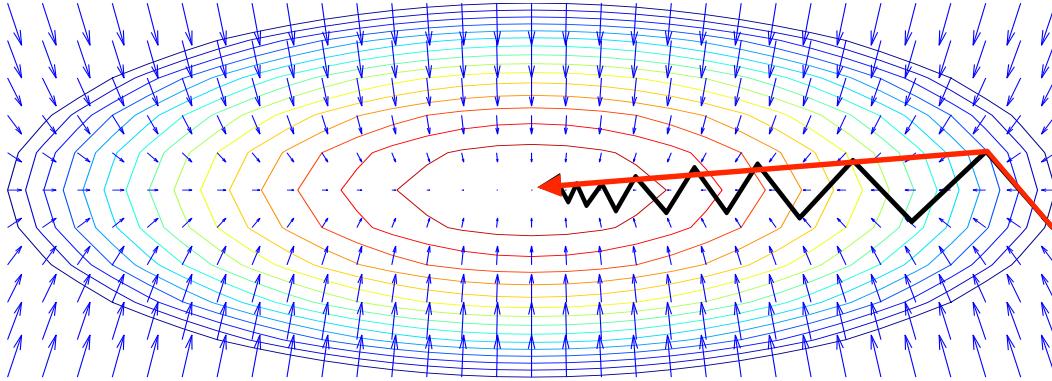


Figure 1.6: Comparison of conjugate gradient descent (red) and steepest descent with line minimization in terms of convergence rate.

is between steepest descent, which is known to have slow convergence rate, and newton’s method, which has a big overhead for computing and storing the hessian matrix for large problems.

Many loss functions can be roughly approximated as a quadratic form near the local minima [13],

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b} + c \quad (1.12)$$

Figure 1.6 compares the convergence rate of steepest descent method (black) and conjugate gradient method (red) when minimizing such a quadratic function. The contour of the objective function is shown in different colors, with the minimum at the bottom center of the narrow “valley”. The plot also depicts the gradients at different point with blue arrows. In the two-dimensional case, one might have hoped that the gradient descent will arrive at the minimum in two steps, where the first step takes you to the bottom of the valley, and the second step directly down to the center. Steepest descent method instead, makes many small steps following the gradient directions. It has a very slow convergence rate, repeatedly searching in the same directions, even when the loss is a perfectly quadratic function. The reason is that with the line minimization rule, the gradients at two consecutive iterations are perpendicular to each other. In other words, you are making a right turn at each iteration, which are not likely to take you to minimum unless matrix \mathbf{A} is an identity matrix.

Algorithm 1.1 Conjugate Gradient Method.

```
1: Initialization:  $\mathbf{x}_0 = \mathbf{0}$ ,  $\mathbf{g}_0 = -\nabla f(\mathbf{x}_0)$ ,  $\mathbf{d}_0 = \mathbf{g}_0$ 
2: for  $k = 0$  to  $n - 1$  do
3:    $\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{d}_k^\top \mathbf{A} \mathbf{d}_k}$ 
4:    $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$ 
5:    $\mathbf{g}_{k+1} = \mathbf{g}_k - \alpha_k \mathbf{A} \mathbf{d}_k$ 
6:    $\beta_{k+1} = \frac{(\mathbf{g}_{k+1} - \mathbf{g}_k)^\top \mathbf{g}_{k+1}}{\mathbf{g}_k^\top \mathbf{g}_k}$ 
7:    $\mathbf{d}_{k+1} = \mathbf{g}_{k+1} + \beta_{k+1} \mathbf{d}_k$ 
8: end for
```

Conjugate gradient method, on the other hand, can solve the problem with at most n iterations, where n is the number of variables in \mathbf{x} . For the example we plot in figure 1.6, at most 2 steps are required, as depicted in the red arrow. Given a positive definite matrix \mathbf{A} , a set of nonzero vectors $\mathbf{d}_0, \dots, \mathbf{d}_k$ is conjugate (\mathbf{A} -orthogonal), if

$$\mathbf{d}_i^\top \mathbf{A} \mathbf{d}_j = 0, \quad \forall i \text{ and } j \text{ such that } i \neq j. \quad (1.13)$$

The conjugate directions are linearly independent, therefore a set of n conjugate directions will span the n dimensional space where the minimum \mathbf{x}^* lies in. If we set the update rule as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$$

where $\mathbf{d}_0 = -\nabla f(\mathbf{x}_0)$, and \mathbf{d}_k is conjugate to $\mathbf{d}_0, \dots, \mathbf{d}_{k-1}$, and find the stepsize α_k by line minimization rule, then we can also prove that

$$\mathbf{x}_{k+1} = \arg \min_{\mathbf{x} \in M^k} f(\mathbf{x}) \quad (1.14)$$

where M^k is the subspace spanned by $\mathbf{d}_0, \dots, \mathbf{d}_k$. Please refer to [13] page 132 for the proof. Therefore, at iteration n , we would have found the minimum, $\mathbf{x}_n = \arg \min_{\mathbf{x} \in \mathcal{R}^n} f(\mathbf{x}) \equiv \mathbf{x}^*$. Algorithm 1.1 shows the pseudocode for the conjugate gradient method.

1.3.2 Augmented Lagrangian Method for Constrained Optimization

Some of our problem formulations involve constraints. In this section, we are going to briefly review the augmented Lagrangian method for constrained optimization problem of the following form:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & h_i(\mathbf{x}) = 0, \quad i = 1, \dots, m \\ & g_j(\mathbf{x}) \leq 0, \quad j = 1, \dots, r \end{aligned} \tag{1.15}$$

where the objective function $f : \mathcal{R}^n \rightarrow \mathcal{R}$, the equality constraints $h_i : \mathcal{R}^n \rightarrow \mathcal{R}, i = 1, \dots, m$, and the inequality constraints $g_j : \mathcal{R}^n \rightarrow \mathcal{R}, j = 1, \dots, r$ are continuously differentiable.

The constrained optimization problem can be well approximated with an unconstrained optimization of the augmented lagrangian function

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}} L_c(\mathbf{x}, \mathbf{z}, \lambda, \mu) = f(\mathbf{x}) & + \sum_{i=1}^m \left\{ \lambda_i h_i(\mathbf{x}) + \frac{c}{2} h_i^2(\mathbf{x}) \right\} \\ & + \sum_{j=1}^r \left\{ \mu_j (g_j(\mathbf{x}) + z_j^2) + \frac{c}{2} (g_j(\mathbf{x}) + z_j^2)^2 \right\} \end{aligned} \tag{1.16}$$

provided that either:

- 1) λ and μ are close to the Lagrangian multipliers; or
- 2) The penalty parameter c is large.

Here $z_j^2, j = 1, \dots, r$ are the additional variables introduced to handle inequality constraints.

There are several other popular techniques for constrained optimization. For example, the gradient projection method [127] is widely used in problems involves constraints of convex set. It adds to the steepest descent method for unconstrained optimization an additional projection step to project the search direction to the feasible set. However, in order for

these methods to make practical sense, it is necessary that the projection operation is fairly simple, which in turn means that the constraints can only take simple structure, such as bounded or linear constraints. (In)exact Penalty method is another alternative. It follows the same philosophy of augmented Lagrangian method, that is, to transform a constrained problem into an unconstrained one, and then solve with techniques we introduced in previous section. It adds any violation in the constraints as additional penalty terms to the objective function. The problem with penalty method is that in order to recover a feasible solution to the original constrained problem, the penalty factors c for each active constraints have to go to infinity. As c increases, the condition number of the Hessian matrix increases as well, causing the unconstrained formulation to become difficult to solve.

The advantage of augmented Lagrangian method lies in the fact that it is workable even if the penalty factor c is not increased to ∞ , thus alleviates the ill-conditioning problem and is in general more reliable than penalty method.

update rule. A good heuristic for updating the penalty factor c is to start with a relatively small value c^0 and increase it slightly whenever the constraints are violated. The Lagrangian multipliers are updated as

$$\lambda_i^{k+1} = \lambda_i^k + c^k h_i(\mathbf{x}^k) \quad \mu_j^{k+1} = \mu_j^k + c^k \max(0, g_j(\mathbf{x}^k, \mu^k, c^k)) \quad (1.17)$$

In our experiments, we found that it is important to roughly tune the initial penalty factor c_0 at some small values so that the augmented unconstrained problem is solvable. Starting from a too large penalty factor often results in ill-conditioning in the later iterations.

1.3.3 Least-Squares Problem

In this section, we are going to review the least-squares problem which serves as building blocks for a couple of our algorithms. Least-squares is arguably the most well known subclass of convex optimization problems. A least-squares problem is unconstrained, and it takes an objective which is a sum of squared residuals of the form $\mathbf{a}^\top \mathbf{x} - b$,

$$\min_{\mathbf{x} \in \mathcal{R}^d} f(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|^2 = \sum_{i=1}^n (\mathbf{a}_i^\top \mathbf{x} - b_i)^2 \quad (1.18)$$

where $\mathbf{A} \in \mathcal{R}^{n \times d}$, and \mathbf{a}_i^\top corresponds to the rows of \mathbf{A} . The vector $\mathbf{x} \in \mathcal{R}^d$ is the variables we optimize over.

We can rewrite the objective function in eq. (1.18) as

$$\begin{aligned} \min_{\mathbf{x}} \quad & (\mathbf{Ax} - \mathbf{b})^\top (\mathbf{Ax} - \mathbf{b}) \\ & = \mathbf{x}^\top \mathbf{A}^\top \mathbf{Ax} - 2\mathbf{x}^\top \mathbf{A}^\top \mathbf{b} + \mathbf{b}^\top \mathbf{b} \end{aligned} \quad (1.19)$$

Take the derivative of (1.19) and set it to zero, we have

$$\mathbf{A}^\top \mathbf{Ax} = \mathbf{A}^\top \mathbf{b} \quad (1.20)$$

In other words, solving the least-squares problem is equivalent to solve the system of linear equations in (1.21), which has closed form solution

$$\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}. \quad (1.21)$$

Solving the least-squares problem has a time complexity of $O(d^2n)$ with a known constant. A variety of efficient and reliable implementations, such as gaussian elimination method, for solving the linear system of equations are available. Running a problem with $d = 5,000$ and $n = 100,000$ on a desktop with dual 6 cores Intel i7 cpus with 2.66Ghz takes seconds.

The least-squares problem is the basis for regression analysis, optimal control, and many parameter estimation and data fitting methods [25]. It can be further generalized to slightly different formats without breaking the closed form solution. For example, we can include different weights for different rows of \mathbf{A} , which corresponds to having different costs for different examples in the regression analysis or data fitting.

$$\min_{\mathbf{x}} \quad f_\omega(\mathbf{x}) = \sum_{i=1}^n \omega_i (\mathbf{a}_i^\top \mathbf{x} - b_i)^2 \quad (1.22)$$

where the weights $\omega_1, \dots, \omega_n$ are non-negative. Let $\Omega = [\omega_i]_{ii}$ denote a diagonal matrix with the weights ω_i 's on the diagonal, we can then rewrite the weighted version of the least-squares problem as

$$\min_{\mathbf{x}} \quad f_\omega(\mathbf{x}) = (\mathbf{Ax} - \mathbf{b})^\top \Omega (\mathbf{Ax} - \mathbf{b}), \quad (1.23)$$

which has analytical solution as $\mathbf{x} = (\mathbf{A}^\top \Omega \mathbf{A})^{-1} \mathbf{A}^\top \Omega \mathbf{b}$.

We can also add additional terms that regularize the variables \mathbf{x} to the objective function. For example, when a positive multiple of the sum of squares of the variables are added, we have

$$\min_{\mathbf{x}} f_\lambda(\mathbf{x}) = \sum_{i=1}^n (\mathbf{a}_i^\top \mathbf{x} - b_i)^2 + \lambda \|\mathbf{x}\|^2. \quad (1.24)$$

where λ is a positive factor penalizing large values in \mathbf{x} . The new objective function is commonly referred to as ridge regression. Different regularization reduces to Maximum likelihood estimation of the variables \mathbf{x} with different priors. For example, ridge regression corresponds to estimation of \mathbf{x} with a gaussian prior; and L_1 regularization corresponds to Laplace prior.

Note that the closed form solution no longer stands in the L_1 regularization case. However, with the introduction of some auxiliary variables, we can solve the problem within a few iterations, where at each iteration an analytical solution is available. We refer interested readers to [25].

1.3.4 Block Coordinate descent (Alternating Optimization)

Another optimization technique we used quite often in our algorithms is the (block) coordinate descent method, also known as the alternating optimization. It partitions the variables of the optimization problem $\mathbf{x} \in \mathcal{R}^n$ into several exclusive subsets $\mathbf{x} = (\mathbf{x}_{(1)}; \dots; \mathbf{x}_{(t)})$, where $\mathbf{x}_{(i)} \in \mathcal{R}^{n_i}$, satisfying $\sum_{i=1}^t n_i = n$. It minimizes the objective function $f(\mathbf{x})$ jointly over all the variables in \mathbf{x} by iteratively carrying out alternating restricted minimization over each subset of variables $\mathbf{x}_{(1)}; \dots; \mathbf{x}_{(t)}$. In other words, given the current iterate $\mathbf{x}^k = (\mathbf{x}_{(1)}^k; \dots; \mathbf{x}_{(i-1)}^k; \mathbf{x}_{(i)}^k; \mathbf{x}_{(i+1)}^k; \dots; \mathbf{x}_{(t)}^k)$, it generates the next iterate \mathbf{x}^{k+1} as

$$(\mathbf{x}_{(1)}^k; \dots; \mathbf{x}_{(i-1)}^k; \mathbf{x}_{(i)}^{k+1}; \mathbf{x}_{(i+1)}^k; \dots; \mathbf{x}_{(t)}^k),$$

where

$$\mathbf{x}_{(i)}^{k+1} = \arg \min_{\mathbf{z} \in \mathcal{R}^{n_i}} f(\mathbf{x}_{(1)}^k; \dots; \mathbf{x}_{(i-1)}^k; \mathbf{z}; \mathbf{x}_{(i+1)}^k; \dots; \mathbf{x}_{(t)}^k) \quad (1.25)$$

That is, the variables from all the subsets except subset i is clamped to the best values found in previous iterations, and the cost function is minimized w.r.t each of the block coordinates $\mathbf{x}_{(i)}^k$. The same process is repeated for every block in cyclic order until convergence. The method is extremely useful if the minimization problem we have in eq. (1.25) is substantially easier than jointly minimizing the loss over all the variables at once. Note that, this approach can get stuck at some local minimum. But it converges to the global minima if the cost function is jointly convex over all the coordinates.

Chapter 2

Single View Co-training

In this chapter, we are going to explicate our first strategy, single view co-training, for learning with weak supervision or domain adaptation. The co-training algorithm of Blum and Mitchell [23] was originally designed for multi-view data. Section 2.1 details *Pseudo Multi-view Co-training* (PMC). PMC generalizes co-training to the more common single view data with an automatic feature decomposition scheme. The rote learning procedure employed in co-training is a natural fit to learn from weakly labeled data, by cherry-picking the instances that resemble the training examples. We demonstrate on challenging tasks that PMC improves the generalization performance substantially with the help of weakly labeled data. Section 2.2 details *Co-training for Domain Adaptation* (CODA). CODA adds to PMC another feature selection component to address the missing feature problem in domain adaptation. Combining these two components, CODA is able to slowly adapt the training set from source to target distribution, both data-wise and feature-wise. CODA significantly out-performs the state-of-the-art on the 12-domain benchmark data set of Blitzer et al. [19]. Indeed, over a wide range (65 of 84 comparisons) of target supervision CODA achieves the best performance.

2.1 Pseudo Multi-view Co-training

In this section, we present Pseudo Multi-view Co-training (PMC). PMC is a variant of the Co-training algorithm [23] for multi-view data. In co-training, one trains two classifiers, one on each view, that teach each other with the most confident predictions of the unlabeled data. PMC extends co-training to learning scenarios without an explicit multi-view representation.

Based on a theoretical analysis of Balcan et al. [5], we introduce a novel algorithm that splits the feature space during learning, explicitly to encourage co-training to be successful. We demonstrate the efficacy of our proposed method in a weakly-supervised setting on the challenging Caltech-256 object recognition task, where we improve significantly over previous results [11] in almost all training-set size settings.

The section is organized as follows. Section 2.1.2 recaps the setup in learning with weak supervision. Section 2.1.3 describes the original co-training algorithm and explains why it is particularly suitable for learning on weakly-labeled data. We survey several theoretical work on relaxing the assumptions in co-training and related work on generalizing co-training to single view data. Section 2.1.4 details the PMC algorithm itself. The algorithm is based on the theoretical work of Balcan et al. [5]. Different from most of the previous approaches, which decompose the feature space in a preprocessing step, we learn the feature decomposition along with the two classifiers, explicitly to satisfy the conditions for co-training to succeed. Section 2.1.5 extends the framework to multi-class settings. In Section 2.1.6, we report the performance of PMC on the Caltech-256 object recognition task, using images retrieved from BingTM as weakly labeled data.

2.1.1 Introduction

Co-training [23] is an approach to semi-supervised learning [36, 174] which assumes that the available data is represented with two views. In its original formulation, these two views must satisfy two conditions: 1) each one is sufficient to train a low-error classifier and 2) both are class-conditionally independent. Given a datasets of two such views (representations), co-training trains two classifiers, one on each view. It then utilizes unlabeled data by adding the most confident predictions of each classifier to the training set of the other classifier – effectively letting the classifiers “teach each other”. Blum and Mitchell [23] show drastic improvements on data sets where the multi-view assumptions naturally hold. Co-training and its variants have been applied to many applications across computer science and beyond [45, 66, 116, 98, 27, 33].

However in many learning scenarios, the available data does not originate from two explicitly different sources. For example, in the MNIST handwritten digit dataset we introduced in

chapter 1, pixel values extracted from each 28×28 image of digit form a *single* feature vector of 784 real values. It is unclear how one can easily split the features into two subsets that satisfies the class-conditionally independent assumption of co-training. One might also be faced with assorted features that were obtained through various means. For example, in the medical domain features might correspond to different examinations. It is hard to guarantee that each one of them is sufficiently informative about the patient’s condition. Meeting the class-conditionally independent assumption is even more unlikely.

We extend co-training to this more common single-view setting (features obtained through various means can be combined into a single view representation). We utilize recent advances in learning theory that have significantly weakened the strong assumptions of co-training. Most notably, Balcan et al. [5] proved that the class-conditional independence assumption is unnecessarily strong and that a weaker *expanding* condition on the underlying distribution of the multi-view data is sufficient for iterative co-training to succeed. We propose a novel feature decomposition algorithm, which automatically divides the features of a single-view data set into two mutually exclusive subsets – thereby creating a pseudo-multi-view representation for co-training. This feature division is learned explicitly to satisfy the necessary conditions to enable successful co-training. We derive a single optimization problem, which divides the feature space, trains both classifiers and enforces an approximation of the ϵ -expanding condition [5] through hard constraints. We refer to our algorithm as *Pseudo Multi-view Co-training* (PMC).

Our broadening of the scope of co-training is particularly useful for weakly supervised learning scenarios. Through the success of web-search, it is now possible to obtain large quantities of data for almost any topic or class description (*e.g.* through automated image search or wikipedia lookups). Often, however, only a small fraction of the retrieved search results are truly relevant to someone’s learning task. As co-training explicitly cherry-picks data instances with similar characteristics as the labeled training data, it is naturally suited for learning with such noisy (weak) labels. We demonstrate this capability by effectively utilizing weakly labeled image-search results to improve the classification accuracy on the Caltech 256 object recognition data set – surpassing previously published results on the same task by Bergamo and Torresani [11].

2.1.2 Recap of Learning with Weak Supervision

Getting a large quantity of good labels is often expensive and time-consuming. On the other hand, obtaining weakly labeled data, a candidate pool containing both good and bad labels, has becoming almost “cost-free” with the booming era of web search. Nowadays, we can acquire enormous amount of data for any concept or class description by simply performing search and crawling the returned results.

Figure 2.1 shows a Venn diagram that illustrates the setup we have in learning with weak supervision. Let $\mathcal{X} \in \mathcal{R}^d$ denote the feature space, and \mathcal{Y} the output space. Let \mathcal{D}' denote the joint distribution of the weakly labeled data on $\mathcal{X} \times \mathcal{Y}$, \mathcal{D} the testing distribution. As shown in the figure, the support of the distribution \mathcal{D}' is a superset of that of distribution \mathcal{D} . In other words, examples that are in the support of the testing distribution, *i.e.*, the examples resembling the testing data, can be drawn from \mathcal{D}' as well. But \mathcal{D}' also generates examples that do not follow the testing distribution, *i.e.*, the “noisy” examples.

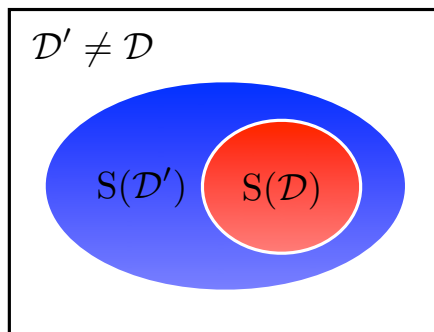


Figure 2.1: Venn diagram on learning with weak supervision.

Notation. We have access to a small set of labeled examples $L = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$ sampled i.i.d. from the testing distribution \mathcal{D} , as well as another much larger set of weakly labeled examples $L' = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}, m \gg n$ sampled i.i.d. from another distribution \mathcal{D}' . We want to find a hypothesis $h \in \mathcal{H}$ using the labeled data from both L and L' , so that the generalization performance improves over a hypothesis h learned on L solely.

2.1.3 Background and Related Work

Co-training [23] is among the first of a few successful semi-supervised learning algorithms developed. Let $\mathcal{X} \subseteq \mathcal{R}^d$ denote the feature space, and \mathcal{Y} the output space. Recall that in semi-supervised learning, we have a small set of labeled data i.i.d sampled from an unknown distribution \mathcal{D} , $L = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$, as well as another set of unlabeled inputs $U = \{\mathbf{x}_{n+1}, \dots, \mathbf{x}_{n+m}\}$ drawn from the same distribution. The labels of the examples in U are unknown though.

Co-training assumes that the feature space can be partitioned into two subspaces, *i.e.*, $\mathcal{X} = \mathcal{X}^1 \times \mathcal{X}^2$. \mathcal{X}^1 and \mathcal{X}^2 are commonly referred to as the two views of the inputs. The two views must satisfy two conditions: 1) Both have to be *sufficient* within the given hypothesis class, \mathcal{H}^1 on \mathcal{X}^1 and \mathcal{H}^2 on \mathcal{X}^2 , for correct classifications; In other words, there exist two hypotheses $h^1 \in \mathcal{H}^1$ and $h^2 \in \mathcal{H}^2$ having low error on inputs from feature space \mathcal{X}^1 and \mathcal{X}^2 respectively. Note that the two hypothesis classes are usually of the same type, but since they are trained on different feature spaces, we denote them differently for clarity. 2) They need to be *class-conditionally independent*, *i.e.*, for a given input $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2) \in \mathcal{X}^1 \times \mathcal{X}^2$ with label $y \in \mathcal{Y}$, $p(\mathbf{x}^1, \mathbf{x}^2|y) = p(\mathbf{x}^1|y)p(\mathbf{x}^2|y)$. Under these assumptions, co-training can boost any initial weak learner to arbitrarily high accuracy using *unlabeled* examples only, provided that the target class is PAC learnable from random classification noises [23].

The fundamental idea behind co-training is what Blum and Mitchell [23] describe as “rote learning” on the unlabeled data set. Initially, two classifiers h^1, h^2 are trained on the labeled set L , both on their respective views. The two classifiers are then evaluated on the unlabeled set U . For each classifier, the examples on which it is most confident are removed from U and added to L to “teach” the other classifier at the next iteration. Both classifiers are now re-trained on the expanded labeled data set and the procedure is repeated until some stopping criterion is met. By carrying out this “rote learning” procedure, co-training can bootstrap from a small labeled “seed” set and iteratively improve its performance with the help of unlabeled data. Under the semi-supervised learning setting, the unlabeled data helps as it reduces the version space size [110]. A version space refer to the subset of all hypotheses in \mathcal{H} that are consistent with the observed training examples. Here, the two classifiers have to agree upon not only the small set of labeled examples in L , but also on the much larger set of unlabeled data U .

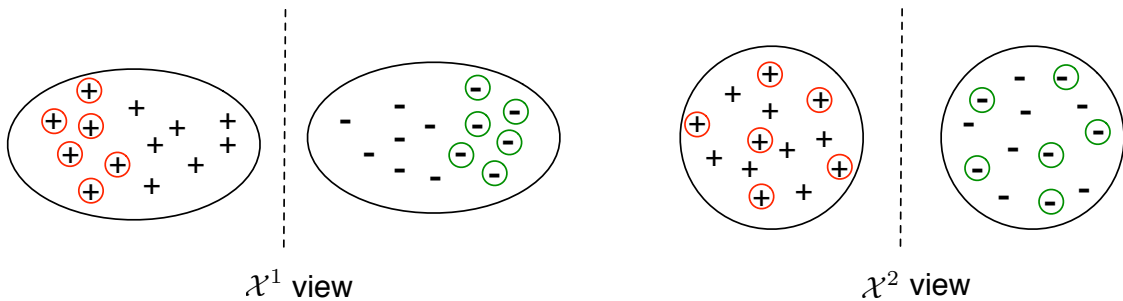


Figure 2.2: Conditionally-independent assumption of Co-training.

The first assumption that both views are sufficiently informative for correct classification ensures that we can trust the “labels” automatically annotated by the two classifiers h^1 and h^2 . The second assumption that the two views are conditionally independent ensures that the examples that one classifier is confident about are i.i.d sampled examples for another classifier, as shown in figure 2.2. The examples h^1 is confident about are circled in red (positive predictions) and green (negative predictions). These confident predictions, while concentrated in regions further away from the decision boundary in \mathcal{X}^1 (figure 2.2 left), are scattered in view \mathcal{X}^2 (figure 2.2 right). Under these assumptions, provided that the target function $f^1(\mathbf{x}^1)$, $\mathbf{x}^1 \in \mathcal{X}^1$ and $f^2(\mathbf{x}^2)$, $\mathbf{x}^2 \in \mathcal{X}^2$ are PAC learnable from random classification noise, one can conclude that co-training can succeed with unlabeled data, given two initially useful predictors $h^1(\mathbf{x}^1)$ and $h^2(\mathbf{x}^2)$. Please refer to [23] for the proof.

Compared to the various assumptions made in other semi-supervised learning algorithms, such as the cluster or manifold assumptions, the multi-view assumption of co-training is less affected by the fact that the weakly-labeled data comes from a different distribution than the test data. Furthermore, as co-training explicitly cherry-picks data instances with similar characteristics as the labeled training data (which comes from the same distribution as the test inputs), it is naturally suited for learning with such noisy (weak) labels.

Unfortunately, finding two views satisfying these conditions is by no means an easy task. As pointed out by Nigam and Ghani [116], the assumption that the two views are class conditionally independent can easily be violated in practice. Abney [2] claims that the assumption is unnecessarily strong and instead, a weaker rule independence assumption, which allows the two views to deviate from fully class-conditionally independent by a certain amount, suffices. Balcan et al. [5] further weaken this requirement. Intuitively, co-training

expands its confidence set by letting the two classifier teach each other iteratively. For the two classifiers to be able to teach each other, they must make confident predictions on different subsets of the unlabeled data. Balcan et al. [5] formalize this condition as a concept of ϵ -*expandability*, which we will explain in more detail in section 2.1.4.

Related Work

Despite advances in theoretical analyses of the algorithm, the applicability of co-training has been largely dependent on domain experts manually creating multi-views from the data and feeding them into the algorithm. There is some previous work on automatically decomposing the feature space of a single view data for co-training. Broadly speaking, these approaches fall into two categories: random feature splitting and greedy approach. Nigram and Ghani [116] perform an extensive empirical study on co-training. For datasets without natural feature split, they create an artificial split by randomly breaking the feature set into two subsets. Chan et al. [33] also investigate the feasibility of random feature splitting and apply co-training to email-spam classification. Abney [2] proposes a greedy agreement algorithm that iteratively adds unit rules that agree on unlabeled data to build two views for co-training. Zhang and Zheng [171] propose to decompose the feature space by first applying PCA and then greedily dividing the orthogonal components to minimize the energy diversity of the two feature sets.

Instead of enforcing the input to be represented in two views on which two classifiers of the same type can be learned, Goldman and Zhou [70] take a slightly different approach: the whole feature set is used, but instead two classifiers of different types are trained. The high confident predictions of one classifier, identified with a set of statistical tests, are used to teach the other classifier and vice versa. Later they generalize this framework to include multiple learners [172], so that a democratic voting strategy among the different learners is employed to increase the accuracy on the predictions. For methods of this kind to work, it is essential that the different learners employed in the framework are diverse and make confident predictions on different instances from U . However in general, predictions from classifiers on the same feature space (*e.g.*, support vector machines [46] and logistic regression [115]) are usually highly correlated.

2.1.4 Pseudo Multi-view Co-training

The algorithm we propose shares a common goal of the ones introduced in section 2.1.3, that is, to extend co-training to the scenarios when the two views $\mathcal{X}^1, \mathcal{X}^2$ are not known. Different from most of the previous approaches, which decompose the feature space in a preprocessing step, we learn the feature decomposition along with the two classifiers, explicitly to satisfy the conditions for co-training to succeed. Our algorithm is based on the theoretical analysis of ϵ -expandability from Balcan et al. [5], which we briefly introduced in previous section. In this section, we are going to first formalize this condition, and then describe the Pseudo Multi-view Decomposition (PMD) algorithm we come up with to create pseudo multi-views from single view data.

ϵ -expandability

Similar to the original theoretical analysis of co-training in [23, 2], the analysis from Balcan et al. [5] also assumes that the inputs consist of two views, *i.e.*, $\mathcal{X} = \mathcal{X}^1 \times \mathcal{X}^2$. Let $h^1 \in \mathcal{H}^1$ and $h^2 \in \mathcal{H}^2$ be the two classifiers, trained on the two views $\mathcal{X}^1, \mathcal{X}^2$ respectively. Each input \mathbf{x} contains the features from both views, $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2)$. The unlabeled data U sampled from the distribution \mathcal{D} on \mathcal{X} can therefore be partitioned into two sets. Let $U^1 = \{\mathbf{x}_{n+1}^1, \dots, \mathbf{x}_{n+m}^1\}$ denote the set of unlabeled data from view \mathcal{X}^1 , and $U^2 = \{\mathbf{x}_{n+1}^2, \dots, \mathbf{x}_{n+m}^2\}$ the corresponding inputs from view \mathcal{X}^2 . Let us denote the subsets of U^1, U^2 on which these two classifiers are confident as C^1, C^2 respectively. For any $S \subseteq U$, let \mathbf{S}^i denote the event that an input $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2) \in S$ satisfies $\mathbf{x}^i \in C^i$. We can express the probability of an instance in S to be classified confidently by both classifiers as $\Pr(\mathbf{S}^1 \wedge \mathbf{S}^2)$, by *exactly one* of the two classifiers as $\Pr(\mathbf{S}^1 \oplus \mathbf{S}^2)$ and by none as $\Pr(\overline{\mathbf{S}^1} \wedge \overline{\mathbf{S}^2})$. The ϵ -expanding condition can then be stated as follows:

Definition 2.1. [5] \mathcal{D} is ϵ -expanding with respect to the hypothesis class $\mathcal{H}_1 \times \mathcal{H}_2$ if for any $S \subseteq U$ and any two classifiers $h^1 \in \mathcal{H}^1$ and $h^2 \in \mathcal{H}^2$, the following statement holds

$$\Pr(\mathbf{S}^1 \oplus \mathbf{S}^2) \geq \epsilon \min[\Pr(\mathbf{S}^1 \wedge \mathbf{S}^2), \Pr(\overline{\mathbf{S}^1} \wedge \overline{\mathbf{S}^2})].$$

Intuitively, the condition ensures that with high probability there are data instances in the unlabeled set for which *exactly one* of the two classifiers is confident. These instances can then be added to the labeled set to teach the classifier which is uncertain about them. It is shown that the conditional independence of [23] and the weak rule independence assumption of [2] can be seen as extreme cases of the ϵ -expanding condition. In other words, ϵ -expandability is a much less restrictive assumption for co-training.

Let h_0^1 and h_0^2 denote the two initial classifiers trained on the small set of labeled data L . Let $\rho_0 = \Pr(\mathbf{S}_0^1 \oplus \mathbf{S}_0^2)$ denote the probability of an instance drawn from \mathcal{D} to be classified confidently by either of the two classifiers.

Proposition 2.1. *Let ϵ_N and δ_N be the final desired accuracy and confidence parameters. Given that the data distribution \mathcal{D} satisfies the ϵ -expanding condition, running co-training for $N = O(\frac{1}{\epsilon} \log \frac{1}{\epsilon_N} + \frac{1}{\epsilon} \cdot \frac{1}{\rho_0})$ rounds, each time learning two hypothesis classifiers with accuracy and confidence parameters set to $\frac{\epsilon_N}{8}$ and $\frac{\delta_N}{2N}$, achieves error rate ϵ_N with probability $1 - \delta_N$.*

Intuitively, given the ϵ -expanding condition, the set of examples on which at least one of the two classifiers is confident grows by a factor proportional to ϵ after each iteration. Please refer to [5] for a proof sketch.

Pseudo Multi-view Decomposition

Based on the analysis, we can conclude that if the distribution \mathcal{D} is ϵ -expanding, and the two classifiers are never “confident but wrong”, co-training will succeed. We translate the conclusion into three conditions: 1) both classifiers must perform well on the labeled data; 2) they must be trained on strictly different features; 3) together they are likely to satisfy Balcan et al. [5]’s condition of ϵ -expandability. In this section, we describe how to learn two classifiers on a single view \mathcal{X} to satisfy these three conditions. For now we focus on binary problems and set $\mathcal{Y} = \{+1, -1\}$. We extend to multi-class problems in section 2.1.5.

Loss function. Co-training itself makes no assumption on the exact type of classifiers used during the “rote-learning” procedure. For now we only consider linear classifiers, $h_{\mathbf{u}}(\mathbf{x}) = \text{sign}(\mathbf{u}^\top \mathbf{x} + b)$ with weight vector \mathbf{u} and bias b . To simplify notation we drop the bias b and

assume that a constant 1 is attached as an additional dimension to each input \mathbf{x}_i , which is *not split* between the two classifiers. A classifier $h_{\mathbf{u}}$ is trained by minimizing the log-loss over the data set L :

$$\ell(\mathbf{u}; L) = \sum_{(\mathbf{x}, y) \in L} \log \left(1 + e^{-\mathbf{u}^T \mathbf{x} y} \right). \quad (2.1)$$

Our framework is agnostic to the specific choice of loss-function, however we choose logistic regression [115] as it explicitly models the probability of labels y conditioned on the input \mathbf{x} , which provides a natural measure of classifier-confidence. This is mostly a matter of convenience and not a requirement — alternative loss functions such as support vector machines [46] would be equally applicable.

For co-training to work we need two classifiers, whose weight vectors we denote by \mathbf{u} and \mathbf{v} . To meet the first condition that “both classifiers must perform well on the labeled set”, we train these two jointly, and make sure that *both* suffer low loss by minimizing the maximum of the two,

$$\min_{\mathbf{u}, \mathbf{v}} \max [\ell(\mathbf{u}; L), \ell(\mathbf{v}; L)]. \quad (2.2)$$

As eq. (2.18) is non-differentiable we introduce a slight relaxation and replace the max term with a more manageable softmax. The optimization then becomes

$$\min_{\mathbf{u}, \mathbf{v}} \log \left(e^{\ell(\mathbf{u}; L)} + e^{\ell(\mathbf{v}; L)} \right), \quad (2.3)$$

which is continuous and differentiable.

Feature Decomposition. A crucial aspect of co-training is that the two classifiers are trained on different views of the data set. Unconstrained, the minimization problem in (2.3) would result in two identical weight vectors $\mathbf{u} = \mathbf{v}$, *both* using up all the features. Instead, we want the two classifiers to divide up the feature space so that each feature can only be used by one of the two. For linear classifiers parameterized by a weight vector \mathbf{u} , dropping a feature i is equivalent to set the corresponding weight \mathbf{u}_i to zero. Therefore, we can fulfill the second requirement that “*both* must be trained on strictly different features” by constraining that for each feature i , at least one of the two classifiers must have a zero weight in the i^{th}

dimension. We can write this constraint as

$$\forall i, 1 \leq i \leq d, \quad \mathbf{u}_i \mathbf{v}_i = 0. \quad (2.4)$$

For a feature space $\mathcal{X} \in \mathcal{R}^d$, d constraints should be enforced. Although correct, this formulation is unnecessarily hard to optimize and can result in numerical instabilities. Instead, we square both sides and sum over all features to obtain the following constraint:

$$\sum_{i=1}^d \mathbf{u}_i^2 \mathbf{v}_i^2 = 0. \quad (2.5)$$

It is important to point out that any solution to (2.5) strictly implies (2.4).

ϵ -Expandability. The final condition is that the two classifiers must make confident predictions on different inputs from the unlabeled set U . For classifier $h_{\mathbf{u}}$, let $\hat{y} = \text{sign}(\mathbf{u}^\top \mathbf{x}) \in \{\pm 1\}$ denote the class prediction and $p(\hat{y}|\mathbf{x}; \mathbf{u}) = (1 + e^{-\mathbf{u}^\top \mathbf{x} \hat{y}})^{-1}$ its confidence. We define a binary confidence indicator function² as

$$c_{\mathbf{u}}(\mathbf{x}) = \begin{cases} 1 & \text{if } p(\hat{y}|\mathbf{x}; \mathbf{u}) > \tau \\ 0 & \text{otherwise} \end{cases} \quad (2.6)$$

where τ is a predefined confidence threshold.³ Let us define the subset of inputs on which classifier $h_{\mathbf{u}}$ is confident as $C_{\mathbf{u}} = \{\mathbf{x} \in X \mid c_{\mathbf{u}}(\mathbf{x}) = 1\}$, and $C_{\mathbf{v}}$ respectively. For any $S \subseteq X$, let $\mathbf{S}_{\mathbf{u}}$ denote the event that an input \mathbf{x} in S belongs to the confident set of $h_{\mathbf{u}}$, that is, $\mathbf{x} \in S \cap C_{\mathbf{u}}$. $\mathbf{S}_{\mathbf{v}}$ is defined analogously.

The ϵ -expanding condition from section 2.1.4 becomes

$$\Pr(\mathbf{S}_{\mathbf{u}} \oplus \mathbf{S}_{\mathbf{v}}) \geq \epsilon \min[\Pr(\mathbf{S}_{\mathbf{u}} \wedge \mathbf{S}_{\mathbf{v}}), \Pr(\overline{\mathbf{S}_{\mathbf{u}}} \wedge \overline{\mathbf{S}_{\mathbf{v}}})]. \quad (2.7)$$

As pointed out in [5], requiring that the ϵ -expanding condition holds for every pair of $C_{\mathbf{u}}$ and $C_{\mathbf{v}}$ might still be unnecessarily strict in practice. For our optimization, we relax it and only require that the expanding condition holds on average for the confident sets of the solution

²In our implementation, the 0-1 indicator was replaced by a very steep differentiable sigmoid function.

³In our implementation, τ was set to 0.8 across different experiments.

$h_{\mathbf{u}}, h_{\mathbf{v}} \in \mathcal{H}$. More explicitly we add the following constraint to our optimization:

$$\sum_{\mathbf{x} \in U} [c_{\mathbf{u}}(\mathbf{x})\bar{c}_{\mathbf{v}}(\mathbf{x}) + \bar{c}_{\mathbf{u}}(\mathbf{x})c_{\mathbf{v}}(\mathbf{x})] \geq \epsilon \min \left[\sum_{\mathbf{x} \in U} c_{\mathbf{u}}(\mathbf{x})c_{\mathbf{v}}(\mathbf{x}), \sum_{\mathbf{x} \in U} \bar{c}_{\mathbf{u}}(\mathbf{x})\bar{c}_{\mathbf{v}}(\mathbf{x}) \right] \quad (2.8)$$

Here, $\bar{c}_{\mathbf{u}}(\mathbf{x}) = 1 - c_{\mathbf{u}}(\mathbf{x})$ indicates that classifier $h_{\mathbf{u}}$ is *not* confident about input \mathbf{x} . Intuitively, the constraint in eq. (2.8) ensures that the total number of inputs in U that can be used for rote-learning as *exactly one* classifier is confident (LHS), is larger than the set of inputs which can *not* as *both* classifiers are already confident or *both are not* confident (RHS).

In summary, we want to learn two logistic regression classifiers, both with small loss on the labeled data set, while satisfying two constraints to ensure feature decomposition and ϵ -expandability. We combine eqs (2.3-2.8) as the following optimization problem, which we will later refer to as *Pseudo Multi-view Decomposition* (PMD):

$$\begin{aligned} & \min_{\mathbf{u}, \mathbf{v}} \quad (e^{\ell(\mathbf{u}; L)} + e^{\ell(\mathbf{v}; L)}) \\ & \text{subject to:} \\ & \quad (1) \quad \sum_{i=1}^d \mathbf{u}_i^2 \mathbf{v}_i^2 = 0 \\ & \quad (2) \quad \sum_{\mathbf{x} \in U} [c_{\mathbf{u}}(\mathbf{x})\bar{c}_{\mathbf{v}}(\mathbf{x}) + \bar{c}_{\mathbf{u}}(\mathbf{x})c_{\mathbf{v}}(\mathbf{x})] \geq \epsilon \min \left[\sum_{\mathbf{x} \in U} c_{\mathbf{u}}(\mathbf{x})c_{\mathbf{v}}(\mathbf{x}), \sum_{\mathbf{x} \in U} \bar{c}_{\mathbf{u}}(\mathbf{x})\bar{c}_{\mathbf{v}}(\mathbf{x}) \right] \end{aligned}$$

The objective guarantees that none of the loss functions is too high. The first constraint enforces that no feature is used with both classifiers. The last constraint explicitly models the ϵ -expandability condition. We optimize this constrained optimization problem with the augmented Lagrangian method [13] introduced in section 1.3.2.

Pseudo Multi-view Co-training

Finally, we use our feature decomposition method to apply iterative co-training on single-view data. We refer to the resulting algorithm as *Pseudo Multi-view Co-training* (PMC). A detailed pseudo-code implementation is presented in Algorithm 2.1.

At each iteration of PMC, we find the two linear classifiers parameterized by \mathbf{u}^* and \mathbf{v}^* by solving the PMD problem introduced in previous section. The two classifiers $h_{\mathbf{u}}$ and

Algorithm 2.1 Pseudo Multi-view Co-training.

- 1: Inputs: L and U .
 - 2: Initialize \mathbf{u} , \mathbf{v} and l .
 - 3: **repeat**
 - 4: Find \mathbf{u}^* , \mathbf{v}^* by optimizing PMD on L and U .
 - 5: Apply $h_{\mathbf{u}^*}$ and $h_{\mathbf{v}^*}$ on all elements of U .
 - 6: Move up-to l confident inputs from U to L .
 - 7: **until** No more predictions are confident
 - 8: Train final classifier h on L with all features \mathcal{X} .
 - 9: Return h
-

$h_{\mathbf{v}}$ “label” the unlabeled examples in U , and the confident predictions are moved from U to expand the label set L . The process continues until we can no longer find confident predictions.

Please note that there is another interesting difference between PMC and traditional co-training. Not only is there no pre-defined split of the features but the automatically-found split can vary between iterations.

2.1.5 Extension

So far, we restrict ourselves to binary problems. One way to extend PMC to multiclass problems is by training multiple binary classifiers, one for each class, using the one-versus-the-rest scheme. However, such an approach cannot capture the correlations between different classes. A more natural and efficient way is to construct a hypothesis considering all the classes at once.

Let us denote the label space as $\mathcal{Y} = \{1, 2, \dots, K\}$. Let $\mathbf{U} = [\mathbf{u}^1, \dots, \mathbf{u}^K] \in \mathcal{R}^{d \times K}$ and $\mathbf{V} = [\mathbf{v}^1, \dots, \mathbf{v}^K] \in \mathcal{R}^{d \times K}$ denote the parameters of the two classifiers. Then the log-loss of a classifier $h_{\mathbf{U}}$ over the data set L is defined as:

$$\ell(\mathbf{U}; L) = - \sum_{(\mathbf{x}, y) \in L} \log \frac{e^{\mathbf{x}^\top \mathbf{u}^y}}{\sum_k e^{\mathbf{x}^\top \mathbf{u}^k}}. \quad (2.9)$$

The confidence indicator function $c_{\mathbf{U}}(\mathbf{x}), c_{\mathbf{V}}(\mathbf{x})$ are defined as in (2.20) with the class prediction computed as

$$\hat{y} = \max_k (\mathbf{x}^\top \mathbf{u}^k), \quad (2.10)$$

and the confidence as

$$p(\hat{y}|\mathbf{x}; \mathbf{U}) = \frac{e^{\mathbf{x}^\top \mathbf{u}^{\hat{y}}}}{\sum_k e^{\mathbf{x}^\top \mathbf{u}^k}}. \quad (2.11)$$

We can also decompose the instance space by constraining that eq. (2.5) holds for all classes, that is,

$$\sum_{k=1}^K \sum_{i=1}^d (\mathbf{u}_i^k)^2 (\mathbf{v}_i^k)^2 = 0. \quad (2.12)$$

However, without additional regularization, eq. (2.12) would result in K different decompositions, one for each class. For the classifiers to be compatible, we want to ensure a consistent partition of the instance space $\mathcal{X} = \mathcal{X}^1 \times \mathcal{X}^2$ across different classes. More precisely, for each feature i , at least one of the two classifiers must have zero weights *across all classes*.

A similar problem emerges in the context of feature selection in multi-task settings [4, 118], when similar parameter sparsity patterns across different tasks needs to be imposed. An effective regularization is the group lasso, defined over a matrix \mathbf{U} as

$$\|\mathbf{U}\|_{2,1} = \sum_{i=1}^d \sqrt{\sum_{k=1}^K (\mathbf{U}_{ik})^2}. \quad (2.13)$$

Intuitively, eq. (2.13) enforces the l_1 norm on the l_2 -norms of all rows in \mathbf{U} , enforcing sparsity on a per-row level – effectively forcing $h_{\mathbf{U}}$ to pick a feature for all classes or none. We encourage readers to refer to [118] for an intuitive geometric interpretation of the regularization in (2.13).

We then combine the other two constraints with the regularized objective into the following optimization problem:

$\min_{\mathbf{U}, \mathbf{V}} \log (e^{\ell(\mathbf{U}; L)} + e^{\ell(\mathbf{V}; L)}) + \lambda(\ \mathbf{U}\ _{2,1} + \ \mathbf{V}\ _{2,1})$ <p>subject to:</p> <p>(1) $\sum_{k=1}^K \sum_{i=1}^d (\mathbf{u}_i^k)^2 (\mathbf{v}_i^k)^2 = 0$</p> <p>(2) $\sum_{\mathbf{x} \in U} [c_{\mathbf{U}}(\mathbf{x}) \bar{c}_{\mathbf{V}}(\mathbf{x}) + \bar{c}_{\mathbf{U}}(\mathbf{x}) c_{\mathbf{V}}(\mathbf{x})] \geq \epsilon \min [\sum_{\mathbf{x} \in U} c_{\mathbf{U}}(\mathbf{x}) c_{\mathbf{V}}(\mathbf{x}), \sum_{\mathbf{x} \in U} \bar{c}_{\mathbf{U}}(\mathbf{x}) \bar{c}_{\mathbf{V}}(\mathbf{x})]$</p>

Again, we solve this problem using augmented Lagrangian method.

2.1.6 Experimental Results

In this section we evaluate PMC empirically on an artificially created dataset for sanity check and a real-world dataset to showcase that PMC enables learning from weakly labeled data.

Paired Handwritten Digits

As a first test, we construct a data set with binary class labels for which a class-conditionally independent feature split exists, but is unknown to the algorithm. Each instance in the set is a pair of digits sampled from the USPS handwritten digits set. As shown in figure 2.3, if the class label is +1 (upper row), the left digit is uniformly picked from the set of *ones* and *twos* and the right digit is picked from the set of *fives* or *sixes*. For the negative class -1 (lower row), the left digit is a *three* or *four* and the right digit a *seven* or *eight*. Given the class-label, the identities of the two digits in the image are class-conditionally independent. For example, given that the label of an instance is positive, the probability of the left digit being 1 is $\frac{1}{2}$, no matter what the right digit is. We construct $m = 6000$ such instances. By design, a natural decomposition is to split the feature space into two views such that one covers the left digit and the other covers the right digit.

Feature Decomposition. In the first experiment, we test our feature decomposition framework, Pseudo Multi-view Decomposition (PMD) when a sufficient amount of labels are provided. Among the 6000 instances created, 2000 are labeled. In other words, $|L| = 2000, |U| = 4000$. We solve PMD loss for \mathbf{u} and \mathbf{v} , starting with a random initialization. Figure 2.4 shows the heat maps of \mathbf{u} and \mathbf{v} before and after training. We also report the log-loss of the two

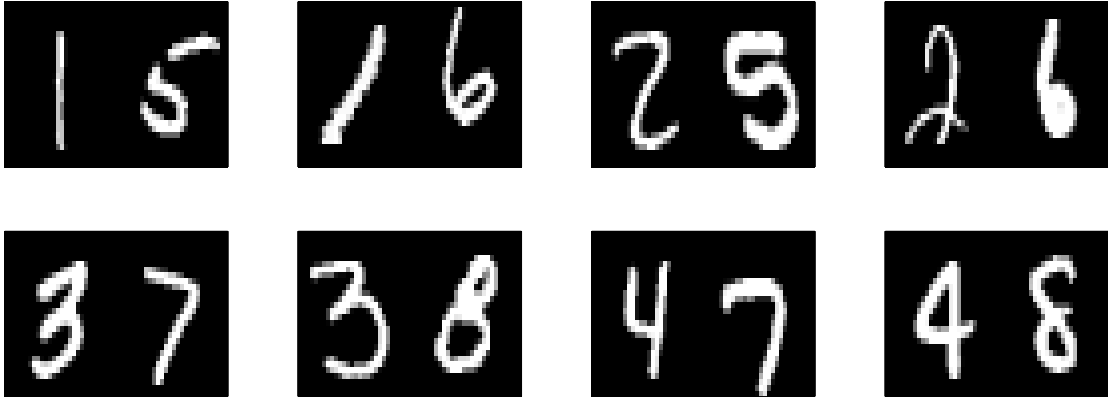


Figure 2.3: Sample images from the paired handwritten digit dataset. Upper row: positive examples; lower row: negative examples.

classifiers $h_{\mathbf{u}}$ and $h_{\mathbf{v}}$ on the labeled set L before and after training on the top of the heatmaps. The top two images show the heatmaps of randomly initialized \mathbf{u} and \mathbf{v} . Both weight vectors span over the entire image. Meanwhile, both suffer from high losses on the labeled data. The bottom two images in the figure show that once \mathbf{u}, \mathbf{v} are trained to minimize the loss function, constraining on (2.5) and (2.8), their non-zero weights are divided almost exactly into the two class-conditionally independent feature sets. In particular, classifier $h_{\mathbf{u}}$ takes the pixels of the left digit as its features, while classifier $h_{\mathbf{v}}$ uses only the right digit.

Co-Training. In the second experiment, we test our Pseudo Multi-view Co-training (PMC) algorithm on this multi-view data. Note that the algorithm itself is not aware of the existence of the two views. Only two labeled examples, one from each class, are given to the algorithm, *i.e.*, $|L| = 2$ and $|U| = 5998$. We run 12 sets of identical experiments with different labeled images and random initializations. In this setup we use the transductive setting. In other words, the test set coincides with the unlabeled set.

Table 2.1: Comparison of PMC and existing work on the paired handwritten digit dataset.

Test Err(%)	Baseline	RFS	ICA-RFS	PMC
Mean	18.64	13.78	12.22	3.99
STD	8.86	14.24	13.59	3.24

Table 2.1 summarizes the mean classification error and standard deviation of various algorithms on the paired digits dataset. We compare against three alternative methods: 1) the

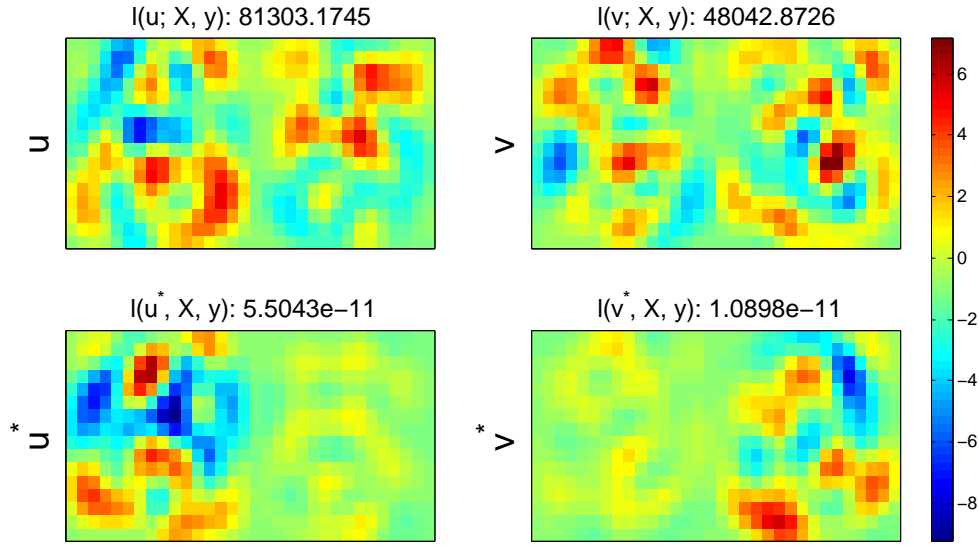


Figure 2.4: The heatmap of two randomly initialized weight vectors \mathbf{u}, \mathbf{v} (upper row) and $\mathbf{u}^*, \mathbf{v}^*$ learned with PMD (lower row) on the input space.

baseline, which trains logistic regression exclusively with the labeled instances; 2) co-training on two views obtained by random feature splitting (*RFS*); 3) co-training with random feature splitting where the features are pre-processed with Independent Component Analysis⁴ [84] (*ICA-RFS*). For *RFS* and *ICA-RFS*, 10 different random feature decompositions are considered for each run, and the average performance and the standard deviation across 120 runs are reported. As shown in Table 2.1, PMC achieves by far the lowest error with very small standard deviation. Random splitting method on the other hand, are very unreliable. We can observe a large fluctuation in accuracies at different runs. ICA is common technique to de-correlates non-gaussian data. However, we can see that it is not helpful in this case as independent is not equivalent to class-conditionally independent. Although the left and right digit is independent given the label, they are dependent when the labels are unknown. In other words, for a given input $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2) \in \mathcal{X}^1 \times \mathcal{X}^2$ with label $y \in \mathcal{Y}$, $p(\mathbf{x}^1, \mathbf{x}^2|y) = p(\mathbf{x}^1|y)p(\mathbf{x}^2|y)$, however, $p(\mathbf{x}^1, \mathbf{x}^2) \neq p(\mathbf{x}^1)p(\mathbf{x}^2)$. For example, knowing the left digit is a *one* brings up the probability of the right digit being a *five* or *six* to half, and brings down the probability of the right digit being a *seven* or *eight* to zero.

⁴We used the open-source implementation from <http://cs.nyu.edu/~roweis/kica.html>.

Figure 2.5 shows the heatmaps of the two weight vectors \mathbf{u} and \mathbf{v} in different PMC iterations. Confident predictions are moved from the unlabeled set U to the labeled set L in each PMC iteration, causing the loss function (2.3) and the constraint (2.8) to change between iterations. As a result, the automatically discovered feature splits vary between iterations. As more confident predictions were added to L , PMC gradually approximates the class-conditionally independent feature split from figure 2.4.

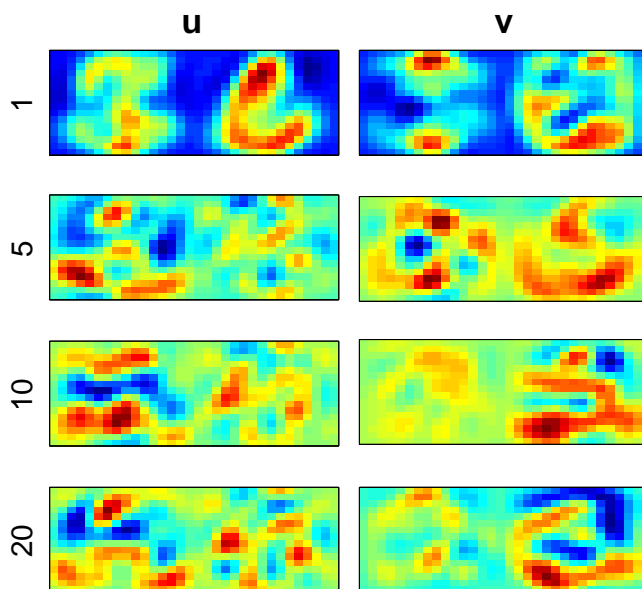


Figure 2.5: The heatmaps of \mathbf{u}, \mathbf{v} in 20 PMC iterations (1st iteration on top).

Figure 2.6 depicts the progress of the two classifiers $h_{\mathbf{u}}, h_{\mathbf{v}}$ in one run of the experiments. The magenta line indicates the test error of the baseline, a logistic regression model trained on the labeled data L only. The blue and green curves plot the errors of the two classifiers between iterations. During the “rote-learning” procedure, the two classifiers “learn” from each other, and finally converge to a almost perfect predictor. The red curve shows the test error of a classifier that combines the two weight vectors learned at each iteration $h_{\mathbf{u}+\mathbf{v}}$. As the two classifiers are confident on different instances, the combined classifiers has lower error in all iterations. Similar trends were observed in the other 11 runs.

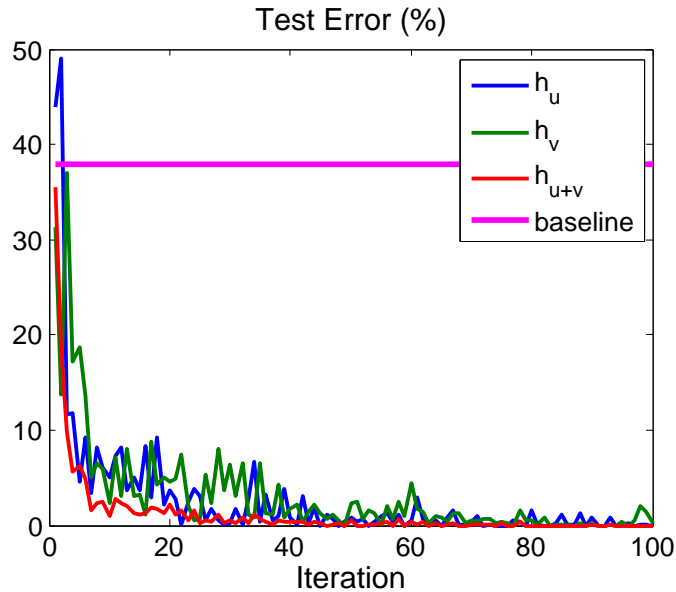


Figure 2.6: The progress made by the two classifiers h_u, h_v in different PMC iterations.

Caltech-256 with weakly labeled web images

We test the multi-class version of PMC on the challenging Caltech-256 object recognition task [73]. The dataset consists of images of objects from a set of 256 object categories. The task is to classify a new image into its object category. Additional images are retrieved from the BingTM image search engine using the category names as search queries, and used to help learning. These additional images are referred to as *weakly-labeled*, as the quality of the retrieved images is far from the original training data. As illustrated in figure 1.2, a large fraction of the retrieved images do not contain the correct object.

A recent work by Bergamo and Torresani [11] is able to utilize these additional data to improve learning by carefully down-weighting the web images and employing adequate regularization to suppress the noises introduced by irrelevant and low-quality images. As features, they use *classemes* [101], where each image is represented by a 2625 dimensional vector of predictions from various visual concept classifiers – including predictions on topics as diverse as “wetlands”, “ballistic missile” or “zoo”⁵. They achieved an improvement of 65% (27.1%

⁵A detailed list of the categories is available at http://www.cs.dartmouth.edu/~lorenzo/projects/classemes/classeme_keywords.txt.

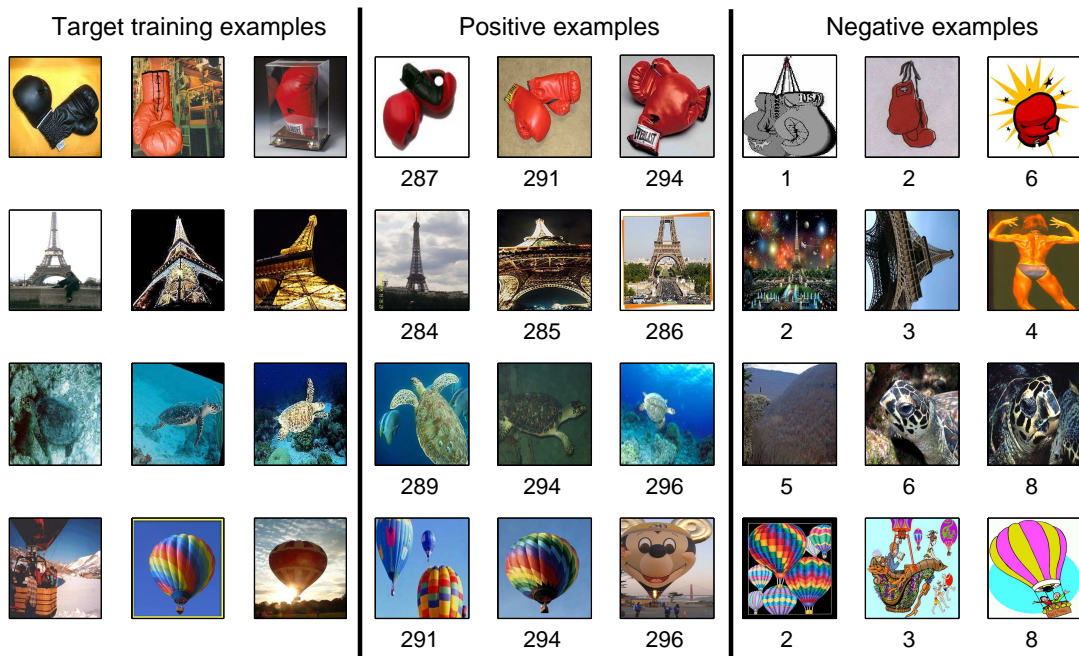


Figure 2.7: Refining image search ranking with Multi-class PMC.

compared to 16.7%) over the previously best published result on the set with 5 labeled training examples per class using these weakly-labeled data.

In this experiment, we apply PMC to the same dataset from [11] as a showcase for learning from weak supervision. We use images from Caltech-256 as labeled data L sampled from the testing distribution \mathcal{D} , and images retrieved from Bing as the “unlabeled” (weakly-labeled) data U sampled from a different distribution \mathcal{D}' . Different from classical semi-supervised learning settings, we are not fully blind about the labels of the unlabeled data in this case. Therefore in each PMC iteration, only the images obtained with the matching search query are used as the “unlabeled” set for each class to search for confident predictions (algorithm 2.1 line 6).

We argue that PMC is particularly well suited for this task for two reasons: 1) The “rote-learning” procedure of co-training adds confident instances iteratively. As a result, images that possess similar characteristics as the original training images will be picked as the confident instances, naturally ruling out irrelevant and low-quality images in the unlabeled set. 2) Classemes features are a natural fit for PMC as they consist of the predictions of

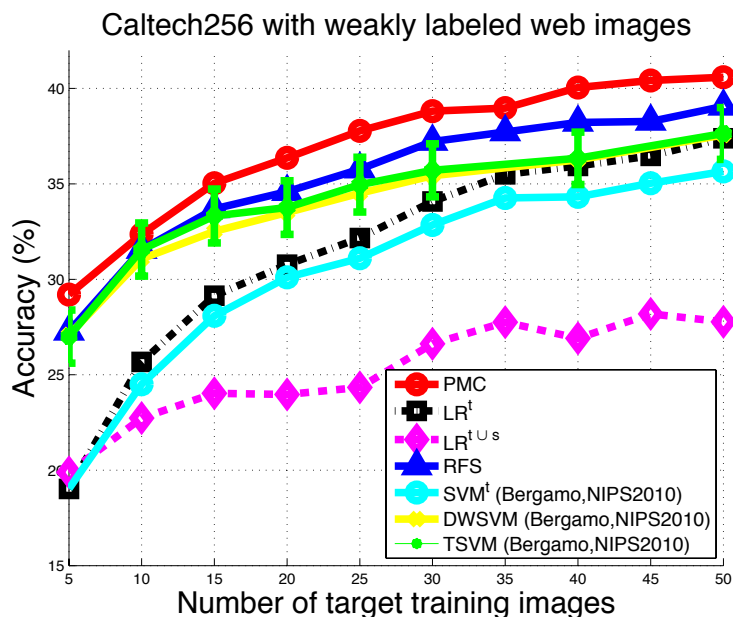


Figure 2.8: Comparison of PMC and existing works on recognition accuracy obtained with 300 web images and a varying number of Caltech-256 training examples.

many (2625) different visual concepts. It is highly likely that there exists two mutually exclusive subsets of visual concepts that satisfy the conditions for co-training.

Figure 2.7 shows example images of the Caltech-256 training set (left three columns), positive examples that PMC picks out from the “unlabeled” set to use as additional labeled images (middle three columns) and negative examples which PMC chooses to ignore (right three columns). The number below the images indicates its rank of the BingTM search engine (out of 300). For this figure, we selected the highest ranked negative and lowest ranked positive images. The results are obtained using 5 training images from Caltech-256, and 300 weakly-labeled web images for each class. The figure showcases how PMC effectively identifies relevant images that are similar in style to the training set for rote-learning. Also, it showcases that PMC can potentially be used for image re-ranking of search engines, which is particularly visible in the middle row where it ignores completely irrelevant images to the category “Eiffel Tower”, which are ranked second to fourth on BingTM.

Baselines. Figure 2.8 provides a quantitative analysis of the performance of PMC. The graph shows the accuracy achieved by different algorithms under a varying number of training

examples from Caltech-256 and 300 weakly-labeled BingTM image-search results. The meta-parameters of all algorithms were set by 5-fold cross-validation on the small labeled set (except for the group-lasso trade-off λ of PMC, which was set to $\lambda = .1$).

We train our algorithm with the multi-class loss and compare it against three baselines and three previously published results in the literature. The three baselines are: 1) multi-class logistic regression trained only on the original labeled training examples from Caltech-256 (LR^t); 2) the same model trained with both the original training images and web images ($LR^{t\cup s}$); 3) co-training with random feature splits on the labeled and weakly-labeled data (RFS).

The three previously published algorithms are: 1) linear support vector machines trained on the labeled Caltech-256 images (SVM^t) only; 2) the algorithm proposed in [11], which weighs the loss over the weakly labeled data less than over the original data ($DWSVM$); 3) transductive-SVM as introduced in [88] ($TSVM$). All previously published results are taken from [11]. All algorithms, including PMC, are linear and make no particular assumptions on the data.

General Trends. As a first observation, $LR^{t\cup s}$ performs drastically worse than the baseline trained on the Caltech-256 data LR^t only. This indicates that the weakly-labeled images are noisy enough to be harmful when they are not filtered or down-weighted. However, if the weakly labeled images are incorporated with specialized algorithms, the performance improves as can be seen by the clear gap between the purely supervised (SVM^t and LR^t) and the adaptive semi-supervised algorithms. The result of co-training with random splitting (RSF) is surprisingly good, which could potentially be attributed to the highly diverse classemes features. Finally PMC outperforms all other algorithms by a visible margin across all training set sizes. PMC achieved an accuracy of 29.2% when only 5 training images per class from Caltech-256 are used, comparing to 27.1% as reported in [11]. In terms of computational time, for a total of around 80,000 labeled and unlabeled images, PMC took around 12 hours to finish the entire training phase (Testing time is in the order of milliseconds).

2.1.7 Conclusion

In this chapter, we have introduced Pseudo Multi-view Co-training (PMC), a new framework to take advantage of weakly labeled data readily available on the web (*e.g.*, through web search) to aid learning. PMC extends co-training to single-view data by automatically decomposes the feature space and creates a pseudo-multi-view representation explicitly designed for co-training to succeed. It involves a single optimization problem, which simultaneously divides the feature space, trains two classifiers, and enforces an approximation of Balcan et al. [5]’s ϵ -expanding property. We further extend PMC to multi-class settings.

We demonstrate empirically that PMC is able to automatically discover the class-conditionally independent views of multi-view data when they exist, and it is able to create pseudo-multi-views from single view data to enable successful co-training. Although random feature splitting could serve as a good alternative for datasets with a diverse and redundant set of features, it is very unreliable otherwise, resulting in drastically degenerated performance. We showcase PMC’s efficacy on the challenging Caltech256 object recognition task, using web retrieved images as weakly labeled data. It achieves record performance on this dataset using simple features and linear classifiers⁶.

The ability of PMC to effectively select high quality instances from large collections of weakly labeled search results opens the door to future work on diverse sets of web-specific applications across very diverse domains – including web-spam classification, sentiment analysis or information retrieval.

2.2 Co-training for Domain Adaptation

In this section, we present Co-training for Domain Adaptation (CODA). CODA bridges the gap between source and target domains by slowly adding to the training set both the target features and instances in which the current algorithm is the most confident. CODA is a variant of the PMC algorithm we introduced in section 2.1. While PMC is developed in the context of semi-supervised or weakly-supervised learning, we show that with a small

⁶The performance is comparable to works which employ much more sophisticated features and classifiers, such as convolutional neural networks [146] and multiple kernel combiners [65].

modification, it works surprisingly well for domain adaptation as well. In each iteration of CODA, we formulate a single optimization problem which simultaneously learns a target predictor, a split of the feature space into views, and a subset of source and target features to include in the predictor. CODA significantly out-performs the state-of-the-art on the 12-domain benchmark data set [19].

The section is organized as follows. Section 2.2.2 recaps the setup in domain adaptation. Section 2.2.3 briefly reviews several existing domain adaptation methods that are closely related to ours. We start with a semi-supervised approach and describe a rote-learning procedure used to automatically annotate target domain inputs in section 2.2.4. The algorithm maintains and grows a training set that is iteratively adapted to the target domain. Section 2.2.5 adds an additional feature selection component to address the different features used in source and target domain. Finally, we replace the rote-learning procedure with the PMC algorithm to utilize the unlabeled data more efficiently in section 2.2.6. Experimental results comparing CODA to the state-of-the-art domain adaptation algorithms are summarized in section 2.2.7.

2.2.1 Introduction

We focus primarily on domain adaptation problems that are characterized by missing features. This is often the case in natural language processing, where different genres often use very different vocabulary to describe similar concepts. For example, in our experiments we use the sentiment data of Blitzer et al. [19], where “*a breeze to use*” is a way to express positive sentiment for kitchen appliances, but not for books. In this situation, most domain adaptation algorithms seek to eliminate the difference between source and target distributions, either by re-weighting source instances [82, 105] or by learning a new feature representation [21, 167].

We present an algorithm that differs from both of these approaches. Our method seeks to slowly adapt its training set from the source to the target domain, using ideas from co-training. We accomplish this in two ways: First, we train on our own output in rounds, where in each round, we expand our training data with the target instances we are most confident of in our prediction. Second, we select a subset of shared source and target features based

on their compatibility. Different from most previous work on selecting features for domain adaptation, the compatibility is measured across the training set and the unlabeled set, instead of across the two domains. As more target instances are added to the training set, target specific features become compatible across the two sets, therefore are included in the predictor. Finally, we exploit the PMC algorithm [42] introduced in previous chapter to exploit the unlabeled data more efficiently. These three intuitive ideas can be combined in a single optimization problem. We name our algorithm Co-Training for Domain Adaptation (CODA).

By allowing us to slowly change our training data from source to target, CODA has an advantage over instance weighing approaches [82, 105], which can not handle missing features, as well as representation-learning algorithms [21, 167], since they must decide a priori what the best representation is. In contrast, each iteration of CODA can choose exactly those few target features which can be related to the current (source and pseudo-labeled target) training set. We find that in the sentiment prediction data set of Blitzer et al. [19] CODA improves the state-of-the-art cross widely varying amounts of target labeled data in 65 out of 84 settings.

2.2.2 Recap of Domain Adaptation

Domain adaptation addresses the problem of generalizing from a *source* distribution for which we have ample labeled training data to a *target* distribution for which we have only a few or no labels [82, 167, 10]. It is of practical importance in many areas of applied machine learning, ranging from computational biology [123] to natural language processing [51, 106] to computer vision [132]. The major obstacle for porting a learner trained on a *source* domain to a *target* one is the difference between the distributions of the two domains.

Figure 2.9 shows a Venn diagram that illustrates the setup we have in domain adaptation. Let $\mathcal{X} \in \mathcal{R}^d$ denote the feature space, and \mathcal{Y} the output space. Let \mathcal{D}_T denote the *target* distribution on $\mathcal{X} \times \mathcal{Y}$. Let \mathcal{D}_S denote the distribution of the *source* domain. If \mathcal{D}_S and \mathcal{D}_T can be any arbitrary distributions, then we cannot expect to learn model that generalizes to the *target* distribution using the source data only. One way to constrain these two distributions is to assume that the supports of the two distribution \mathcal{D}_S and \mathcal{D}_T intersect, as shown in

figure 2.9. In other words, there are examples that can be generated from both distributions, *i.e.*, the ones that are invariant across domains. But domain specific examples also exist. Most domain adaptation algorithms use the invariance to bridge these two distributions. In general, the smaller the intersection is, the more difficult it is to generalize across domains.

Ben-David et al. [9] presented a theoretical analysis on generalizing across domains by constraining that there exists some classifier $h^* \in \mathcal{H}$ which has low error on both \mathcal{D}_S and \mathcal{D}_T , *i.e.*, $\kappa = \min_{h \in \mathcal{H}} \epsilon_{\mathcal{D}_S}(h) + \epsilon_{\mathcal{D}_T}(h)$ is small. Then a bound on the target error of the following form [20]

$$\epsilon_{\mathcal{D}_T}(h) \leq \epsilon_{\mathcal{D}_S}(h) + \kappa + \text{div}(\mathcal{D}_S, \mathcal{D}_T)$$

can be obtained. Here, $\text{div}(\mathcal{D}_S, \mathcal{D}_T)$ denotes the divergence between the *source* and *target* marginal distributions. In other words, the proximity of the two distributions dictates the difficulty to adapt from the source distribution to the target.

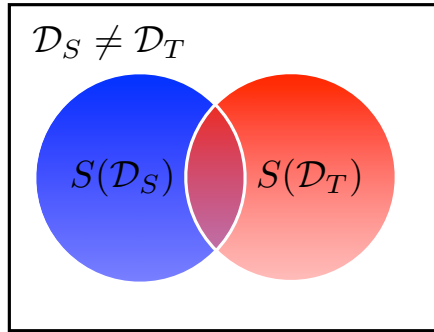


Figure 2.9: Venn diagram on domain adaptation.

Notation. In this chapter, we focus on the scenario where we have access to a sufficiently large amount of labeled data $L_S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{n_S}, y_{n_S})\} \subseteq \mathcal{X} \times \mathcal{Y}$ sampled i.i.d. from the *source* distribution \mathcal{D}_S , as well as a small (or empty) set of labeled examples $L_T = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{n_T}, y_{n_T})\} \subseteq \mathcal{X} \times \mathcal{Y}$ sampled i.i.d. from the *target* distribution \mathcal{D}_T , and another set of unlabeled data $U_T = \{\mathbf{x}_{n_T+1}, \dots, \mathbf{x}_{m_T}\}, m_T \gg n_T$ drawn i.i.d. from the same target distribution. We want to find a hypothesis $h \in \mathcal{H}$ that achieves good generalization performance on the *target* distribution using L_S, U_T and L_T if not empty.

2.2.3 Background and Related Work

The major obstacle to generalize from a source domain to a target one is the difference between the distributions of the two domains. Most domain adaptation algorithms seek to eliminate the difference, either by re-weighting source instances [82, 105] or learning a new feature representation [167, 21].

Instance Weighting

Instance weighting aims at eliminating the difference between the two distributions by assigning instance-dependent weights to the loss function when minimizing the expected loss over the training data, that is

$$\min_{h \in \mathcal{H}} \sum_i \alpha_i \ell(h(\mathbf{x}_i), y_i)$$

where α_i 's are the instance-dependent weights. As derived in [86], weighting each instance (\mathbf{x}_i, y_i) , randomly sampled from the source distribution, by $\frac{P_T(\mathbf{x}_i, y_i)}{P_S(\mathbf{x}_i, y_i)}$ provides a well-justified solution to the problem. However, it is not possible to compute the exact value of $\frac{P_T(\mathbf{x}_i, y_i)}{P_S(\mathbf{x}_i, y_i)}$ for each instance, as for domain adaptation there are not enough labeled instances in the target domain to evaluate the target distribution. In most previous works, it is either assumed that the conditional distribution of X is the same across the two domains, i.e., $P_S(X|y) = P_T(X|y)$ [99, 93]; or given the same observation \mathbf{x} , the class distribution is the same across the two domains, i.e., $P_S(y|\mathbf{x}) = P_T(y|\mathbf{x})$, but the marginal distributions are different, i.e., $P_S(\mathbf{x}) \neq P_T(\mathbf{x})$. The latter case is closely related to the problem of covariate shift [141, 147, 170, 15, 83], also known as sample selecting bias. In this case, α_i should approximate the posterior distribution of a source example being drawn from the target distribution. The posterior distribution is often estimated using kernel density estimators. However, instance weighting strategy is mostly used for low-dimensional, dense and continuous features. It is less effective for high-dimensional, sparse features spaces such as those for texts. It is also known to perform poorly in situations where target-specific features are important for good performance [87].

Change of Feature Representation

The second family of algorithms learns a new shared representation Z for the source and target domain, in which $P_S(Z, y) = P_T(Z, y)$. These algorithms are designed for highly divergent domains, which can contain completely different features, and are more closely related to our work. Ben-David et al. [9] formally analyzed the effect of representation change for domain adaptation. They proved a generalization bound for domain adaptation that is dependent on the distance between the induced $P_S(Z, y)$ and $P_T(Z, y)$.

Structural Correspondence Learning. Blitzer et al. [21] proposed Structural Correspondence Learning (SCL) to find a low-rank representation that is suitable for domain adaptation using unlabeled data from both domains. The main idea is to use pivot features, which appear frequently in both domain and behave similarly, to put domain specific words in correspondence. The original feature space is augmented by the new representation which encodes the correspondence of domain-specific features to the pivot features, and a model is then trained on the augmented feature space.

Note that, SCL operates in an unsupervised setting, without using the class labels of training inputs to derive a more informative transformation. Nevertheless, this property also guarantees that SCL can be broadly applied as a preprocessing technique for domain adaptation, regardless of the number of labeled instances available for the target domain. It has been empirically shown [21] that the low-rank representation found by SCL indeed decreases the distance between the distributions of the two domains.

Coupled Subspace. Blitzer et al. [22] later proposed another algorithm that follows the same line of thinking, named Coupled Subspace. Coupled subspace targets at utilizing target-specific features, which are important for natural language processing tasks, more effectively. While SCL finds a single projection that maps domain-specific features to the shared ones and learns a hypothesis on the shared subspace only, coupled subspace learns two projections, one for each domain, to couple shared features with the domain-specific ones. Two hypotheses are learned, one on the source specific subspace and one on the target. They are *coupled* to form the target predictor.

At a high level, SCL works better in the cases where the shared space is large. Coupled subspace on the other hand performs better when the shared space is small, then the model has the advantage of modeling the domain-specific subspaces more accurately.

Feature subsetting. Selecting a subset of features shared by the two domains is another option. Satpal and Sarawagi [134] proposed a feature selection method for domain adaptation, where the criterion is to select features whose expected values differs the least across these two distributions. A model is then learned to minimize the loss on the available training data, with an additional weighted L_1 regularization for feature selection.

We used a similar rationale in CODA. But different from this work, where the feature selection is done only once to select features shared across domain, we perform the feature selection multiple times, promoting *target* domain specific features to be gradually added to the learner for better target performance.

Feature Augmentation

Daume III [51] proposed EasyAdapt (EA) to learn a *general*, a *source* specific and a *target* specific hypothesis simultaneously through feature augmentation. They define an augmented input space, in which each feature from the original space is made three version of, a general version, a source specific version, and a target specific version. The source and the target data are mapped to the augmented space respectively by

$$\Phi_S(\mathbf{x}) = \langle \mathbf{x}, \mathbf{x}, \mathbf{0} \rangle, \quad \Phi_T(\mathbf{x}) = \langle \mathbf{x}, \mathbf{0}, \mathbf{x} \rangle.$$

Finally, a linear hypothesis $\mathbf{h} = \langle g_C, g_S, g_T \rangle \in \mathcal{R}^{3d}$ is trained in the augmented space with the labeled instances from both domains. One particularly nice property of EA lies in its simplicity. It is incredibly easy to implement as a preprocessing step and it is agnostic to the classifiers used. However, EA requires labeled data in both source and target, and hence applies to fully supervised domain adaptation settings only. To overcome the limitation, Daume III et al. [50] generalize this work to a semi-supervised setting so that they can leverage unlabeled instances from the target domain by simply adding a new feature mapping for unlabeled instances.

$$\Phi_U(\mathbf{x}) = \langle \mathbf{0}, \mathbf{x}, -\mathbf{x} \rangle$$

The extension is called EasyAdapt++ (EA++).

2.2.4 Self-training

The first component of our algorithm, a rote learning procedure, gradually shifts the distribution of the training set from source to target by “labeling” the target inputs in U_T and adding the confident predictions to the training set.

For now, we focus on binary problems. First, we assume the existence of a base classifier that provides some estimate of confidence in its prediction. Throughout this section we simply use logistic regression. That is, our classifier $h_{\mathbf{w}}(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$ is parameterized by a weight-vector $\mathbf{w} \in \mathcal{R}^d$ and a bias b . To simplify notation we drop the bias b and assume that a constant 1 is attached as an additional dimension to each input. The weight vector $\mathbf{w} \in \mathcal{R}^{d+1}$ is then set to minimize the logistic loss function

$$\ell(\mathbf{w}; L) = \frac{1}{|L|} \sum_{(\mathbf{x}, y) \in L} \log \left(1 + e^{-y\mathbf{w}^\top \mathbf{x}} \right), \quad (2.14)$$

where L is the training set. Let $\hat{y} = \text{sign}(\mathbf{w}^\top \mathbf{x})$ denote the prediction of $h_{\mathbf{w}}$ on an input \mathbf{x} , the probability $P_h(\hat{y}|\mathbf{x}; \mathbf{w}) = \left(1 + e^{-\mathbf{w}^\top \mathbf{x} \hat{y}} \right)^{-1}$ is a natural metric of certainty. But other methods [121] can be used as well.

Self-training [106] is a simple and intuitive iterative algorithm to leverage unlabeled data. During training one maintains a labeled training set L and an unlabeled testing set U , initialized as $L = L_S \cup L_T$ and $U = U_T$. In each iteration, a classifier $h_{\mathbf{w}}$ is trained to minimize the loss function ℓ over L and is evaluated on all elements of U . The l most confident predictions on U are *moved* to L for the next iteration, labeled as $h_{\mathbf{w}}(\mathbf{x}_i) = \text{sign}(\mathbf{w}^\top \mathbf{x}_i)$. The algorithm terminates when U is empty or all predictions are below a pre-defined confidence threshold (and considered unreliable). Algorithm 2.2 summarizes self-training in pseudocode.

Algorithm 2.2 Self-training.

- 1: Inputs: labeled set $L = L_S \cup L_T$, unlabeled set $U = U_T$.
 - 2: Initialize \mathbf{w} and l .
 - 3: **repeat**
 - 4: $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \ell(\mathbf{w}; L)$
 - 5: Apply $h_{\mathbf{w}^*}$ on all elements of U .
 - 6: Move up-to l confident inputs \mathbf{x}_i from U to L , labeled as $h_{\mathbf{w}^*}(\mathbf{x}_i)$.
 - 7: **until** No more predictions are confident
 - 8: Return $h_{\mathbf{w}^*}$
-

2.2.5 Feature Selection

So far, we have not addressed the missing feature problem, *e.g.*, different vocabularies are used to express similar concept in different domains. For illustration, consider the sentiment analysis problem in section 2.2.7, where each input consists of unigram and bigram bag-of-words features and the task is to classify if a book-review (source domain) or dvd-review (target domain) is positive or negative. Here, the bigram feature “must read” is indicative of a positive opinion within the source (“books”) domain, but rarely appears in the target (“dvd”) domain. A classifier, trained on the source-dominated set L , that relies too heavily on such features will not make enough high-confidence predictions on the set $U = U_T$.

The second component of our algorithm, a feature selection component, addresses the problem by adding a weighted l_1 regularization to encourage the classifier to only use features that behave similar in both L and U .

Different from previous work on feature selection for domain adaptation [134], where the goal is to find a new representation to minimize the difference between the distributions of the source and target domain, what we are proposing is to minimize the difference between the distributions of the labeled training set L and the unlabeled set U (which coincides with the testing set in our setting). This difference is crucial, as it makes the empirical distributions of L and U align *gradually*. For example, after some iterations, the classifier can pick features that are *never* present in the source domain, but which have entered L through the rote-learning procedure.

We perform the feature selection implicitly through regularization on \mathbf{w} . For a feature α , let us denote the Pearson correlation coefficient (PCC)⁷ between feature value \mathbf{x}_α and the label y for all pairs $(\mathbf{x}, y) \in L$ as $\rho_L(\mathbf{x}_\alpha, y)$. It can be shown that $\rho_L(\mathbf{x}_\alpha, y) \in [-1, 1]$ with a value of +1 if a feature is perfectly aligned with the label (*i.e.*, the feature *is* the label), 0 if it has no correlation, and -1 if it is of opposite polarity (*i.e.*, the *inverted* label). Similarly, let us define the PCC for all pairs in U as $\rho_{U;\mathbf{w}}(\mathbf{x}_\alpha, Y)$, where the unknown label Y is a random variable drawn from the conditional probability $P_h(Y|\mathbf{x}; \mathbf{w})$. The two PCC values indicate how predictive a feature is of the (estimated) class label in the two respective data sets. Ideally, we would like to choose features that are similarly predictive across the two sets. We measure how similarly a feature behaves across L and U with the product $\rho_L(\mathbf{x}_\alpha, y)\rho_{U;\mathbf{w}}(\mathbf{x}_\alpha, Y)$. With this notation, we define the feature weight that reflects the cross-domain *incompatibility* of a feature as

$$\Delta_{\mathbf{w};L,U}(\alpha) = (1 - \rho_L(\mathbf{x}_\alpha, y)\rho_{U;\mathbf{w}}(\mathbf{x}_\alpha, Y)). \quad (2.15)$$

It is straight-forward to show that $\Delta_{\mathbf{w};L,U} \in [0, 2]$. Intuitively, $\Delta_{\mathbf{w};L,U}$ expresses to what degree we would like to *remove* a feature. A perfect feature, that is the label itself (and the prediction in U), results in a score of 0. A feature that is not correlated with the class label in at least one of the two domains (and therefore is too domain-specific) obtains a score of 1. A feature that switches polarization across domains (and therefore is “malicious”) has a score $\Delta_{\mathbf{w};L,U}(\alpha) > 1$ (in the extreme case if it is the label in L and the *inverted* label in U , its score would be 2).

We incorporate (2.15) into a weighted ℓ_1 regularization

$$s(\mathbf{w}; L, U) = \sum_{\alpha=1}^d \Delta_{\mathbf{w};L,U}(\alpha) |\mathbf{w}_\alpha|. \quad (2.16)$$

Intuitively (2.16) encourages feature sparsity with a strong emphasis on features with little or opposite correlation across the domains, whereas good features that are consistently predictive in both domains become *cheap*.

⁷The PCC for two random variables X, Y is defined as $\rho = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$, where μ_X denotes the mean and σ_X the standard deviation of X .

Algorithm 2.3 Self-training for Domain Adaptation (SEDA).

- 1: Inputs: labeled set $L = L_S \cup L_T$, unlabeled set $U = U_T$.
 - 2: Initialize $\gamma = \gamma_0$, $0 < \eta < 1$ and l .
 - 3: **repeat**
 - 4: $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \ell(\mathbf{w}; L) + \gamma s(\mathbf{w}; L, U)$
 - 5: $\gamma = \frac{\gamma}{1+\eta}$.
 - 6: Apply $h_{\mathbf{w}^*}$ on all elements of U .
 - 7: Move up-to l confident inputs \mathbf{x}_i from U to L , labeled as $h_{\mathbf{w}^*}(\mathbf{x}_i)$.
 - 8: **until** No more predictions are confident
 - 9: Return $h_{\mathbf{w}^*}$
-

Combining the feature selection component with the self-training component in previous section, we have an algorithm which shifts the distribution of the training data from *source* to *target* both data-wise and feature-wise. We refer to this version of the algorithm as Self-training for Domain Adaptation (SEDA). Algorithm 2.3 describes SEDA in pseudo-code. it is very similar to the Self-training algorithm we described in algorithm 2.2, except a different loss function is solved for the weight vector. SEDA finds a weight vector \mathbf{w} that minimizes a weighted L_1 regularized logistic loss as follows:

$$\mathbf{w} = \operatorname{argmin}_{\mathbf{w}} \ell(\mathbf{w}; L) + \gamma s(\mathbf{w}; L, U). \quad (2.17)$$

Here, $\gamma \geq 0$ denotes the loss-regularization trade-off parameter. As we have very few labeled inputs from the *target* domain in the early iterations, stronger regularization is imposed so that only features shared across the two domains are used. When more and more inputs from the *target* domain are included in the training set, we gradually decrease the regularization to accommodate *target* specific features. The algorithm is very insensitive to the exact initial choice of γ . The guideline is to start with a relatively large number, and decrease it until the selected feature set is not empty. In our implementation, we start with $\gamma_0 = 0.1$, and we divide it by a factor of 1.1 after each iteration.

2.2.6 Co-training for Domain Adaptation

For rote-learning to be effective, we need to move inputs from U to L that are 1) correctly classified (with high probability) and 2) useful to improve the classifier in future iterations.

The former is addressed by the feature selecting regularization from the previous section – restricting the classifier to a subset of features that are known to be cross-data set compatible reduces the generalization error on U . In this section we fulfill the second requirement. In other words, we want to add inputs \mathbf{x}_i that contain additional information, which was not used to obtain the prediction $h_{\mathbf{w}}(\mathbf{x}_i)$ and would enrich the training set L .

If the exact labels of the inputs in U were known, a good active learning [137] strategy would be to move inputs to L on which the current classifier $h_{\mathbf{w}}$ is *most uncertain*. In our setting, this would be clearly ill advised as the uncertain prediction is going to be used as the label in the next iteration. A natural solution to this dilemma is the co-training [23] algorithm we introduced in previous chapter. Co-training assumes the data set is presented in two separate views and two classifiers are trained, one on each view. In each iteration, only inputs that are confident according to *exactly one* of the two classifiers are moved to the training set. This way, one classifier provides the (estimated) labels to the inputs which the *other* classifier is uncertain about.

Same here we *do not* have multiple views. Moreover, the features selected vary in each iteration. Hence, co-training does not apply out-of-the-box. We can, however, *split* the selected features into two mutually exclusive views in each iteration, such that co-training is effective. To this end we follow the Pseudo Multiview Co-training (PMC) [42] introduced in section 2.1.

Recall that we want to train *two* classifiers on a single view \mathcal{X} such that: 1) both perform well on the labeled data; 2) both are trained on strictly different features; 3) together they are likely to satisfy Balcan et al. [5]’s condition of ϵ -expandability [5], a necessary and sufficient pre-condition for co-training to work⁸. These three aspects can be formulated explicitly as three modifications of our optimization problem (2.17).

Loss. Two classifiers are required for co-training, whose weight vectors we denote by \mathbf{u} and \mathbf{v} . The performance of each classifier is measured by the log-loss $\ell(\cdot; L)$ in eq. (2.14). To ensure that both classifiers perform well on the training set L , *i.e.*, both having a small

⁸Provided that the classifiers are never confident but wrong — which can be violated in practice.

training loss, we train them jointly while minimizing the soft-maximum⁹ of the two losses,

$$\log(e^{\ell(\mathbf{u};L)} + e^{\ell(\mathbf{v};L)}). \quad (2.18)$$

Feature Decomposition. Co-training requires the two classifiers to be trained on different views. We create them for single view data by splitting the feature-space into two mutually exclusive subsets. More precisely, for each feature α , at least one of the two classifiers must have a zero weight in the α^{th} dimension. We can enforce this across all features with the equality constraint

$$\sum_{\alpha=1}^d \mathbf{u}_{\alpha}^2 \mathbf{v}_{\alpha}^2 = 0. \quad (2.19)$$

ϵ -Expandability. In the original co-training formulation [23], it is assumed that the two views of the data are class conditionally independent. This assumption is very strong and can easily be violated in practice [116]. Recent work [5] weakens this requirement significantly to a condition of ϵ -*expandability*. Loosely phrased, for the two classifiers to be able to teach each other, they must make confident predictions on different subsets of the unlabeled set U .

For the classifier $h_{\mathbf{u}}$, let $\hat{y} = \text{sign}(\mathbf{u}^{\top} \mathbf{x}) \in \{\pm 1\}$ denote the class prediction and $P_h(\hat{y}|\mathbf{x}; \mathbf{u})$ its confidence. Define $c_{\mathbf{u}}(\mathbf{x})$ as a confidence indicator function (for some confidence threshold $\tau > 0$)¹⁰

$$c_{\mathbf{u}}(\mathbf{x}) = \begin{cases} 1 & \text{if } p(\hat{y}|\mathbf{x}; \mathbf{u}) > \tau \\ 0 & \text{otherwise,} \end{cases} \quad (2.20)$$

and $c_{\mathbf{v}}$ respectively. Then the ϵ -expanding condition translates to

$$\sum_{\mathbf{x} \in U} [c_{\mathbf{u}}(\mathbf{x}) \bar{c}_{\mathbf{v}}(\mathbf{x}) + \bar{c}_{\mathbf{u}}(\mathbf{x}) c_{\mathbf{v}}(\mathbf{x})] \geq \epsilon \min \left[\sum_{\mathbf{x} \in U} c_{\mathbf{u}}(\mathbf{x}) c_{\mathbf{v}}(\mathbf{x}), \sum_{\mathbf{x} \in U} \bar{c}_{\mathbf{u}}(\mathbf{x}) \bar{c}_{\mathbf{v}}(\mathbf{x}) \right], \quad (2.21)$$

for some $\epsilon > 0$. Here, $\bar{c}_{\mathbf{u}}(\mathbf{x}) = 1 - c_{\mathbf{u}}(\mathbf{x})$ indicates that classifier $h_{\mathbf{u}}$ is *not* confident about input \mathbf{x} . Intuitively, the constraint in eq. (2.21) ensures that the total number of inputs in U

⁹The soft-max of a set of elements S is a differentiable approximation of $\max(S) \approx \log(\sum_{s \in S} e^s)$.

¹⁰In our implementation, the 0-1 indicator was replaced by a very steep differentiable sigmoid function, and τ was set to 0.8 across different experiments.

that can be used for rote-learning because *exactly one* classifier is confident (LHS), is larger than the set of inputs which *cannot* be used because *both* classifiers are already confident or *both are not* confident (RHS).

In summary, the framework splits the feature space into two mutually exclusive subsets, on which two logistic regression classifiers can be trained, both with small loss on the labeled data set, while satisfying the ϵ -expandability [5]. Our final classifier has the weight vector $\mathbf{w} = \mathbf{u} + \mathbf{v}$. To address the missing feature problem, we add the weighted L_1 regularization $s(\mathbf{w}; L, U)$ as defined in (2.16) to the optimization. We refer to the resulting algorithm as Co-training for Domain Adaptation (CODA), which can be stated concisely with the following optimization problem:

$$\begin{aligned}
 & \min_{\mathbf{w}, \mathbf{u}, \mathbf{v}} \log(e^{\ell(\mathbf{u}; L)} + e^{\ell(\mathbf{v}; L)}) + \gamma s(\mathbf{w}; L, U) \\
 & \text{subject to:} \\
 & \quad (1) \sum_{i=1}^d \mathbf{u}_i^2 \mathbf{v}_i^2 = 0 \\
 & \quad (2) \sum_{\mathbf{x} \in U} [c_{\mathbf{u}}(\mathbf{x}) \bar{c}_{\mathbf{v}}(\mathbf{x}) + \bar{c}_{\mathbf{u}}(\mathbf{x}) c_{\mathbf{v}}(\mathbf{x})] \geq \epsilon \min [\sum_{\mathbf{x} \in U} c_{\mathbf{u}}(\mathbf{x}) c_{\mathbf{v}}(\mathbf{x}), \sum_{\mathbf{x} \in U} \bar{c}_{\mathbf{u}}(\mathbf{x}) \bar{c}_{\mathbf{v}}(\mathbf{x})] \\
 & \quad (3) \mathbf{w} = \mathbf{u} + \mathbf{v}
 \end{aligned}$$

The optimization is non-convex. However, as it is not particularly sensitive to initialization, we initialize \mathbf{u}, \mathbf{v} randomly and solve them using the augmented lagrangian method introduced in section 1.3.2. Algorithm 2.4 shows a pseudo-code of CODA. The implementation is essentially identical to that of SEDA (algorithm 2.3) where the above optimization problem is solved instead of eq. (2.17). We move inputs that one classifier is confident about while the other one is uncertain to the training set L to improve the classifiers in future iterations.

2.2.7 Experimental Results

We evaluate our algorithm together with several other domain adaptation algorithms on the “Amazon reviews” benchmark data sets [19].

Algorithm 2.4 Co-training for Domain Adaptation.

- 1: Inputs: labeled set $L = L_S \cup L_T$, unlabeled set $U = U_T$.
 - 2: Initialize \mathbf{u}, \mathbf{v} .
 - 3: Initialize $\gamma = \gamma_0$, $0 < \eta < 1$ and l .
 - 4: **repeat**
 - 5: Solve the constrained optimization problem on L, U and γ for $\mathbf{u}^*, \mathbf{v}^*$ and \mathbf{w}^* .
 - 6: $\gamma = \frac{\gamma}{1+\eta}$
 - 7: Apply $h_{\mathbf{u}^*}$ and $h_{\mathbf{v}^*}$ on all elements of U .
 - 8: Move up-to l inputs \mathbf{x}_i from U to L , on which $h_{\mathbf{u}^*}$ is confident and $h_{\mathbf{v}^*}$ is not, labeled as $h_{\mathbf{u}^*}(\mathbf{x}_i)$.
 - 9: Move up-to l inputs \mathbf{x}_i from U to L , on which $h_{\mathbf{v}^*}$ is confident and $h_{\mathbf{u}^*}$ is not, labeled as $h_{\mathbf{v}^*}(\mathbf{x}_i)$.
 - 10: **until** No more predictions are confident
 - 11: Return $h_{\mathbf{w}^*}$
-

Dataset. The data set contains reviews of four different types of products: books, dvds, electronics, and kitchen appliances from Amazon.com. In the original dataset, each review is associated with a rating of 1-5 stars. For simplicity, we are only concerned about whether a review is positive (higher than 3 stars) or negative (3 stars or lower). That is, $y_i \in \{+1, -1\}$, where $y_i = 1$ indicates that it is a positive review, and -1 otherwise. The data from four domains results in 12 directed adaptation tasks (*e.g.* *books* \rightarrow *dvds*). Each domain adaptation task consists of 2,000 labeled source inputs and around 4,000 unlabeled target test inputs (varying slightly between domains). Table 2.2 shows the statistics of each domain, including the number of labeled, unlabeled examples, as well as the portion of the negative examples. It is a balanced dataset.

Table 2.2: Statistics of the Amazon review dataset [19].

DOMAIN	LABELED	UNLABELED (TEST)	NEG. INPUTS
BOOKS	2000	4465	50%
DVDS	2000	3586	50%
ELECTRONICS	2000	5681	50%
KITCHEN	2000	5945	50%

The original feature space of unigrams and bigrams is of approximately 100,000 dimensions across different domains. To reduce the dimensionality, we only use features that appear

at least 10 times in a particular domain adaptation task (with approximately 40,000 features remaining). Further, we pre-process the data set with standard tf-idf [133] feature re-weighting.

Methods. We compare the three algorithms we introduced in 2.2.4, 2.2.5 and 2.2.6 respectively with one baseline method and three state-of-the-art domain adaptation algorithms. 1) logistic regression (Logistic Regression), a baseline model that ignores the difference between source and target distribution, and train a classifier on the union of both labeled data sets; 2) Coupled Subspace (Coupled) [22], a representation change approach that couples a source specific and a target specific subspace together to utilize domain specific features; 3) EasyAdapt [51], learns a general, a source specific and a target specific hypothesis simultaneously through feature augmentation ; 4) EasyAdapt++ [50], extends EasyAdapt with the help of unlabeled target data; Section 2.2.3 contains more detailed descriptions of these algorithms. 5) Self-training, which adds the semi-supervised learning approach to label and gradually add target inputs to the training set, as described in section 2.2.4; 6) SEDA, includes a feature selection component to address the missing feature problem, as described in section 2.2.5. We optimize over 100 iterations of self-training, at which stage the regularization was effectively zero and the classifier converged. ; 7) CODA, replaces self-training with pseudo-multi-view co-training algorithm to better utilize new predictions, as described in section 2.2.6.;

We compare the various algorithms under different level of target supervision by varying the amount of labeled *target* data from 0 to 1600. For each setting when target labels are available, we ran 10 experiments with different, randomly chosen, labeled instances.

Figure 2.10 summarizes a comparison of the three algorithms we introduced, self-training, SEDA and CODA to the logistic regression baseline. As the 12 adaptation tasks vary in difficulties, before averaging we divide the test error of the new method by the test error of the baseline. The relative test-error reduction averaged over all 12 adaptation tasks is plotted as a function of the number of target labeled data. We observe two trends: 1) there are clear gaps between logistic regression, self-training, SEDA, and CODA; One can conclude that self-training, feature-selection and co-training each lead to substantial improvements in classification error. 2) the relative improvement over logistic regression reduces as more labeled target data becomes available. This is not surprising, as with sufficient target labels

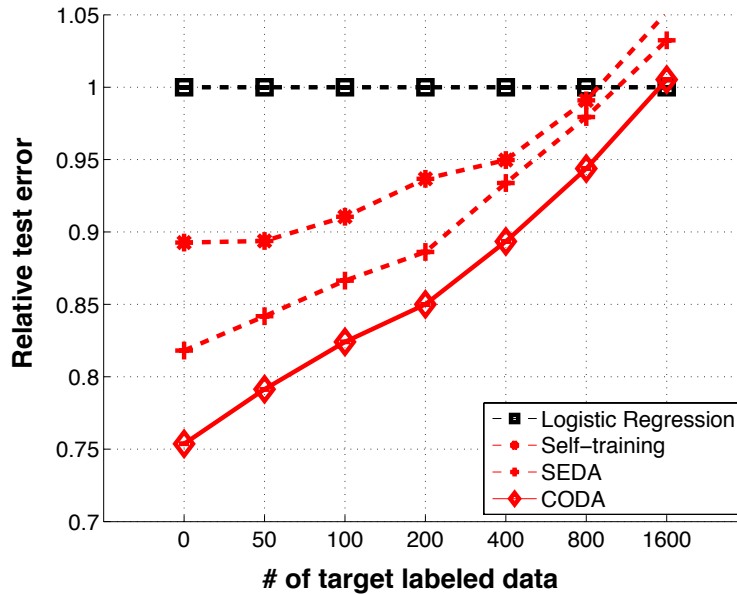


Figure 2.10: Comparison of self-training, SEDA and CODA on the relative test-error reduction over logistic regression, averaged across all 12 domain adaptation tasks.

the task turns into a classical supervised learning problem and the source data becomes irrelevant.

Figure 2.11 summarizes a comparison of CODA against three state-of-the-art domain adaptation algorithms, *i.e.*, coupled subspace (Coupled), EasyAdapt and EasyAdapt++. Again we plot the relative test error reduction over the logistic regression baseline, averaged across 12 adaptation tasks. Since coupled subspace method does not utilize labeled target data, its result is depicted as a single point. Coupled subspace improves over the logistic regression baseline, however, since it does not utilize the available target supervision, the improvement is not substantial. Both EasyAdapt and EasyAdapt++ require sufficient target supervision in order to be effective. We can see that CODA outperforms the other methods significantly across all target supervision levels. In particular, the performance is much improved when the number of target labels is small.

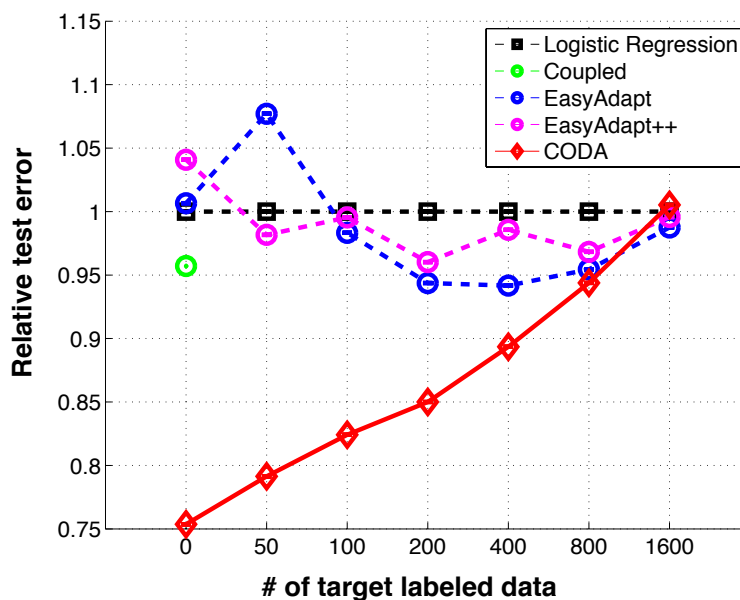


Figure 2.11: Comparison of CODA and existing work on the relative test-error reduction over logistic regression, averaged across all 12 domain adaptation tasks.

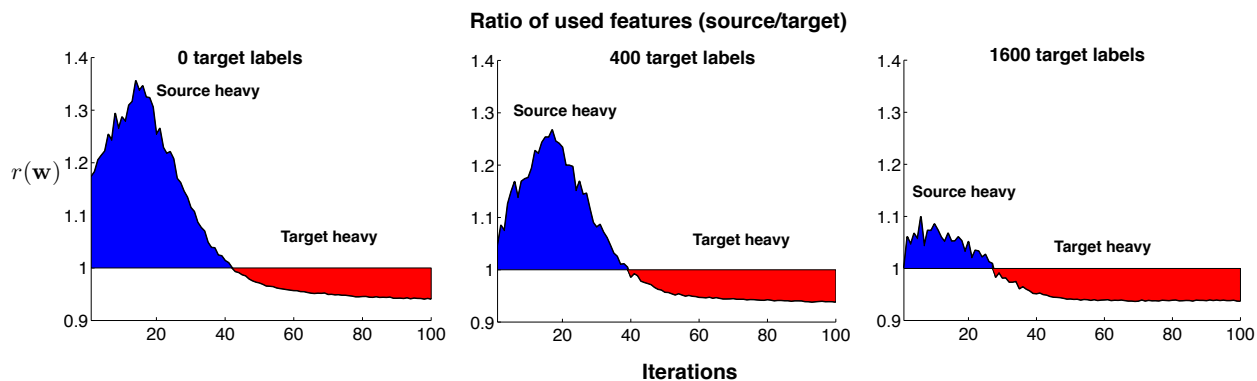


Figure 2.13: The ratio of the average number of used features between source and target inputs (2.22), tracked throughout the CODA optimization. The three plots show the same statistic at different amounts of target labels.

Figure 2.12 shows individual results on all the 12 adaptation tasks with absolute classification error rates. The error bars show the standard deviation across the 10 runs with different labeled instances. EasyAdapt and EasyAdapt++, both consistently improve over logistic regression once sufficient target data is available. It is noteworthy that, on average, CODA

outperforms the other algorithms in almost all settings when 800 labeled target points or less are present. With 1600 labeled target points all algorithms perform similar to the baseline and additional source data is irrelevant. All hyper-parameters of competing algorithms were carefully set by 5-fold cross validation.

Finally, we investigate the feature-selection process during CODA training. In each iteration of CODA, only a subset of features are used in the classifiers due to the weighted L_1 regularization. Let us define an indicator function $\delta(a) \in \{0, 1\}$ to be $\delta(a) = 0$ if and only if $a = 0$, which operates element-wise on vectors. The vector $\delta(\mathbf{w}) \in \{0, 1\}^d$ indicates which features are used in the classifier and $\delta(\mathbf{x}_i) \in \{0, 1\}^d$ indicates which features are present in input \mathbf{x}_i . For a classifier \mathbf{w} , $\delta(\mathbf{w})^\top \delta(\mathbf{x}_i)$ counts the number of *used* features that are active in input \mathbf{x}_i . We can then denote the ratio between the average number of *used* features in labeled source inputs over those in unlabeled target inputs in each CODA iteration as

$$r(\mathbf{w}) = \frac{\frac{1}{|L_S|} \sum_{\mathbf{x}_s \in L_S} \delta(\mathbf{w})^\top \delta(\mathbf{x}_s)}{\frac{1}{|U_T|} \sum_{\mathbf{x}_t \in U_T} \delta(\mathbf{w})^\top \delta(\mathbf{x}_t)}. \quad (2.22)$$

Figure 2.13 shows the plot of $r(\mathbf{w})$ for all weight vectors during the 100 iterations of CODA, averaged (geometric mean) across all 12 adaptation tasks. The three plots show the same statistic under varying amounts of target labels. Two trends can be observed: 1) during CODA training, the classifier initially selects more source-specific features. For example when there are zero labeled target data (left plot), during early iterations the source inputs contain on average 20 – 35% more *used* features compared to target inputs. However, this source-heavy feature distribution changes and eventually turns into target-heavy distribution as the classifier adapts to the target domain. 2) we observe that with more target labels (right plot), this spike in source features is much less pronounced whereas the final target-heavy ratio is unchanged but starts earlier. This indicates that as the target labels increase, the classifier makes less use of the source data and relies sooner and more directly on the target signal.

Computational Time. Concerning computational requirements, it is fair to say that CODA is significantly slower than the other algorithms, as each iteration is of comparable complexity as logistic regression or EasyAdapt. In typical domain adaptation settings this

is generally not a problem, as training sets tend to be small. In our experiments, the average training time for CODA¹¹ was about 20 minutes.

2.2.8 Conclusion

In this chapter, we have introduced Co-training for Domain Adaptation (CODA), an intuitive and effective algorithm for domain adaptation. The algorithm combines three core components: rote-learning, feature selection and pseudo-multiview feature splitting. In particular, the rote-learning gradually shifts the distribution of the training data from source to target, the new feature selection component slowly shifts its active feature distribution from source to target, and the pseudo-multiview feature splitting helps better utilize the new predictions. In our empirical evaluation we demonstrated that each one of these components contributes substantially to CODA's strong performance. CODA achieves state-of-the-art classification results with impressive consistency across a wide range of available target labels.

¹¹We used a straight-forward MatlabTM implementation.

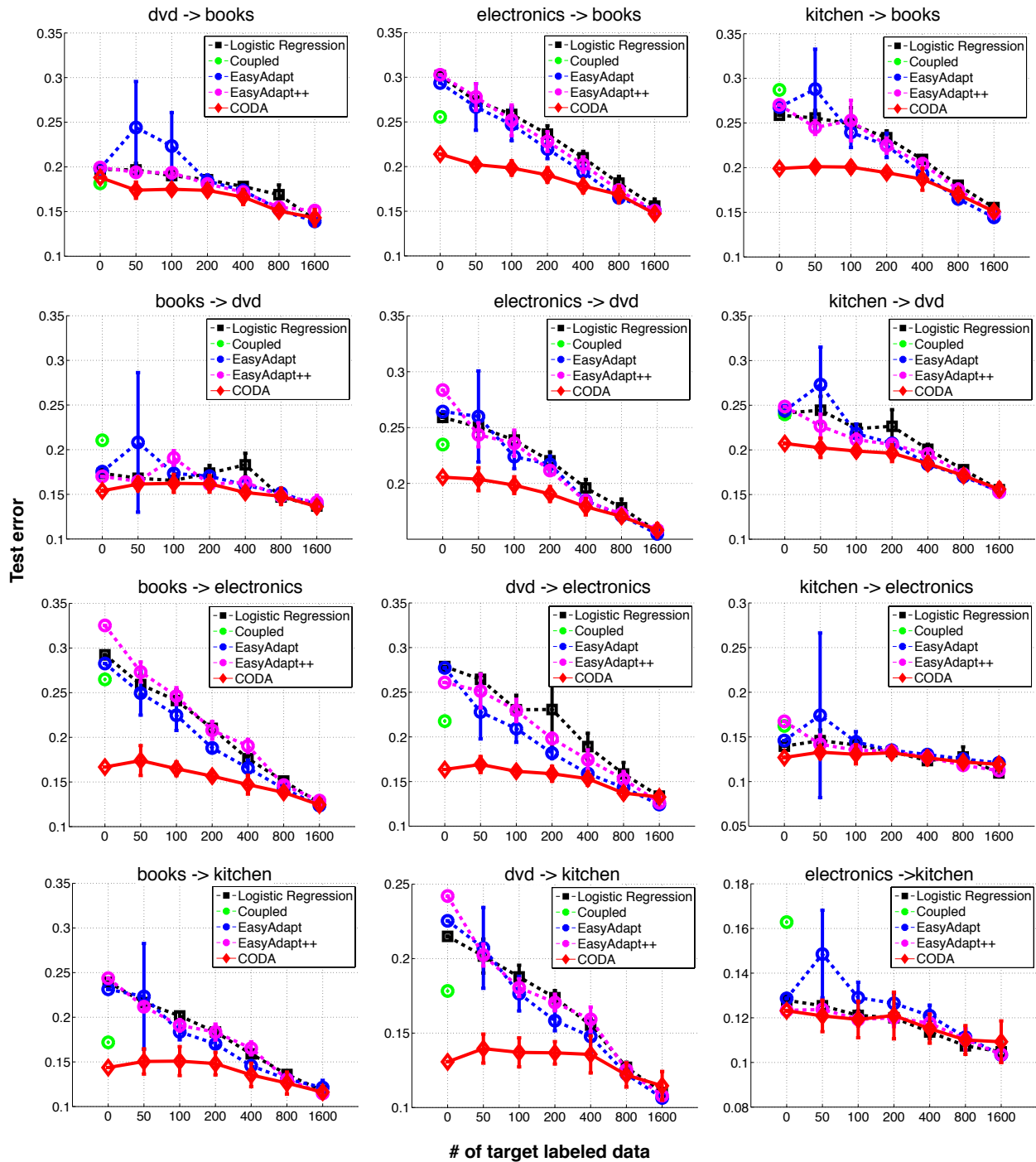


Figure 2.12: Comparison of CODA and existing work on the absolute test error under varying amounts of labeled target data on 12 adaptation tasks.

Chapter 3

Marginalized Dropout

In this chapter, we manifest our second strategy, marginalized dropout, in three different ways for learning from multiple domains, learning from corrupted distribution and learning with partial supervision. Section 3.1 details *marginalized Stacked Denoising Autoencoders* (mSDA). mSDA exploits marginalized dropout in learning a new feature representation for learning from multiple domains. By marginalizing out the corrupting distribution applied on the examples, mSDA is able to take into consideration all possible corruptions when learning the new representation. It achieves state-of-the-art performance and orders of magnitude speedup in training comparing to competing algorithms. Section 3.2 details learning with *Marginalized Corrupted Features* (MCF). Instead of learning on the small set of training examples generated from the testing distribution, MCF learns on “infinitely many” examples sampled from a corrupted distribution. By marginalizing out the corrupting distribution applied on the examples, MCF is able to train classifiers on infinitely many examples without increasing computational cost. It can be used as an alternative to regularizations or learning with prior methods to improve robustness of trained classifiers. Section 3.3 details *FastTag* (FastTag). FastTag utilizes marginalized dropout to recover missing labels for learning with partial supervision. We take the task of image annotation with incomplete user tags as an example. By marginalizing out the corrupting distribution applied on the tags, FastTag is able to recover the correlation between tags considering all possible random tag removal. FastTag gives rise to a jointly convex loss function, which can be efficiently optimized with closed form updates. It achieves the state-of-the-art tagging quality, and orders of magnitude speedup in both training and testing.

3.1 Marginalized Corrupted Feature Learning

In this section, we present marginalized Stacked Denoising Autoencoders (mSDA). mSDA is inspired by the Stacked Denoising Autoencoders (SDAs) algorithm [69] successfully applied to domain adaptation. Robust data representation is learnt by reconstruction, recovering original features from data that are artificially corrupted with noise. SDAs have attained record accuracy on standard benchmark tasks of sentiment analysis across different text domains. mSDA addresses two crucial limitations of SDAs: high computational cost and lack of scalability to high-dimensional features. In contrast to SDAs, our approach of mSDA marginalizes noise and employs a linear model as the basic building block, and thus does not require stochastic gradient descent or other optimization algorithms to learn parameters in fact, they are computed in closed-form. Consequently, mSDA, which can be implemented in only 20 lines of MATLABTM, significantly speeds up SDAs by two orders of magnitude. Furthermore, the representations learnt by mSDA are as effective as the traditional SDAs, attaining almost identical accuracies in benchmark tasks.

The section is organized as follows. Section 3.1.2 recaps the setup in learning from multiple domains. Section 3.1.3 reviews the two existing works that are closely related to mSDA, the Structural Correspondence Learning and the Stacked Denoising Autoencoders algorithm. Section 3.1.4 details our modified version of SDA, marginalized SDAs. Section 3.1.6 presents the experimental results comparing mSDA to state-of-the-art domain adaptation algorithms, in particular SDA on a domain adaptation benchmark dataset for sentiment analysis.

3.1.1 Introduction

Recall the goal in domain adaptation [10, 82, 165, 167] is to generalize a classifier that is trained on a *source* domain, for which typically plenty of training data is available, to a *target* domain, for which data is scarce. Cross-domain generalization is important in many application areas of machine learning, where such an imbalance of training data may occur. Examples are computational biology [123], natural language processing [51, 106] and computer vision [132].

However, data in the *source* and the *target* are often distributed differently. This presents a major obstacle in adapting predictive models. Recent work has investigated several techniques for alleviating the difference: instance reweighting [82, 105], sub-sampling from both domains [42] and learning joint source and target feature representations [21, 69, 167].

Recently, Glorot et al. [69] proposed a new approach that falls into the third category. The authors propose to learn robust feature representations with stacked denoising autoencoders (SDA) [159]. Denoising autoencoders are one-layer neural networks that are optimized to reconstruct input data from partial and random corruption. These denoisers can be stacked into deep learning architectures. The outputs of their intermediate layers are then used as input features for SVMs [97]. Glorot et al. [69] demonstrate in their work that using SDA-learned features in conjunction with linear SVM classifiers yields record performance on the benchmark tasks of sentiment analysis across different product domains [21].

Despite their remarkable and promising results, SDAs are limited by their high computational cost. They are significantly slower to train than competing algorithms [21, 41, 167], primarily because of their reliance on iterative and numerical optimization to learn model parameters. The challenge is further compounded by the dimensionality of the input data and the need for computationally intensive model selection procedures to tune hyperparameters. Consequently, even a highly optimized implementation [12] may require hours (even days) of training time.

We address this challenge with a variant of SDA. The proposed method, which we refer to as *marginalized Stacked Denoising Autoencoder* (mSDA), adopts the greedy layer-by-layer training of SDAs. However, a crucial difference is that we use *linear* denoisers as the basic building blocks. The key observation is that, in this setting, the random feature corruption can be marginalized out. Conceptually, this is equivalent to training the models with an infinitely large number of corrupted input data. Fitting models on such a scale would be impossible for the conventional SDAs, which often rely on stochastic gradient descent, and need to sweep through all the training data.

Our contributions are summarized as follows: 1) we contribute to deep learning by demonstrating that linear denoisers can be used as building blocks for learning feature representations. 2) we show that linearity can significantly simplify parameter estimation — our

approach results in closed-form solutions for the optimal parameters. 3) we evaluate our approach rigorously on established domain adaptation benchmark data sets and compare with several competing state-of-the-art algorithms. We show that the classification performance of mSDA matches that of SDA across our benchmark data sets, while achieving tremendous speedups during training time (reducing training from up to 2 days for SDA to a few minutes with mSDA).

3.1.2 Recap of Learning from Multiple Domains

Similar to the setup we introduced in section 2.2.2, we are still aiming at generalizing to a *target* distribution for which we have few or no labels. However, in this case, we assume there are multiple *source* domains, and each is associated with a distinct distribution.

Figure 3.1 shows a Venn diagram that illustrates the setup we have in learning from multiple source domains. Let $\mathcal{X} \in \mathcal{R}^d$ denote the feature space, and \mathcal{Y} the output space. Let \mathcal{D}_T denote a joint distribution on $\mathcal{X} \times \mathcal{Y}$. It is the *target* distribution we are ultimately interested in. We assume there are N distinct source domains. Each source domain $S_j, j = 1, \dots, N$ is associated with an unknown underlying distribution \mathcal{D}_{S_j} . In the example of figure 3.1, two source domains are available.

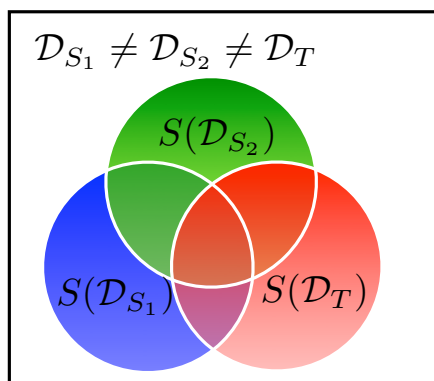


Figure 3.1: Venn diagram on learning from multiple source domains.

Ben-David et al. [10] extends their theory to the case of multiple source domains, in which the labeled data do not have to come from a single source domain, but multiple domains of

different distribution. A similar bound can be reached, however, the assumption needs to be strengthened. In this case, they assume there exists a hypothesis h^* which has low error on both α -weighted combination of sources and the target domain. The bound obtained is depending on the divergence of the target distribution and the mixture of sources.

Notation. Each source domain $S_j, j = 1, \dots, N$ is associated with a distinct underlying distribution \mathcal{D}_{S_j} on $\mathcal{X} \times \mathcal{Y}$, from which we can sample i.i.d a set of labeled data $L_{S_j} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{n_{S_j}}, y_{n_{S_j}})\}$ as well as a much larger set of unlabeled data $U_{S_j} = \{\mathbf{x}_{n_{S_j}+1}, \dots, \mathbf{x}_{m_{S_j}}\}$. Note that the labeled set might be empty for some source domains. From the target distribution \mathcal{D}_T , we can sample a small (or empty) set of labeled examples $L_T = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{n_T}, y_{n_T})\}$, and another set of unlabeled data $U_T = \{\mathbf{x}_{n_T+1}, \dots, \mathbf{x}_{m_T}\}$ with $m_T \gg n_T$.

3.1.3 Background and Related Work

We assume that our data originates from N source domains S_1, \dots, S_N and one target domain T . From each source domains, we sample unlabeled data $U_{S_j} = \{\mathbf{x}_1, \dots, \mathbf{x}_{m_{S_j}}\} \subset \mathcal{R}^d, j = 1, \dots, N$. We follow the setup in [69] and assume that part of the data from one source domain, w.l.o.g., S_1 , come with labels $L_{S_1} = \{y_1, \dots, y_{n_{S_1}}\}$. Whereas from the target domain we are only able to sample data *without labels* $U_T = \{\mathbf{x}_1, \dots, \mathbf{x}_{m_T}\} \subset \mathcal{R}^d$. We *do not* assume that these domains use identical features and we pad all input vectors with zeros to make both domains be of equal dimensionality d . Our goal is to learn a classifier $h \in \mathcal{H}$ that accurately predicts the labels of the data from the target domain T .

Structural Correspondence Learning

The first class of methods that is closely related to ours are the domain adaptation algorithms we discussed in section 2.2.3, in particular the Structural Correspondence Learning (SCL) method of Blitzer et al. [21]. SCL also learns a joint source/target representation for domain adaptation in an unsupervised fashion. The difference is that SCL requires the

identification of pivot features, which appears frequently in both domains and behave similarly, to put domain specific words in correspondence. The low-rank representation learned in SCL essentially encodes the covariance between non-pivot features and the pivot features. As we are going to detail later that single-layer mSDA also learns the correlations between *all* the features. In other words, mSDA no longer uses pivot features, which can be hard to identify as pointed out the authors [21]. We learn more robust and powerful features by introducing corruption and by stacking multiple layers. Meanwhile, we are able to keep the computational time comparable to SCL with our new framework.

Stacked Denoising Autoencoder

The second class of methods that are closely related is the autoencoders, which motivates our algorithm. Various forms of autoencoders have been developed in the deep learning literature [131, 6, 91, 97, 159, 126]. In its simplest form, an autoencoder has two components, an encoder $h(\cdot)$ maps an input $\mathbf{x} \in \mathcal{R}^d$ to some hidden representation $h(\mathbf{x}) \in \mathcal{R}^{d_h}$, and a decoder $g(\cdot)$ maps this hidden representation back to a reconstructed version of \mathbf{x} , such that $g(h(\mathbf{x})) \approx \mathbf{x}$. The parameters of the autoencoders are learned to minimize the reconstruction error, measured by some loss $\ell(\mathbf{x}, g(h(\mathbf{x})))$. Choices for the loss include squared error or Kullback-Leibler divergence when the feature values are in $[0, 1]$.

Denoising Autoencoders (DAs) incorporate a slight modification to this setup and corrupt the inputs before mapping them into the hidden representation. They are trained to reconstruct (or *denoise*) the original input \mathbf{x} from its corrupted version $\tilde{\mathbf{x}}$ by minimizing $\ell(\mathbf{x}, g(h(\tilde{\mathbf{x}})))$. Typical choices of corruption include additive isotropic Gaussian noise or binary masking noise. As in [159], we use the latter and set a fraction of the features of each input to *zero*. This is a natural choice for bag-of-word representations of texts, where typical class-specific words can be missing due to the writing style of the author or differences between train and test domains.

The stacked denoising autoencoder (SDA) of [159] stacks several DAs together to create higher-level representations, by feeding the hidden representation of the t^{th} DA as input into the $(t + 1)^{th}$ DA. The training is performed greedily, layer by layer.

Feature Generation. Many researchers have seen autoencoders as a powerful tool for automatic discovery and extraction of nonlinear features. For example, Lee et al. [97] demonstrate that the hidden representations computed by either all or partial layers of a convolutional neural network (CNN) make excellent features for classification with SVMs. The pre-processing with a CNN improves the generalization by increasing robustness against noise and label-invariant transformations.

Glorot et al. [69] successfully apply SDAs to extract features for domain adaptation in document sentiment analysis. The authors train an SDA to reconstruct the input vectors (ignoring the labels) on the union of the source and target data. A classifier (*e.g.* a linear SVM) trained on the resulting feature representation $h(\mathbf{x})$ transfers significantly better from source to target than one trained on \mathbf{x} directly. Similar to CNNs, SDAs also combine correlated input dimensions, as they reconstruct removed feature values from uncorrupted features. It is shown that SDAs are able to disentangle hidden factors which explain the variations in the input data, and automatically group features in accordance with their relatedness to these factors [69]. This helps transfer across domains as these generic concepts are invariant to domain-specific vocabularies.

As an intuitive example, imagine that we classify product reviews according to their sentiments. The source data consists of *book* reviews, the target of *kitchen appliances*. A classifier trained on the original source never encounters the bigram “energy efficient” during training and therefore assigns zero weight to it. In the learned SDA representation, the bigram “energy efficient” would tend to reconstruct, and be reconstructed by, co-occurring features, typically of similar sentiment (*e.g.* “good” or “love”). Hence, the source-trained classifier can assign weights even to features that never occur in its original domain representation, which are “re-constructed” by the SDA.

Although SDAs generate excellent features for domain adaptation, they have several drawbacks: 1) Training with (stochastic) gradient descent is slow and hard to parallelize (although a dense-matrix GPU implementation exists [12] and an implementation based on reconstruction sampling exists [53] for sparse inputs); 2) There are several hyper-parameters (learning rate, number of epochs, noise ratio, mini-batch size and network structure), which need to be set by cross validation — this is particularly expensive as each individual run can take several hours; 3) The optimization is inherently non-convex and dependent on its initialization.

3.1.4 marginalized Stacked Denoising Autoencoders

In this section we introduce a modified version of SDA, which preserves its strong feature learning capabilities, and alleviates the concerns mentioned above through speedups of several orders of magnitudes, fewer meta-parameters, faster model-selection and layer-wise convexity.

Single-layer Denoiser

The basic building block of our framework is a one-layer denoising autoencoder. We take the unlabeled inputs $\mathbf{x}_1, \dots, \mathbf{x}_n$ from $U = U_{S_1} \cup \dots \cup U_{S_1} \cup U_T$ ($n = m_{S_1} + \dots + m_{S_N} + m_T$) and corrupt them by random feature removal — each feature is set to 0 with probability $p \geq 0$. Let us denote the corrupted version of \mathbf{x}_i as $\tilde{\mathbf{x}}_i$. As opposed to the two-level *encoder* and *decoder* in SDA, we reconstruct the corrupted inputs with a single mapping $\mathbf{W} : \mathcal{R}^d \rightarrow \mathcal{R}^d$, that minimizes the squared reconstruction loss

$$\frac{1}{2n} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{W}\tilde{\mathbf{x}}_i\|^2. \quad (3.1)$$

To simplify notation, we assume that a constant feature is added to the input, $\mathbf{x}_i = [\mathbf{x}_i; 1]$, and an appropriate bias is incorporated within the mapping $\mathbf{W} = [\mathbf{W}, \mathbf{b}]$. The constant feature is *never* corrupted.

The solution to (3.1) depends on which features of each input are randomly corrupted. To lower the variance, we perform multiple passes over the training set, each time with different corruption. We solve for the \mathbf{W} that minimizes the overall squared loss

$$\mathcal{L}_{sq}(\mathbf{W}) = \frac{1}{2mn} \sum_{j=1}^m \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{W}\tilde{\mathbf{x}}_{i,j}\|^2, \quad (3.2)$$

where $\tilde{\mathbf{x}}_{i,j}$ represents the j^{th} corrupted version of the original input \mathbf{x}_i .

Let us define the design matrix $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathcal{R}^{d \times n}$ and its m -times repeated version as $\bar{\mathbf{X}} = [\mathbf{X}, \dots, \mathbf{X}]$. Further, we denote the corrupted version of $\bar{\mathbf{X}}$ as $\tilde{\mathbf{X}}$. With this notation,

Algorithm 3.1 mDA in MATLABTM.

```
function [W,h]=mDA(X,p);
X=[X;ones(1,size(X,2))];
d=size(X,1);
q=[ones(d-1,1).*(1-p); 1];
S=X*X';
Q=S.*(q*q');
Q(1:d+1:end)=q.*diag(S);
P=S.*repmat(q',d,1);
W=P(1:end-1,:)/(Q+1e-5*eye(d));
h=tanh(W*X);
```

the loss in eq. (3.1) reduces to

$$\mathcal{L}_{sq}(\mathbf{W}) = \frac{1}{2nm} \text{tr} \left[\left(\bar{\mathbf{X}} - \mathbf{W}\tilde{\mathbf{X}} \right)^\top \left(\bar{\mathbf{X}} - \mathbf{W}\tilde{\mathbf{X}} \right) \right]. \quad (3.3)$$

The solution to (3.3) can be expressed as the well-known closed-form solution for ordinary least squares [16]:

$$\mathbf{W} = \mathbf{P}\mathbf{Q}^{-1} \text{ with } \mathbf{Q} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top \text{ and } \mathbf{P} = \bar{\mathbf{X}}\tilde{\mathbf{X}}^\top. \quad (3.4)$$

In practice this can be computed as a system of linear equations, without the costly matrix inversion. Please refer to section 1.3.3 for a brief introduction to least square problems.

Marginalized Denoising Autoencoder

The larger m is, the more corruptions we average over. Ideally we would like $m \rightarrow \infty$, effectively using infinitely many copies of noisy data to compute the denoising transformation \mathbf{W} .

By the weak law of large numbers, the matrices \mathbf{P} and \mathbf{Q} , as defined in (3.3), converge to their expected values as m becomes very large. If we are interested in the limit case, where $m \rightarrow \infty$, we can derive the expectations of \mathbf{Q} and \mathbf{P} , and express the corresponding mapping \mathbf{W} as

$$\mathbf{W} = E[\mathbf{P}]E[\mathbf{Q}]^{-1}. \quad (3.5)$$

In the remainder of this section, we compute the expectations of these two matrices. For now, let us focus on

$$E[\mathbf{Q}] = \sum_{i=1}^n E[\tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^\top]. \quad (3.6)$$

An off-diagonal entry in the matrix $\tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^\top$ is uncorrupted if the two features α and β both “survived” the corruption, which happens with probability $(1-p)^2$. For the diagonal entries, this holds with probability $1-p$. Let us define a vector $\mathbf{q} = [1-p, \dots, 1-p, 1]^\top \in \mathcal{R}^{d+1}$, where \mathbf{q}_α represents the probability of a feature α “surviving” the corruption. As the constant feature is never corrupted, we have $\mathbf{q}_{d+1} = 1$. If we further define the scatter matrix of the original uncorrupted input as $\mathbf{S} = \mathbf{X}\mathbf{X}^\top$, we can express the expectation of the matrix Q as

$$E[\mathbf{Q}]_{\alpha,\beta} = \begin{cases} \mathbf{S}_{\alpha\beta} \mathbf{q}_\alpha \mathbf{q}_\beta & \text{if } \alpha \neq \beta \\ \mathbf{S}_{\alpha\beta} \mathbf{q}_\alpha & \text{if } \alpha = \beta \end{cases}. \quad (3.7)$$

Similarly, we obtain the expectation of \mathbf{P} in closed-form as $E[\mathbf{P}]_{\alpha\beta} = \mathbf{S}_{\alpha\beta} \mathbf{q}_\beta$.

With the help of these expected matrices, we can compute the reconstructive mapping \mathbf{W} directly in closed-form without ever explicitly constructing a single corrupted input $\tilde{\mathbf{x}}_i$. We refer to this algorithm as marginalized Denoising Autoencoder (mDA). Algorithm 3.1 shows a 10-line MATLABTM implementation. The mDA has several advantages over traditional denoisers: 1) It requires only a single sweep through the data to compute the matrices $E[\mathbf{Q}], E[\mathbf{P}]$; 2) Training is convex and a globally optimal solution is guaranteed; 3) The optimization is performed in non-iterative closed-form.

Nonlinear feature generation and stacking

Arguably two of the key contributors to the success of the SDA are its *nonlinearity* and the *stacking* of multiple layers of denoising autoencoders to create a “deep” learning architecture. Our framework has the same capabilities.

In SDAs, the nonlinearity is injected through the nonlinear *encoder* function $h(\cdot)$, which is learned together with the reconstruction weights \mathbf{W} . Such an approach makes the training procedure highly non-convex and requires iterative procedures to learn the model parameters. To preserve the closed-form solution from the linear mapping in equation (3.3) we insert

Algorithm 3.2 mSDA in MATLABTM.

```
function [Ws,hs]=mSDA(X,p,l);
    [d,n]=size(X);
    Ws=zeros(d,d+1,l);
    hs=zeros(d,n,l+1);
    hs(:,:,1)=X;
    for t=1:l
        [Ws(:,:,t), hs(:,:,t+1)]=mDA(hs(:,:,t),p);
    end;
```

nonlinearity into our learned representation *after* the weights \mathbf{W} are computed. A nonlinear squashing-function is applied on the output of each mDA. Several choices are possible, including sigmoid, hyperbolic tangent, $\tanh()$, or the rectifier function [113]. Throughout this work, we use the $\tanh()$ function.

Inspired by the layer-wise stacking of SDA, we stack several mDA layers by feeding the output of the $(t-1)^{th}$ mDA (after the squashing function) as the input into the t^{th} mDA. Let us denote the output of the t^{th} mDA as \mathbf{h}^t and the original input as $\mathbf{h}^0 = \mathbf{x}$. The training is performed greedily layer by layer: each map \mathbf{W}^t is learned (in closed-form) to reconstruct the previous mDA output \mathbf{h}^{t-1} from all possible corruptions and the output of the t^{th} layer becomes $\mathbf{h}^t = \tanh(\mathbf{W}^t \mathbf{h}^{t-1})$. In our experiments, we found that even without the nonlinear squashing function, stacking still improves the performance. However, the nonlinearity improves over the linear stacking significantly. We refer to the stacked denoising algorithm as marginalized Stacked Denoising Autoencoders (mSDA). Algorithm 3.2 shows a 8-lines MATLABTM implementation of mSDA.

mSDA for Domain Adaptation

We apply mSDA to domain adaptation by first learning features in an unsupervised fashion on the union of the source and target data sets. One observation reported in [69] is that if multiple domains are available, sharing the unsupervised pre-training of SDA across all domains is beneficial compared to pre-training on the source and target only. We observe a similar trend with our approach. The results reported in section 3.1.6 are based on features learned on data from all available domains. Once a mSDA is trained, the output of all layers,

after squashing, $\tanh(\mathbf{W}^t \mathbf{h}^{t-1})$, combined with the original features \mathbf{h}^0 , are concatenated and form the new representation. All inputs are transformed into the new feature space. A linear Support Vector Machine (SVM) [34] is then trained on the transformed source inputs and tested on the target domain. There are two meta-parameters in mSDA: the corruption probability p and the number of layers l . In our experiments, both are set with 5-fold cross validation on the labeled data from the *source* domain. As the mSDA training is almost instantaneous, this grid search is almost entirely dominated by the SVM training time.

3.1.5 Extension to High Dimensional Data

Many data sets (*e.g.* bag-of-words text documents) are naturally high dimensional. As the dimensionality increases, hill-climbing approaches used in SDAs can become prohibitively expensive. In practice, a work-around is to truncate the input data to the $r \ll d$ most common features [69]. Unfortunately, this prevents SDAs from utilizing important information found in rarer features. (As we show in section 3.1.6, including these rarer features leads to significantly better results.) High dimensionality also poses a challenge to mSDA, as the system of linear equations in (3.5) of complexity $O(d^3)$ becomes too costly. In this section we describe how to approximate this calculation with a simple division into $\frac{d}{r}$ sub-problems of $O(r^3)$.

We combine the concept of “pivot features” from [21] and the use of most-frequent features from [69]. Instead of learning a single mapping $\mathbf{W} \in \mathcal{R}^{d \times (d+1)}$ to reconstruct all corrupted features, we learn *multiple mappings* but only reconstruct the $r \ll d$ most frequent features (here, $r = 5000$). For an input \mathbf{x}_i we denote the shortened r -dimensional vector of only the r most-frequent features as $\mathbf{z}_i \in \mathcal{R}^r$. We perform this reconstruction with S random non-overlapping sub-sets of input features. Without loss of generality, we assume that the feature-dimensions in the input space are in random order and divide-up the input vectors as $\mathbf{x}_i = [\mathbf{x}_i^1 \top, \dots, \mathbf{x}_i^S \top]^\top$. For each one of these sub-spaces we learn an independent mapping \mathbf{W}^s which minimizes

$$\mathcal{L}_s(\mathbf{W}^s) = \frac{1}{2n} \sum_{i=1}^n \sum_{s=1}^S \|\mathbf{z}_i - \mathbf{W}^s \tilde{\mathbf{x}}_i^s\|^2. \quad (3.8)$$

Each mapping \mathbf{W}^s can be solved in closed-form as in (3.5), following the method described in section 3.1.4. We define the output of the first layer in the resulting mSDA as the average

of all reconstructions,

$$\mathbf{h}^1 = \tanh \left(\frac{1}{S} \sum_{s=1}^S \mathbf{W}^s \mathbf{x}^s \right). \quad (3.9)$$

Once the first layer, of dimension $r \ll d$, is built, we can stack multiple layers on top of it using the regular mSDA as described in section 3.1.4 and Algorithm 3.2. It is worth pointing out that, although features might be separated in different sub-sets within the first layer, they can still be combined in subsequent layers of the mSDA.

3.1.6 Experimental Results

In this section, we evaluate mSDA on the *Amazon reviews* benchmark data sets [21] together with several other algorithms for representation learning and domain adaptation.

Dataset. The dataset contains more than 340,000 reviews from 25 different types of products from Amazon.com. For simplicity (and comparability), we follow the convention of [42, 69] and only consider the binary classification problem whether a review is positive (higher than 3 stars) or negative (3 stars or lower). As mSDA and SDA focus on feature learning, we use the raw bag-of-words (bow) unigram/bigram features as their input. To be fair to other algorithms that we compare to, we also pre-process with tf-idf [133] and use the transformed feature vectors as their input if that leads to better results. Finally, we remove five domains which contain less than 1,000 reviews.

Different domains in the complete set vary substantially in terms of number of instances and class distribution. Some domains (books and music) have hundreds of thousands of reviews, while others (food and outdoor) have only a few hundred. There are a total of 380 possible transfer tasks (*e.g.* *Apparel* \rightarrow *Baby*). The proportion of negative examples in different domains also differs greatly. To counter the effect of class- and size-imbalance, a more controlled smaller dataset was created by Blitzer et al. [19], which contains reviews of four types of products: books, DVDs, electronics, and kitchen appliances. Here, each domain consists of 2,000 labeled inputs and approximately 4,000 unlabeled ones (varying slightly between domains) and the two classes are exactly balanced. Table 3.1 contains the statistics on the complete set as well as the control set. Almost all prior work provides results

only on this smaller set with its more manageable *twelve* transfer tasks. We focus most of our comparative analysis on this smaller set but also provide results on the entire data for completeness.

Table 3.1: Statistics of the large and small set of the Amazon review dataset [19].

DOMAIN	LABELED	UNLABELED (TEST)	NEG. INPUTS
COMPLETE (LARGE) SET			
APPAREL	4470	4470	14.52%
BABY	2046	2045	21.46%
BEAUTY	1314	1314	15.94%
BOOKS	27169	27168	12.09%
CAMERA	2652	2652	16.35%
DVDs	23044	23044	14.16%
ELECTRONICS	10197	10196	21.94%
FOOD	692	691	13.02%
GROCERY	1238	1238	13.57%
HEALTH	3254	3253	21.25%
JEWELRY	982	982	14.82%
KITCHEN	9233	9233	20.96%
MAGAZINES	1195	1195	22.64%
MUSIC	62181	62181	8.33%
OUTDOOR	729	729	20.71%
SOFTWARE	1033	1032	37.72%
SPORTS	2679	2679	18.78%
TOYS	6318	6318	19.67%
VIDEO	8695	8694	13.64%
VIDEOGAME	720	720	17.15%
CONTROLLED (SMALL) SET			
BOOKS	2000	4465	50%
DVDs	2000	3586	50%
ELECTRONICS	2000	5681	50%
KITCHEN	2000	5945	50%

Methods. As *baseline*, we train a linear SVM on the raw bag-of-words representation of the labeled *source* and test it on *target*. We also include the results of the same setup with dense features obtained by projecting the entire data set (labeled and unlabeled *source+target*) onto a low-dimensional sub-space with PCA (we refer to this setting as *PCA*). Besides these two baselines, we evaluate the efficacy of a linear SVM trained on features learned by mSDA and two alternative feature learning algorithms, Structural Correspondence Learning (*SCL*) [21] and 1-layer¹² *SDA* [69]. Finally, we also compare against *CODA* [42], a state-of-the-art domain adaptation algorithm which is based on sample- and feature-selection, applied to tf-idf features. For CODA, SDA and SCL we use implementations provided by the authors. All hyper-parameters are set by 5-fold cross validation on the source training set¹³.

Metrics. Following Glorot et al. [69], we evaluate our results with the *transfer error* $e(S, T)$ and the *in-domain error* $e(T, T)$. The *transfer error* $e(S, T)$ denotes the classification error of a classifier trained on the labeled *source* data and tested on the unlabeled *target* data. The *in-domain error* $e(T, T)$ denotes the classification error of a classifier that is trained on the labeled *target* data and tested on the unlabeled *target* data. Similar to Glorot et al. [69] we measure the performance of a domain adaptation algorithm in terms of the *transfer loss*, defined as $e(S, T) - e_b(T, T)$, where $e_b(T, T)$ defines the in-domain error of the baseline. In other words, the transfer loss measures how much higher the error of an *adapted* classifier is in comparison to a linear SVM that is trained on actual *labeled target* bow data.

The various domain-adaptation tasks vary substantially in difficulty, which is why we do not average the transfer losses (which would be dominated by a few most difficult tasks). Instead, we average the *transfer ratio*, $e(S, T)/e_b(T, T)$, the ratio of the *transfer error* over the *in-domain error*. As with the *transfer loss*, a lower *transfer ratio* implies better domain adaptation.

Timing. For timing purposes, we ignore the time of the SVM training and only report the mSDA or SDA training time. As both algorithms are unsupervised, we do not re-train for

¹²We were only able to obtain the 1-layer implementation from the authors. Anecdotally, multiple-layer *SDA* only leads to small improvements on this benchmark set but increases the training time drastically.

¹³We keep the default values of some of the parameters in SCL, *e.g.* the number of stop-words removed and stemming parameters — as they were already tuned for this benchmark set by the authors.

different transfer tasks within a benchmark set — instead we learn one representation on the union of all domains. CODA [41] does not take advantage of data besides source and target and we report the average training time per transfer task.¹⁴ All experiments were conducted on an off-the-shelf desktop with dual 6-core Intel i7 CPUs clocked at 2.66Ghz.

Comparison with Related Work

In the first set of experiments, we use the setting from [69] on the small Amazon benchmark set. The input data is reduced to only the 5,000 most frequent terms of unigrams and bigrams as features.

Comparison per task. Figure 3.2 presents a detailed comparison of the transfer loss across the twelve domain adaptation tasks using the various methods mentioned. The reviews are from the domains *Books*, *Kitchen appliances*, *Electronics*, *DVDs*. A linear SVM trained on the features generated by SDA and mSDA clearly outperform all the other methods. mSDA and SDA have the advantage over the CODA algorithm we introduced in section 2.2 as they make use of the unlabeled data from multiple source domains. For several tasks, the transfer loss goes to negative — in other words, a SVM trained on the transformed *source* data has higher accuracy than one trained on the original *target* data. This is a strong indication that the learned new representation bridges the gap between domains. It is worth pointing out that in ten out of the twelve tasks mSDA achieves a lower transfer-loss than SDA.

Timing. Figure 3.3 (left) depicts the transfer ratio as a function of training time required for different algorithms, averaged over 12 tasks. It compare the results of mSDA with baseline, PCA, SCL, CODA and SDA. The time is plotted in log scale. We can make three observations: 1) SDA outperforms all other related work in terms of transfer-ratio, but is also the slowest to train (more than 5 hours of training time). 2) SCL and PCA are relatively fast, but their features cannot compete in terms of transfer performance. 3) The training time of mSDA is two orders of magnitude faster that of SDAs ($180\times$ speedup), with

¹⁴In CODA, the feature splitting and classifier training are inseparable and we necessarily include both in our timing.

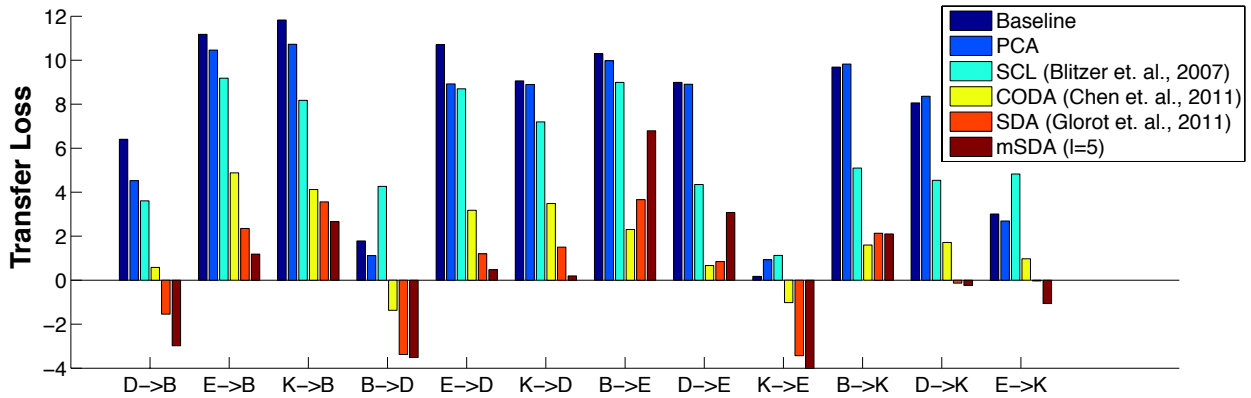


Figure 3.2: Comparison of mSDA and existing works across all twelve domain adaptation task in the small Amazon review dataset.

comparable transfer ratio. Training one layer of mDA on all 27,677 documents from the small set requires less than 25 seconds. A 5-layer mSDA requires less than 2 minutes to train, and the resulting feature transformation achieves slightly better transfer ratio than SDAs.

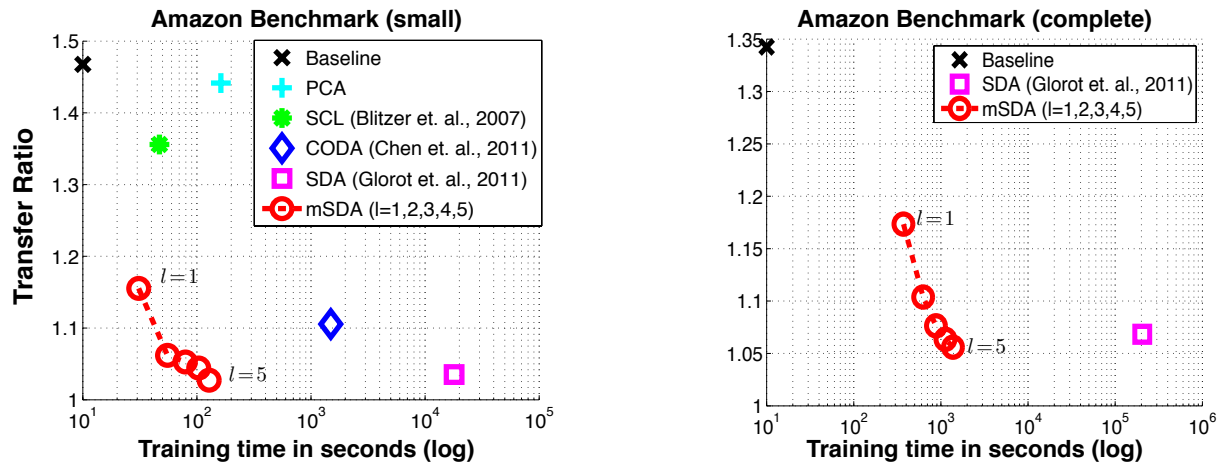


Figure 3.3: Transfer ratio and training times on the small (*left*) and full (*right*) Amazon Benchmark data. Results are averaged across the twelve and 380 domain adaptation tasks in the respective data sets (5,000 features).

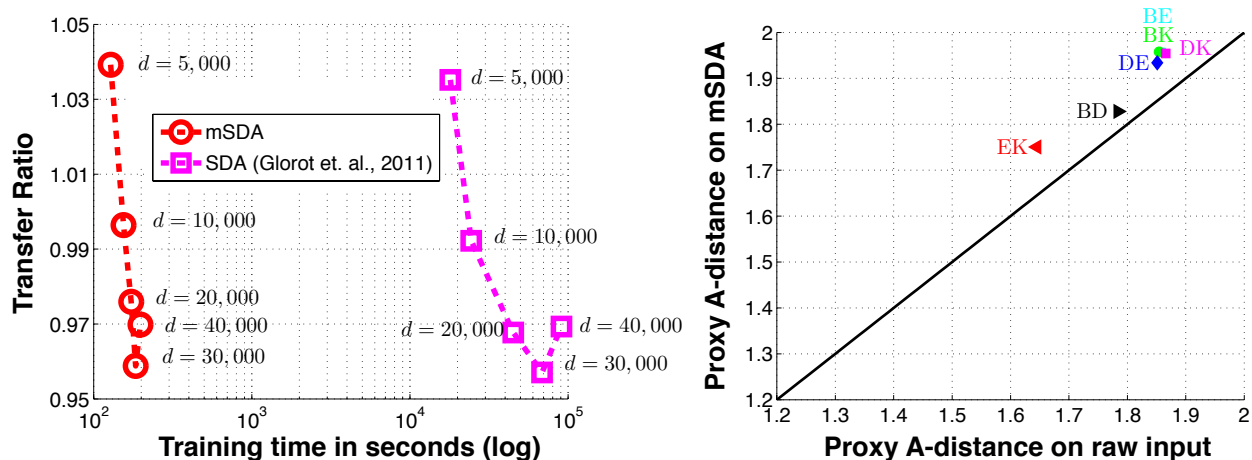


Figure 3.4: *Left*: Transfer ratio as a function of the input dimensionality (terms are picked in decreasing order of their frequency). *Right*: Besides domain adaptation, mSDA *also* helps in domain *recognition* tasks.

Large scale results. To demonstrate the capabilities of mSDA to scale to large data sets, we also evaluate it on the complete set with $n = 340,000$ reviews from 20 domains and a total of 380 domain adaptation tasks (see right plot in Figure 3.3). We compare mSDA to SDA (1-layer). The large set is more heterogenous in terms of the number of domains, domain size and class distribution than the small set and both the transfer error and transfer ratio are averaged across 380 tasks. Nonetheless, a similar trend can be observed. The transfer ratio reported in Figure 3.15 (right) corresponds to averaged transfer errors of (*baseline*) 13.93%, (*one-layer SDA*) 10.50%, (*mSDA, $l = 1$*) 11.50%, (*mSDA, $l = 3$*) 10.47%, (*mSDA, $l = 5$*) 10.33%. With only one layer, mSDA performs a little worse than *SDA* but reduces the training time from over two days to about five minutes ($700\times$ speedup). With three layers, mSDA matches the transfer-error and transfer-ratio of *SDA* and still only requires 14 minutes of training time ($230\times$ speedup).

Further Analysis

In addition to comparison with prior work, we also analyze various other aspects of mSDA.

Low-frequency features. Prior work often limits the input data to the most frequent features [69]. We use the modification from section 3.1.5 to scale mSDA (5-layers) up to high dimensions and include less-frequent uni-grams and bi-grams in the input (small Amazon set). In the case of SDA we make the first layer a dimensionality reducing transformation from d dimensions to 5000. The left plot in Figure 3.4 shows the performance of mSDA and SDA as the input dimensionality increases (words are picked in decreasing order of their frequency). The transfer ratio is computed relative to the baseline with $d = 5000$ feature. Clearly, both algorithms benefit from having more features up to 30,000. mSDA matches the transfer-ratio of SDA consistently and, as the dimensionality increases, gains even higher speed-up. With 30,000 input features, SDA requires over one day and mSDA only 3 minutes ($458\times$ speedup).

Transfer distance. Ben-David et al. [9] suggest the Proxy-A-distance (PAD) as a measure of how different two domains are from each other. The metric is defined as $2(1 - 2\epsilon)$, where ϵ is the generalization error of a classifier (a linear SVM in our case) trained on the binary classification problem to distinguish inputs *between* the two domains. The right plot in Figure 3.4 shows the PAD before and after mSDA is applied. Surprisingly, the distance *increases* in the new representation — *i.e.* distinguishing between two domains becomes *easier* with the mSDA features. We explain this effect through the fact that mSDA is unsupervised and learns a generally better representation for the input data. This helps both tasks, distinguishing between domains and sentiment analysis (*e.g.* in the electronic-domain mSDA might interpolate the feature “dvd player” from “blue ray”, both are not particularly relevant for sentiment analysis but might help distinguish the review from the *book* domain.). Glorot et al. [69] observe a similar effect with the representations learned with SDA.

General Trends

In summary, we observe a few general trends across all experiments: 1) With one layer, mSDA is up to three orders of magnitudes faster but slightly less expressive than the original SDA. This can be attributed to the fact that mSDA has no hidden layer. 2) There is a clear trend that additional “deep” layers improve the results significantly (here, up to five layers). With additional layers, the mSDA features reach (and surpass) the accuracy of 1-layer SDA and

still obtain a several hundred-fold speedup. 3) The mSDA features help diverse classification tasks, domain classification and sentiment analysis, and can be trained very efficiently on high-dimensional data.

3.1.7 Conclusion

Although mSDA first and foremost marginalizes out the corruption in SDA training, the two algorithms differ in several profound ways: First, the mDA layers do not have hidden nodes — this allows a closed-form solution with substantial speed-ups but might entail limitations that still need to be investigated. Second, mSDA only has *two* free meta-parameters, controlling the amount of noise as well as the number of layers to be stacked, which greatly simplifies the model selection. Finally, leveraging on the analytic tractability of linear regression, the parameters of an DA are trained to optimally denoise *all possible* corrupted training inputs — arguably “*infinitely many*”. This is practically infeasible for SDAs.

During our experimentation, we found that the sparsity in the raw feature representation is important for successful application of mSDA. For text data, it is natural to start from the sparse bag-of-word representation. For dense inputs, such as vision data, it is wise to first transform the original input to some sparse representation, *e.g.*, bag-of-visual-word, and then apply mSDA.

We hope that our work on mSDA will inspire future research on efficient training of SDA, beyond domain adaptation, and impact a variety of research problems. The fast training time, the capability to scale to large and high-dimensional data and implementation simplicity make mSDA a promising method with appeal to a large audience within and beyond machine learning.

3.2 Marginalized Corrupted Regularization

The goal of machine learning is to develop predictors that generalize well to test data. Ideally, this is achieved by training on an almost infinitely large training data set that captures all variations in the data distribution. In practical learning settings, however, we do not have

infinite data and our predictors may overfit. Common strategies to deal with overfitting include adding a regularizer to the training objective or by defining a prior over the model parameters and performing Bayesian inference. We propose a third, alternative approach: we extend the training set with infinitely many artificial generated training examples that are obtained by corrupting the original training data. We show that this approach is practical and efficient for a range of predictors and corruption models. We refer to our new approach as Marginalized Corrupted Features (MCF). It trains robust predictors by minimizing the expected value of the loss function under the corruption model. We show empirically on a variety of data sets that MCF classifiers can be trained efficiently, may generalize substantially better to test data, and are also more robust to feature deletion at test time.

The section is organized as follows. Section 3.2.2 recaps the setup in learning from corrupted distribution. Section 3.2.3 reviews several existing works on corrupting training data to deal with overfitting or corruption at test time, *i.e.*, some feature values are missing at test time due to sensor failure or computation constraints. Section 3.2.4 introduces Marginalized Corrupted Features (MCF) regularization. Section 3.2.5 presents an extensive experimental evaluation in which we compare MCF-regularized predictors with standard predictors on various document-classification problems, and in the “nightmare at test time scenario”. In Section 3.2.6, the results of our experiments are discussed in more detail.

3.2.1 Introduction

Dealing with overfitting is one of the key problems one encounters when training machine-learning models, especially when the number of training examples is small. An algorithm overfits if it performs substantially better on its training data than on held-out data. The classifier relates the label with patterns within the noise instead of within the data distribution. Three approaches are commonly used to combat overfitting: 1) *early-stopping* techniques monitor the performance of the model on a validation set during training, and stop the learning as soon as the validation performance deteriorates; 2) *regularization* techniques encourage the learning to find “simple” models by penalizing “complex” models, *e.g.*, models with large parameter values; and 3) *Bayesian* techniques define a prior distribution over models that favor simple models, and perform predictions by averaging over the model

posterior. Regularization and Bayesian techniques are often intimately related, in particular, because regularization techniques can sometimes be viewed as introducing a prior over models and performing maximum-likelihood estimation.

We propose a new alternative to counter overfitting. Instead of requiring the user to define prior distributions or regularizers over *model parameters*, which can be very counter intuitive, we focus on corruptions of the *data*. Our approach is based on the observation that overfitting would completely disappear if we were to train our models on infinite data drawn from the data distribution \mathcal{D} . In such a hypothetical scenario, it is impossible to overfit models; and even high-variance models, such as the nearest neighbor classifier [47], become close to optimal (*viz.* the error of a nearest neighbor classifier is twice the Bayes error). Unfortunately, a learning scenario in which we only obtain a finite training set is more realistic; here, some variations in the data distribution will not be captured and the learned model performs worse at test time than during training.

In many learning scenarios, we may however have some additional knowledge about the data distribution: we might know that certain *corruptions* of data instances do not affect their conditional label distributions. As an example, deleting a few words in a text document rarely changes its topic. With this prior knowledge, we can corrupt existing data to generate new artificial instances that resemble those sampled from the actual data distribution. In fact, we will *corrupt* the existing *finite* training examples with a fixed corrupting distribution to construct an *infinite* training set. Further, we show that for a wide range of learning models and noise distributions, it is practical to train models on such an infinite, augmented training set. We refer to the resulting framework as learning with *marginalized corrupted features* (MCF). So instead of approximating the exact statistics of \mathcal{D} with *finite* data, MCF learns from a slightly modified data distribution \mathcal{D}' with *infinite* data.

Burges and Schölkopf [28] explicitly augment the training set with additional examples that are corrupted through similar transformations. Although the simplicity of such an approach is appealing, it lacks elegance and the computational cost of processing the additional corrupted training examples is prohibitive for most real-world problems. In contrast, we show that it is efficient to train predictors on an infinite amount of corrupted copies of the training data by marginalizing out the corrupting distribution. In particular, we focus on empirical risk minimization and derive analytical solutions for the expected loss under a large family

of corrupting distributions for quadratic and exponential loss functions. This allows us to minimize the expected loss in computational time linear in the number of training examples. For logistic loss functions, which are used in many probabilistic models, we derive practical upper bounds on the expected value of the loss under the corrupting distribution.

Our augmented data distribution \mathcal{D}' is constructed using a simple stochastic rule: pick one of the finite training examples uniformly at random and transform it with some pre-defined corrupting distribution. Many corrupting distributions are possible, but in this work we focus on 1) Gaussian corruption, 2) Poisson corruption, and 3) (unbiased) blankout corruption (random deletion of features). The Gaussian corruption model is mainly of interest for continuous-valued data sets; special cases of MCF with Gaussian corruption have already been studied in the context of Parzen density estimation [120] and in the context of vicinal risk minimization [35]. The Poisson corruption model is of interest when the data comprises count vectors, *e.g.*, in document classification. Poisson corruption is particularly appealing as it introduces *no additional hyper-parameters* and, in our results, improves the test accuracy on almost all data sets that comprise count data. The blankout corruption model is of interest in data sets with heavy-tailed feature distributions, such as filter responses, and in settings where blankout noise is a known source of variance in the original data distribution \mathcal{P} —possibly unobserved in the training data. This happens, *e.g.*, in document classification from term-frequency vectors: a big portion of each document (especially after stop-word removal) is sampled from the tail of the power-law distribution, for which the blankout noise model is a surprisingly good approximation: it models the common case that some of the words related to the class of the document are missing, *e.g.* because the author used other synonyms. Blankout corruption is also of interest in the “nightmare at test time” scenario [67] in which some of the features are deleted during testing, *e.g.*, due to sensors failures or because the feature computation exceeds a time budget, and we wish to make predictions that are robust to this blankout noise. Different from previous work [14, 67, 142, 154], our setting focuses on random feature removal and not on the (in practice much rarer) worst-case adversarial setting.

In summary, we make the following contributions: 1) we introduce learning with marginalized corrupted features, a framework that trains robust classifiers by marginalizing out all possible feature corruptions from a pre-defined distribution; 2) we derive plug-in solutions for the quadratic, exponential, and logistic loss functions for a range of corrupting distributions,

which can be incorporated into learning algorithms out-of-the-box; and 3) on several real-world data sets, we show that training with MCF may lead to more robust classifiers and may substantially decrease generalization errors in various learning settings. Given the simplicity and elegance of MCF, it could become a valuable alternative to the common l_1 or l_2 -norm regularizers.

3.2.2 Recap of Learning from Corrupted Data

Again here we focus on scenarios when the training data from the relevant distribution is insufficient. In learning with corrupted distribution, we create additional training examples by applying feature dropout or corruption to the small set of training examples. The hope is that with properly chosen corrupting distributions, the artificially created examples will resemble real ones and can capture some variation in the testing distribution which was not captured in the small set of training examples. The artificially created examples can later be used as additional training examples to improve robustness.

Figure 3.5 shows a Venn diagram that illustrates the setup we have in learning from corrupted data. Let \mathcal{D} denote the testing distribution, from which we sampled the original small set of training examples. Let \mathcal{D}' denote the distribution of the corrupted data, which is created from the training examples by applying some corrupting distribution on each example independently. We assume that the support of the distribution of \mathcal{D}' and that of the testing distribution \mathcal{D} is the same. As \mathcal{D}' is created by applying a known corrupting distribution on each training example independently, the density function of \mathcal{D}' is known as well.

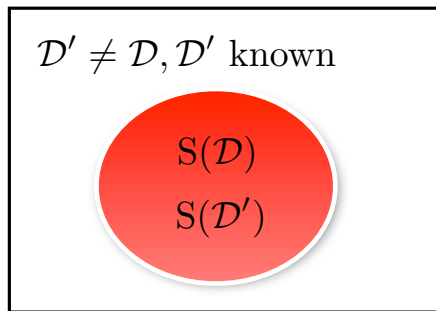


Figure 3.5: Venn diagram on learning from corrupted data.

Notation. Let $\mathcal{X} \in \mathcal{R}^d$ denote the feature space, and \mathcal{Y} the output space. Let \mathcal{D} denote the testing distribution on $\mathcal{X} \times \mathcal{Y}$. Let $L = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$ be a small set of training examples sampled i.i.d. from \mathcal{D} . Let $p(\tilde{\mathbf{x}}|\mathbf{x})$ denote the corrupting distribution we used to create the corrupted data. It generates another set of training examples $L' = \{(\tilde{\mathbf{x}}_{i1}, y_i), \dots, (\tilde{\mathbf{x}}_{im}, y_i), i = 1, \dots, n\} \subseteq \mathcal{X} \times \mathcal{Y}$ where each example $\tilde{\mathbf{x}}_{im} \sim p(\tilde{\mathbf{x}}_{im}|\mathbf{x}_i)$. We are going to introduce a new learning framework which learns on infinitely many such corrupted data, *i.e.*, $m \rightarrow \infty$ to find a classifier that predict accurately on testing data sampled from \mathcal{D} .

3.2.3 Background and Related work

There are two motivations for training a classifier with MCF: 1. to reduce the effects of overfitting during *training*; and 2. to combat the effect of data corruption during *testing*. Both motivations connect with different lines of prior work, and in this section we discuss both of them.

Corruption during training

The initial publication by [28] has inspired various lines of work that *explicitly* corrupt training data during training. Most prominently, Vincent et al. [160, 161] propose to randomly blank out features in the training data that is used as input to autoencoders, whilst leaving the desired output (the original training data) unaltered. The resulting denoising autoencoder model is now commonly used as a building block in deep learning [103, 68, 108], and blankout corruption is also increasingly applied on the hidden units of neural networks as a form of regularization [79]. Since autoencoders are non-linear, the marginalization over the corrupting distribution cannot be performed analytically in such models. Linear denoising autoencoders [43] can be viewed as a special case of MCF that aim to minimize the expected value of the reconstruction error under blankout corruption—and which are stacked in multiple layers. Herbrich and Graepel [77] propose an elegant generalization of SVMs that learns to be invariant to polynomial input transformations via a semi-definite programming formulation.

In online learning and bandit problems, various studies have shown that it is possible to learn from data that are subject to a (possibly unknown) corrupting distribution [62, 1, 31]. For instance, Cesa-Bianchi et al. [32] show that noise in the data does not affect the convergence rate of online learners. A similar regret bound is proven by Rostamizadeh et al. [129] for online learning under the presence of randomly missing features (*i.e.* blankout corruption). These studies differ from our work in that they show that (unknown) corruptions in the data do not *impair* learning performance too much. By contrast, our work shows that adding corruptions to the data can actually *improve* learning performance.

Corruption during testing

Several prior studies consider *implicit* approaches for data sets that are subject to corruptions during *test time* (this is also known as “the nightmare at test-time” scenario). Most of these studies propose to minimize the loss under an adversarial worst-case scenario. In particular, Globerson and Roweis [67] propose a minimax-formulation in which the loss is minimized assuming maximum “damage” through corruption. By contrast, Dekel and Shamir [54] propose a linear-programming formulation that minimizes an approximation to the same quantity for margin-based predictors. Other studies [14, 142, 154, 166] also use minimax approaches that minimize losses under a worst-case scenario, but they focus on a somewhat different corruption model that adds a uniformly drawn constant, within a fixed interval, to each feature.

Chechik et al. [40] propose an algorithm that maximizes the margin in the subspace of the observed features for each training instance to be robust against random feature deletion. Teo et al. [152] generalize the worst-case scenario from simple feature corruption to obtain invariances to transformations such as image rotations or translations. Their framework incorporates several prior formulations on learning with invariants as special cases, most prominently Herbrich and Graepel [77].

Such previous work differs from our approach in that it focuses on adversarial scenarios and does not consider the corruption analytically in expectation. Further, they mostly focus on a very specific corruption model. In particular, existing approaches suffer from three main disadvantages: 1) they are complex and computationally expensive, 2) they minimize the

loss of a worst-case scenario that is very unlikely to be encountered in practice, and 3) they do not provide a flexible framework that can be used with a variety of models and corrupting distributions. By contrast, we propose a much simpler approach that scales linearly in the number of training samples, that considers an average-case instead of a worst-case scenario, and that can readily be used with a variety of loss functions and a large family of corruption models.

3.2.4 Learning with Marginalized Corrupted Features

To derive the *marginalized corrupted features* (MCF) framework, we start by defining a corrupting distribution that specifies how training observations \mathbf{x} are transformed into corrupted versions $\tilde{\mathbf{x}}$. Throughout this work, we assume that the corrupting distribution factorizes over dimensions and that each individual distribution P_E is a member of the natural exponential family. (We will see later that for MCF with quadratic loss functions, these assumptions may be relaxed.) Specifically, we assume a corrupting distribution of the form:

$$p(\tilde{\mathbf{x}}|\mathbf{x}) = \prod_{d=1}^D P_E(\tilde{x}_d|x_d; \eta_d), \quad (3.10)$$

where η_d represents user-defined hyperparameters of the corrupting distribution on dimension d . Corrupting distributions of interest, for P_E , include: 1. independent salt or “blankout” noise in which the d -th feature is randomly set to zero with probability q_d ; 2. bit-swap noise in which the value of the d -th bit is randomly swapped with probability q_d ; 3. independent Gaussian noise on the d -th feature with variance σ_d^2 ; 4. independent Laplace noise on the d -th feature with variance $2\lambda_d^2$; and 5. independent Poisson corruptions in which the d -th feature is used as the rate of the distribution. Note that the definition in (3.10) allows for different features to use arbitrary different corrupting distributions. In the remainder of this section, we propose an efficient way to train a classifier on corrupted inputs $\tilde{\mathbf{x}}$, sampled from the distribution $p(\tilde{\mathbf{x}}|\mathbf{x})\mathcal{P}(\mathbf{x})$. There are at least two motivations for doing so:

1. Nightmare at test-time. In the *nightmare at test-time* scenario, introduced by [67], there is an expectation that corruption will appear during test-time. Although the training

data is sampled from $\mathcal{P}(\mathbf{x})$, the test data will be sampled from the distribution $p(\tilde{\mathbf{x}}|\mathbf{x})\mathcal{P}(\mathbf{x})$. Training the classifier on the corrupted data corrects this distribution drift. This scenario appears, for example, in the context of search engines. A feature can be reliably collected for the training set, but during testing it is dropped if its computation time exceeds a pre-specified time limit. The frequency of these feature drop outs can be measured and incorporated into training with the appropriate drop out distribution. Similarly, one can imagine scenarios (*e.g.* in robotics) where features correspond to unreliable sensor readings.

2. Regularization. The more common motivation is improved generalization. Here, the corrupting distribution is meant to capture some of the variance inherent in $\mathcal{P}(\mathbf{x})$. In this setting, one must guarantee that the classifier is still applicable for the test case, where data is sampled directly from \mathcal{P} . More precisely, it must be *unbiased*. Let $h(\mathbf{x})$ be the classification function, we therefore require of the corrupting distribution that

$$\mathbb{E}[h(\tilde{\mathbf{x}})]_{p(\tilde{\mathbf{x}}|\mathbf{x})} = h(\mathbf{x}). \quad (3.11)$$

In other words, corrupting an input should not change its expected prediction and a test input obtains the expected prediction that the same input would obtain during training. In the linear case, the classifier is parameterized by a vector \mathbf{w} . For regression, the prediction function is defined as $h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ and we require $\mathbb{E}[\tilde{\mathbf{x}}] = \mathbf{x}$. For classification it is $h(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x})$, which necessitates the slightly weaker condition $\mathbb{E}[\tilde{\mathbf{x}}] = s(\mathbf{x})\mathbf{x}$ for any bounded $s(\mathbf{x}) > 0$. Table 3.2 showcases several biased and unbiased distributions.

3. Explicit corruption. Assume we are provided with a training data set $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ and a loss function $L(\mathbf{x}, y; \Theta)$, with model parameters Θ . A simple approach to approximately learn from the distribution $p(\tilde{\mathbf{x}}|\mathbf{x})\mathcal{P}(\mathbf{x})$, is to follow the spirit of Burges and Schölkopf [28] and corrupt each training sample M times, following (3.10). For each $\mathbf{x}_n \in \mathcal{D}$, this results in corresponding corrupted observations $\tilde{\mathbf{x}}_{nm}$ (with $m = 1, \dots, M$) and leads to the construction of a new data set $\tilde{\mathcal{D}}$ of size $|\tilde{\mathcal{D}}| = MN$. This new data set can be used for training in an empirical risk minimization framework, by minimizing the surrogate loss function:

$$\mathcal{L}(\tilde{\mathcal{D}}; \Theta) = \sum_{n=1}^N \frac{1}{M} \sum_{m=1}^M L(\tilde{\mathbf{x}}_{nm}, y_n; \Theta), \quad (3.12)$$

with $\tilde{\mathbf{x}}_{nm} \sim p(\tilde{\mathbf{x}}_{nm}|\mathbf{x}_n)$. Such approaches have recently become popular, in particular, in the deep-learning community as a way to regularize deep neural networks [79].

4. Marginalized Corrupted Feature. Although approaches that explicitly corrupt the training data are effective, they lack elegance and come with high computational costs: the minimization of $\mathcal{L}(\tilde{\mathcal{D}}; \Theta)$ scales linearly in the number of corrupted observations, *i.e.* it scales as $\mathcal{O}(NM)$. It is, however, of interest to consider the limiting case in which $M \rightarrow \infty$. In this case, we can apply the *weak law of large numbers* and rewrite $\frac{1}{M} \sum_{m=1}^M L(\tilde{\mathbf{x}}_m, y_m; \Theta)$ as its expectation [56, §2.10.2] to obtain the following surrogate loss:

$$\mathcal{L}(\mathcal{D}; \Theta) = \sum_{n=1}^N \mathbb{E}[L(\tilde{\mathbf{x}}_n, y_n; \Theta)]_{p(\tilde{\mathbf{x}}_n|\mathbf{x}_n)}. \quad (3.13)$$

Minimizing the expected value of the loss under the corruption model leads to a new approach for training predictors that we refer to as learning with marginalized corrupted features (MCF). MCF may lead to algorithms with an $\mathcal{O}(N)$ training complexity in situations in which the expectation in (3.13) is tractable, which is indeed the case for commonly used loss functions $L(\tilde{\mathbf{x}}_n, y_n; \Theta)$.

In the following, we show that for linear predictors that employ a quadratic or exponential loss function, the required expectations under $p(\tilde{\mathbf{x}}|\mathbf{x})$ in (3.13) can be computed analytically for all corrupting distributions in the natural exponential family. For linear predictors with logistic loss we derive a practical upper bound on the expected loss under $p(\tilde{\mathbf{x}}|\mathbf{x})$, which serves as surrogate loss.

Quadratic loss

Assuming a linear model parametrized by vector \mathbf{w} and a target variable y (for regression, y is continuous; for binary classification, $y \in \{-1, +1\}$), the expected value of the quadratic

loss under corrupting distribution $p(\tilde{\mathbf{x}}|\mathbf{x})$ is given by:

$$\begin{aligned}
\mathcal{L}(\mathcal{D}; \mathbf{w}) &= \sum_{n=1}^N \mathbb{E} \left[\left(\mathbf{w}^\top \tilde{\mathbf{x}}_n - y_n \right)^2 \right]_{p(\tilde{\mathbf{x}}_n|\mathbf{x}_n)} \\
&= \sum_{n=1}^N \mathbb{E} \left[\mathbf{w}^\top \tilde{\mathbf{x}}_n \tilde{\mathbf{x}}_n^\top \mathbf{w} - 2y_n \mathbf{w}^\top \tilde{\mathbf{x}}_n + y_n^2 \right]_{p(\tilde{\mathbf{x}}_n|\mathbf{x}_n)} \\
&= \mathbf{w}^\top \left(\sum_{n=1}^N \mathbb{E}[\tilde{\mathbf{x}}_n] \mathbb{E}[\tilde{\mathbf{x}}_n]^\top + V[\tilde{\mathbf{x}}_n] \right) \mathbf{w} - 2 \left(\sum_{n=1}^N y_n \mathbb{E}[\tilde{\mathbf{x}}_n] \right)^\top \mathbf{w} + N. \tag{3.14}
\end{aligned}$$

Herein, $V[x]$ denotes the variance of x , and all expectations are under $p(\tilde{\mathbf{x}}_n|\mathbf{x}_n)$. Note that irrespective of what corruption model is used, (3.14) is quadratic in \mathbf{w} , and hence convex. The optimal solution \mathbf{w}^* can be computed with a variation of the ordinary least squares closed form:

$$\mathbf{w}^* = \left(\sum_{n=1}^N \mathbb{E}[\tilde{\mathbf{x}}_n] \mathbb{E}[\tilde{\mathbf{x}}_n]^\top + V[\tilde{\mathbf{x}}_n] \right)^{-1} \left(\sum_{n=1}^N y_n \mathbb{E}[\tilde{\mathbf{x}}_n] \right). \tag{3.15}$$

Hence, to minimize the expected quadratic loss under the corruption model, we only need to compute the mean and variance of the corrupting distribution, which is practical for all exponential-family distributions.

Table 3.2 gives an overview of the probability density function (PDF), mean, and variance of five corrupting distributions of interest. These quantities can be plugged into eq. (3.14) to obtain the expected value under the corrupting distribution of the quadratic loss. For disambiguation, we refer to the unbiased version of the feature dropout as blankout corruption.

Special case. An interesting setting of MCF with quadratic loss occurs when the corrupting distribution $p(\tilde{\mathbf{x}}|\mathbf{x})$ is an isotropic Gaussian distribution with mean \mathbf{x} and variance $\sigma^2 \mathbf{I}$. For such a Gaussian corruption model, we obtain the standard l_2 -regularized quadratic loss with regularization parameter $\sigma^2 N$ that is used in ridge regression as special case [35]:

$$\mathcal{L}(\mathcal{D}; \mathbf{w}) = \mathbf{w}^\top \left(\sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top \right) \mathbf{w} - 2 \left(\sum_{n=1}^N y_n \mathbf{x}_n \right)^\top \mathbf{w} + \sigma^2 N \mathbf{w}^\top \mathbf{w} + N. \tag{3.16}$$

Table 3.2: The probability density function (PDF), mean and variance of corrupting distributions of interest.

	Name	PDF	$\mathbb{E}[\tilde{\mathbf{x}}_{nd}]_{p(\tilde{\mathbf{x}}_{nd} \mathbf{x}_{nd})}$	$\mathbf{V}[\tilde{\mathbf{x}}_{nd}]_{p(\tilde{\mathbf{x}}_{nd} \mathbf{x}_{nd})}$
biased	Dropout	$p(\tilde{x}_{nd} = 0) = q_d$ $p(\tilde{x}_{nd} = x_{nd}) = 1 - q_d$	$(1 - q_d)x_{nd}$	$q_d(1 - q_d)x_{nd}^2$
	Bit-swap	$p(\tilde{x}_{nd} = 1 - x_{nd}) = q_d$ $p(\tilde{x}_{nd} = x_{nd}) = 1 - q_d$	$q^{(1-x_{nd})}(1 - q)^{x_{nd}}$	$\begin{pmatrix} 1 - q^{(1-x_{nd})}(1 - q)^{x_{nd}} \\ q^{(1-x_{nd})}(1 - q)^{x_{nd}} \end{pmatrix}$
unbiased	Blankout	$p(\tilde{x}_{nd} = 0) = q_d$ $p(\tilde{x}_{nd} = \frac{1}{1-q_d}x_{nd}) = 1 - q_d$	x_{nd}	$\frac{q_d}{1-q_d}x_{nd}^2$
	Gaussian	$p(\tilde{x}_{nd} \mathbf{x}_{nd}) = \mathcal{N}(\tilde{x}_{nd} \mathbf{x}_{nd}, \sigma^2)$	x_{nd}	σ^2
	Laplace	$p(\tilde{x}_{nd} \mathbf{x}_{nd}) = Lap(\tilde{x}_{nd} \mathbf{x}_{nd}, \lambda)$	x_{nd}	$2\lambda^2$
	Poisson	$p(\tilde{x}_{nd} \mathbf{x}_{nd}) = Poisson(\tilde{x}_{nd} \mathbf{x}_{nd})$	x_{nd}	x_{nd}

Perhaps surprisingly, using MCF with Laplace corruption also leads to ridge regression with the regularization parameter taking value $2\lambda^2N$. Indeed, ridge regression arises for every unbiased corrupting distribution whose variance is not data-dependent.

Leave-one-out errors. A nice property of linear models employing quadratic loss is that they allow us to compute the leave-one-out error of the model from only the training predictions \hat{y}_n [3]. We show that this property still holds for MCF quadratic losses, irrespective of the corrupting distribution that is used. Defining the target vector $\mathbf{y} = [y_1, \dots, y_N]^T$ and the data matrix $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$, we can rewrite the prediction vector $\hat{\mathbf{y}} = [\hat{y}_1, \dots, \hat{y}_N]$ as:

$$\hat{\mathbf{y}} = \mathbf{X}^T \mathbf{w}^* = \mathbf{X}^T \left(\sum_{n=1}^N \mathbb{E}[\tilde{\mathbf{x}}_n] \mathbb{E}[\tilde{\mathbf{x}}_n]^T + V[\tilde{\mathbf{x}}_n] \right)^{-1} \left(\sum_{n=1}^N \mathbb{E}[\tilde{\mathbf{x}}_n] y_n \right) = \mathbf{H} \mathbf{y}, \quad (3.17)$$

where \mathbf{H} is often referred to as the hat matrix. If a single target value y_n is replaced by \hat{y}_n^{-n} , the prediction made by the model trained on all data except example n , the hat matrix \mathbf{H} does not change because it does not depend on \mathbf{y} . If we construct a new target vector \mathbf{z} with this replacement in it, $\mathbf{z} = [y_1, \dots, y_{n-1}, \hat{y}_n^{-n}, y_{n+1}, \dots, y_N]$, and use the simple fact that $\hat{y}_n^{-n} = \sum_{m=1}^N H_{nm} z_m$ (see Allen [3] for details) we obtain:

$$\hat{y}_n - \hat{y}_n^{-n} = \sum_{m=1}^N H_{nm} y_m - \sum_{m=1}^N H_{nm} z_m = H_{nn} y_n - H_{nn} \hat{y}_n^{-n}. \quad (3.18)$$

Using this equality, we can compute \hat{y}_n^{-n} without performing the minimization of the MCF quadratic loss – with example n left out – over \mathbf{w} :

$$\hat{y}_n^{-n} = \frac{H_{nn}y_n - \hat{y}_n}{H_{nn} - 1}. \quad (3.19)$$

This allows us to express the leave-one-out error LOO of the MCF quadratic loss model as:

$$LOO = \sum_{n=1}^N (y_n - \hat{y}_n^{-n})^2 = \sum_{n=1}^N \left(\frac{y_n - \hat{y}_n}{1 - H_{nn}} \right)^2. \quad (3.20)$$

Blankout corruption. We study the use blankout in MCF quadratic loss in more detail here to show what kind of regularizers it produces. By plugging the relevant quantities from Table 3.2 into (3.15), we obtain the following solution for \mathbf{w}^* with MCF blankout (assuming $\forall d : q_d = q$):

$$\mathbf{w}^* = \left(\sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top + q(1 - q)\mathbf{S} \right)^{-1} \left(\sum_{n=1}^N \mathbf{x}_n y_n \right), \text{ where: } \mathbf{S} = \left(\sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top \right) \circ \mathbf{I} \quad (3.21)$$

where \circ denotes an element-wise Hadamard product. In words, the matrix \mathbf{S} consists of only the diagonal of the scatter matrix. Consequently, MCF blankout corruption has the effect of multiplying the diagonal of the scatter matrix by a factor $(1 + q - q^2)$. This makes the MCF regularizer *data-dependent*: the regularization is stronger on dimensions that on average have a larger norm. This may be desirable in situations in which the data dimensions live on different scales as happens, for example, in word-count features. This is in contrast to the *data independent* l_2 -regularizer, that simply adds a fixed value λ to the diagonal of the scatter matrix—treating all features alike.

Poisson corruption. Using Poisson corruption in MCF quadratic loss leads to:

$$\mathbf{w}^* = \left(\sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^\top + \sum_{n=1}^N \mathbf{x}_n \right)^{-1} \left(\sum_{n=1}^N \mathbf{x}_n y_n \right). \quad (3.22)$$

Poisson corruption thus leads to a *parameterless, additive, data-dependent* regularizer that adds larger values to the diagonal of the Gram matrix for features that occur more on average.

Table 3.3: Moment-generating functions (MGFs) of corrupting distributions of interest.

Distribution	$\mathbb{E}[\exp(-\mathbf{y}_n \mathbf{w}_d \tilde{\mathbf{x}}_{nd})]_{p(\tilde{\mathbf{x}}_{nd} x_{nd})}$
Blankout noise	$q_d + (1 - q_d) \exp(-y_n w_d \frac{1}{1 - q_d} x_{nd})$
Gaussian noise	$\exp(-y_n w_d x_{nd} + \frac{1}{2} \sigma^2 y_n^2 w_d^2)$
Laplace noise	$(1 - \lambda^2 y_n^2 w_d^2)^{-1} \exp(-y_n w_d x_{nd})$
Poisson noise	$\exp(x_{nd} (\exp(-y_n w_d) - 1))$

(Note that Poisson corruption can only be used on non-negative data, $\forall n, d : x_{nd} \geq 0$, so the contribution of the regularizer is always non-negative.) The main difference between blankout and Poisson corruption is in how strongly features with large values are regularized; the multiplicative blankout corruption regularizes common features more than the additive Poisson corruption.

Exponential loss

Whilst quadratic losses are of interest to regression problems, they seem less appropriate for use in classification. Exponential loss is a loss function that is commonly used for classification, *e.g.*, in AdaBoost [63]. Assuming a label variable $y \in \{0, 1\}$, the expected value of the exponential loss under corruption model $p(\tilde{\mathbf{x}} | \mathbf{x})$ is given by:

$$\begin{aligned} \mathcal{L}(\mathcal{D}; \mathbf{w}) &= \sum_{n=1}^N \mathbb{E} [\exp(-y_n \mathbf{w}^T \tilde{\mathbf{x}}_n)]_{p(\tilde{\mathbf{x}}_n | \mathbf{x}_n)} \\ &= \sum_{n=1}^N \prod_{d=1}^D \mathbb{E} [\exp(-y_n w_d \tilde{x}_{nd})]_{p(\tilde{x}_{nd} | x_{nd})}, \end{aligned} \quad (3.23)$$

which we refer to as MCF exponential loss. Note that in the derivation, we used the assumption that the corruption is independent across features. The above equation can be recognized as a product of moment-generating functions $\mathbb{E}[\exp(t_{nd} \tilde{x}_{nd})]$ with $t_{nd} = -y_n w_d$. By definition, the moment-generating function (MGF) can be computed for all corrupting distributions that are member of the natural exponential family. An overview of the moment-generating functions for some corrupting distributions of interest is given in Table 3.3. These quantities can be plugged into equations (3.23) and (3.26) to obtain the expected value of

the loss (or surrogate) under the corrupting distribution of the exponential and logistic loss functions, respectively.

The derivation above can readily be extended to a multi-class exponential loss with K classes by replacing the weight vector \mathbf{w} by a $D \times K$ weight matrix \mathbf{W} , and by replacing the labels y by label vectors $\mathbf{y} = \{1, -\frac{1}{K-1}\}^K$ with $\sum_{k=1}^K y_k = 0$ [173].

The MCF exponential loss remains convex for many (but not all) corrupting distributions. Noting that $f(x)$ is convex iff $\forall x : \frac{\partial^2 f(x)}{\partial x^2} \geq 0$ and noting that sums of convex functions are themselves convex, it is straightforward to verify that MCF exponential loss is convex for blankout corruption, bit-swap corruption, and Gaussian corruption. MCF exponential loss is non-convex for Laplace corruption and Poisson corruption. Unlike the minimization of MCF quadratic loss, the minimization of MCF exponential losses cannot be done in closed form and needs to be performed using gradient-descent techniques. Motivated by Sha and Pereira [139], we used an conjugate gradient based optimizer (as introduced in section 1.3.1) to minimize MCF exponential loss in this study.

Blankout corruption. As an illustrative example, we work out MCF exponential loss with blankout corruption:

$$\mathcal{L}(\mathcal{D}; \mathbf{w}) = \sum_{n=1}^N \prod_{d=1}^D (q_d + (1 - q_d) \exp(-y_n w_d x_{nd})). \quad (3.24)$$

The gradient of this loss function is given by:

$$\frac{\partial \mathcal{L}}{\partial w_d} = - \sum_{n=1}^N (1 - q_d) \exp(-y_n w_d x_{nd}) y_n x_{nd} \prod_{d' \neq d} (q_{d'} + (1 - q_{d'}) \exp(-y_n w_{d'} x_{nd'})). \quad (3.25)$$

Although the complexity of training with MCF remains $\mathcal{O}(N)$, evaluating this gradient requires more computation than evaluating the gradient of a standard exponential loss: in practice, the computation of the MCF exponential loss gradient is about twice as expensive when computation from the loss (3.24) is reused efficiently.

Logistic loss

In the case of the logistic loss, the solution to (3.13) cannot be computed in closed form. Instead, we derive an upper bound, which can be minimized as a surrogate loss:

$$\begin{aligned} \mathcal{L}(\mathcal{D}; \mathbf{w}) &= \sum_{n=1}^N \mathbb{E} \left[\log \left(1 + \exp \left(-y_n \mathbf{w}^T \tilde{\mathbf{x}}_n \right) \right) \right]_{p(\tilde{\mathbf{x}}_n | \mathbf{x}_n)} \\ &\leq \sum_{n=1}^N \log \left(1 + \prod_{d=1}^D \mathbb{E} [\exp(-y_n w_d \tilde{x}_{nd})]_{p(\tilde{x}_{nd} | x_{nd})} \right). \end{aligned} \quad (3.26)$$

Herein, we have made use of Jensen's inequality to upper-bound $\mathbb{E}[\log(z)]$. We again recognize a product of MGFs, which can be computed in closed-form for corrupting distributions in the natural exponential family (see Table 3.3). The upper bound on the expected logistic loss (3.26) is convex whenever the moment-generating function is log-linear in w_d and an exponentiated affine function, *e.g.* for blankout and bit-swap corruption.

As before, the above derivation can readily be extended to multi-class logistic loss. (This is achieved by redefining the labels y to be label vectors of the form $\mathbf{y} \in \{0, 1\}^K$ with $\sum_{k=1}^K y_k = 1$; and by defining the loss as the logarithm of the softmax probability of the correct prediction.)

Poisson corruption. As an illustrative example, we show the upper bound of the logistic loss (3.26), where inputs are corrupted with the Poisson distribution:

$$\mathcal{L}(\mathcal{D}; \mathbf{w}) \leq \sum_{n=1}^N \log \left(1 + \exp \left(\sum_{d=1}^D x_{nd} (\exp(-y_n w_d) - 1) \right) \right). \quad (3.27)$$

As with all MCF losses that use Poisson corruption, this regularized loss does not have any additional hyper-parameters. Further, the sum over all features can still be computed efficiently for sparse data by summing only over non-zero entries in \mathbf{x}_n . As for MCF exponential loss, we perform the minimization of this surrogate loss with an conjugate gradient based

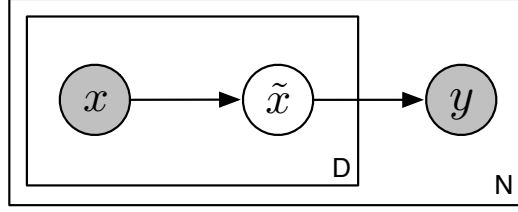


Figure 3.6: Graphical model interpretation of marginalized corrupted features.

optimizer. The gradient of (3.27) is given by:

$$\frac{\partial \mathcal{L}}{\partial w_d} = - \frac{\exp\left(\sum_{d' \neq d} x_{nd'} (\exp(-y_n w_{d'}) - 1)\right)}{1 + \exp\left(\sum_{d=1}^D x_{nd} (\exp(-y_n w_d) - 1)\right)} \exp(y_n w_d) y_n x_{nd}. \quad (3.28)$$

It is fair to say that in practice, the computation of the gradient of this MCF logistic loss takes approximately $10\times$ longer than the computation of a regular logistic loss gradient (due to extra exponentiations and multiplications in the gradient computation).

Alternative upper bound. Indeed, Jensen’s inequality is not the only bound that may be used to derive an upper bound on MCF logistic loss; in preliminary experiments, we also considered bounds of the form:

$$\begin{aligned} \mathcal{L}(\mathcal{D}; \mathbf{w}) &= \sum_{n=1}^N \mathbb{E} [\log(1 + \exp(-y_n \mathbf{w}^T \tilde{\mathbf{x}}_n))] \\ &= \sum_{n=1}^N \mathbb{E} [-\lambda y_n \mathbf{w}^T \tilde{\mathbf{x}}_n] + \mathbb{E} [\log(\exp(\lambda y_n \mathbf{w}^T \tilde{\mathbf{x}}_n) (1 + \exp(-y_n \mathbf{w}^T \tilde{\mathbf{x}}_n)))] \\ &\leq \sum_{n=1}^N \log (\mathbb{E}[(\exp(\lambda y_n \mathbf{w}^T \tilde{\mathbf{x}}_n))] + \mathbb{E}[\exp((\lambda - 1) y_n \mathbf{w}^T \tilde{\mathbf{x}}_n)]) - \lambda y_n \mathbf{w}^T \mathbb{E}[\tilde{\mathbf{x}}_n], \end{aligned} \quad (3.29)$$

which contains two moment-generating functions (with $t = \lambda y_n \mathbf{w}$ and $t = (\lambda - 1) y_n \mathbf{w}$, respectively). Minimizing (3.29) over the hyper parameter $\lambda \in [0, 1]$, using a grid or gradient search, leads to a potentially tighter bound than that in (3.26) (note that the special case $\lambda = 0$ equals (3.26), as a result of which the alternative bound is always at least as tight).

Graphical model interpretation. Figure 3.6 shows a simple Bayesian network that interprets minimizing MCF logistic loss as maximum likelihood estimation of the parameters. Shaded circles represent observed variables; non-shaded circles represent latent variables; and plates denote independent repetitions. The arrow from x_{nd} to \tilde{x}_{nd} represents the corrupting distribution and the arrow from \tilde{x}_{nd} to y_n represents the logistic regression model. In this network, $p(\tilde{x}_d|x_d)$ denotes the corrupting distributions, and $p(y|\tilde{\mathbf{x}})$ is given by a simple linear logistic regressor: $p(y|\tilde{\mathbf{x}}) = 1/(1+\exp(-y\mathbf{w}^T\tilde{\mathbf{x}}))$. Marginalizing out the corrupted data $\tilde{\mathbf{x}}$, we obtain:

$$\begin{aligned} p(y_n|\mathbf{x}_n) &= \mathbb{E} \left[\frac{1}{1 + \exp(-y_n \mathbf{w}^T \tilde{\mathbf{x}}_n)} \right]_{p(\tilde{\mathbf{x}}_n|\mathbf{x}_n)} \\ &\geq \frac{1}{1 + \mathbb{E} [\exp(-y_n \mathbf{w}^T \tilde{\mathbf{x}}_n)]_{p(\tilde{\mathbf{x}}_n|\mathbf{x}_n)}}, \end{aligned} \quad (3.30)$$

where we have used the fact that $\frac{1}{1+z}$ is convex on $[0, \infty)$ and that $\forall z : 1 + \exp(z) > 1$ to lower bound $p(y|\mathbf{x})$ using Jensen's inequality.

Because the corrupting distribution $p(\tilde{\mathbf{x}}|\mathbf{x})$ factorizes over dimensions, the log-likelihood ℓ of data \mathcal{D} can thus be lower bounded by:

$$\ell(\mathcal{D}; \mathbf{w}) \geq - \sum_{n=1}^N \log \left(1 + \prod_{d=1}^D \mathbb{E} [\exp(-y_n w_d \tilde{x}_{nd})]_{p(\tilde{x}_{nd}|x_{nd})} \right). \quad (3.31)$$

It is straightforward to verify that maximizing this lower bound on the log-likelihood is indeed identical to minimizing (3.26). Further, it is possible to develop a similar graphical model for MCF-quadratic loss by setting $p(y_n|\tilde{\mathbf{x}}_n) = \mathcal{N}(y_n|\mathbf{w}^T\tilde{\mathbf{x}}_n, \sigma^2)$.

The graphical model interpretation of MCF also suggests an alternative way to make predictions with models trained using MCF, *viz.* by evaluating (3.30). In practice, this amounts to generating infinitely many corrupted copies of the test data using $p(\tilde{\mathbf{x}}|\mathbf{x})$ and *averaging the over the predictions for all corrupted test points*. While corrupting test data during prediction may sound counterintuitive at first, it does provide an elegant way to remove the restriction for MCF corruption distributions to be unbiased (especially in the case of regression). In classification, corrupting test data provides a natural way to measure the uncertainty of a prediction, even when the original predictor was non-probabilistic (much like conformal prediction; Shafer and Vovk [140]).

3.2.5 Experiments

We evaluate MCF predictors on three tasks: 1) document classification based on word-count features using MCF with blankout and Poisson corruption; 2) image classification based on bag-of-visual-word features using MCF with blankout and Poisson corruption; and 3) classification of objects in the “nightmare at test time” scenario using MCF with blankout corruption. The three sets of experiments are described separately below. Code to reproduce the results of our experiments is available online¹⁵.

Document classification

We first test MCF predictors with blankout and Poisson corruption on document classification tasks. Specifically, we focus on three data sets: the Dmoz data set, the Reuters data set, and the Amazon review benchmark set [19].

Data sets. The Dmoz open directory (<http://www.dmoz.org>) contains a large collection of webpages arranged into a tree hierarchy. We use a subset consisting of $N = 8,980$ webpages from the $K = 16$ categories in the top level of the hierarchy. Each webpage is represented by a bag-of-words representation with $D = 16,498$ words. The Reuters document classification data set consists of $N = 8,293$ news articles that appeared on the Reuters newswire in 1987 belonging to $K = 65$ topics (documents corresponding to multiple topics were removed from the data). The bag-of-words representation contains $D = 18,933$ words for each document. The four Amazon data sets consist of approximately $N = 6,000$ reviews of four types of products: books, DVDs, electronics, and kitchen appliances. Each review is represented by a bag-of-words representation of the $D = 20,000$ most common words. On the Dmoz and Reuters data sets, the task is to classify the documents into one of the predefined categories. On the Amazon data set, the task is to decide whether a review is positive or negative.

¹⁵A Matlab implementation is available at <http://bit.ly/11bn2GG>.

Setup. On the Dmoz and Reuters data sets, we use fixed 75/25 training/test splits. On the Amazon data set, we follow the experimental setup of Blitzer et al. [19] by using a predefined division of the data into approximately 2,000 training examples and about 4,000 test examples (the exact numbers vary slightly between tasks). We perform experiments with linear classifiers that are trained using l_2 -regularized quadratic, exponential, and logistic loss functions. In all experiments, the amount of l_2 -regularization is determined via cross-validation. The minimization of the (expected) exponential and logistic losses is performed by running Mark Schmidt’s `minFunc`-implementation of L-BFGS until convergence or until a predefined maximum number of iterations is reached.

All our predictors included a bias term that is neither regularized nor corrupted. In our experiments with MCF using blankout corruption, we use the same noise level for each feature, *i.e.* we assume that $\forall d : q_d = q$. On all data sets, we first investigate the performance of MCF as a function of the corruption level q (but we still cross-validate over the l_2 -regularizer). In a second set of experiments, we cross-validate over the blankout corruption parameter q and study to what extent the performance (improvements) of MCF depend on the amount of available training data. (MCF with Poisson corruption has no additional hyper-parameters, as a result of which it requires no extra cross-validations.)

Results. Figure 3.7 shows the test error of our MCF predictors on all data sets as a function of the blankout corruption level q . Herein, corruption level $q = 0$ corresponds to the baseline predictors, *i.e.* to predictors that do not employ MCF at all. The results show: i) that MCF consistently improves over standard predictors for both blankout corruption (for all corruption levels q) and Poisson corruption on five out of six tasks; ii) that MCF with Poisson corruption leads to significant performance improvements over standard classifiers whilst introducing no additional hyperparameters; and iii) that the best performance tends to be achieved by MCF with blankout corruption with relatively high corruption levels are used, *i.e.* when q is in the order of 0.8 or 0.9.

The best-performing MCF classifiers reduce the test errors by up to 22% on the Amazon data if q is properly set. In many of the experiments with MCF-trained losses (in particular, when blankout corruption is used), we observe that the optimal level of l_2 -regularization is

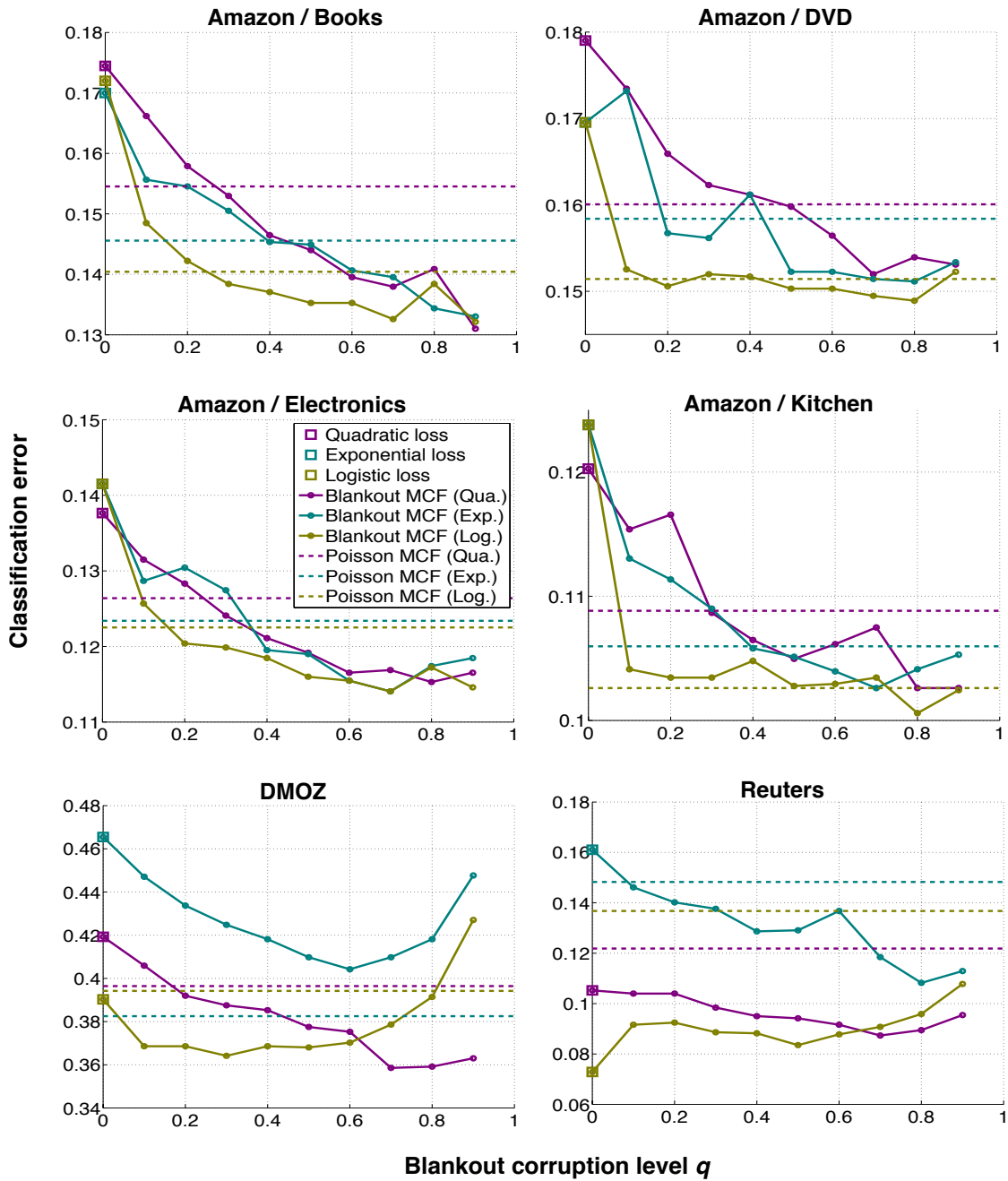


Figure 3.7: Classification errors of MCF predictors using blankout corruption – as a function of the blankout corruption level q – on all data sets for l_2 -regularized quadratic, exponential, and quadratic loss functions. The case of MCF with blankout corruption and $q = 0$ corresponds to a standard l_2 -regularized classifier. Figure best viewed in color.

0. This showcases the regularizing effect of MCF, which can render additional regularization superfluous.

Figure 3.8 presents the results of a second set of experiments on Dmoz and Reuters, in which we study how the performance of MCF depends on the amount of training data. For each training set size, we repeat the experiment five times with randomly sub-sampled training sets; the figure reports the mean test errors and the corresponding standard deviations. The results show that classifiers trained with MCF (solid curves) significantly outperform their counterparts without MCF (dashed curves). The performance improvement is consistent irrespective of the training set size, *viz.* up to 25% on the Dmoz data set.

Explicit vs. implicit feature corruption. Figure 3.9 shows the classification error on Amazon (books) when a classifier without MCF is trained on the data set with additional *explicitly* corrupted samples, as formulated in equation (3.12). Specifically, we use the blackout corruption model with q set by cross-validation for each setting, and we train the classifiers with quadratic loss and l_2 -regularization. The graph shows a clear trend that the error *decreases* when the training set contains more corrupted versions of the original training data, *i.e.* with higher M in equation (3.12). The graph illustrates that the best performance is obtained as M approaches infinity, which is equivalent to MCF with blackout corruption (big marker in the bottom right, with $q = 0.9$).

Image classification

We perform image-classification experiments with MCF on the CIFAR-10 data set [92], which is a subset of the 80 million tiny images [153]. The data set contains RGB images with 10 classes of size 32×32 , and consists of a fixed training set of 50,000 images and a fixed test set of 10,000 images.

Setup. We follow the experimental setup of Coates et al. [44]: we whiten the training images and extract a set of 7×7 image patches on which we apply k -means clustering (with $k = 2048$) to construct a codebook. Next, we slide a 7×7 pixel window over each image and identify the nearest prototype in the codebook for each window location. We construct an

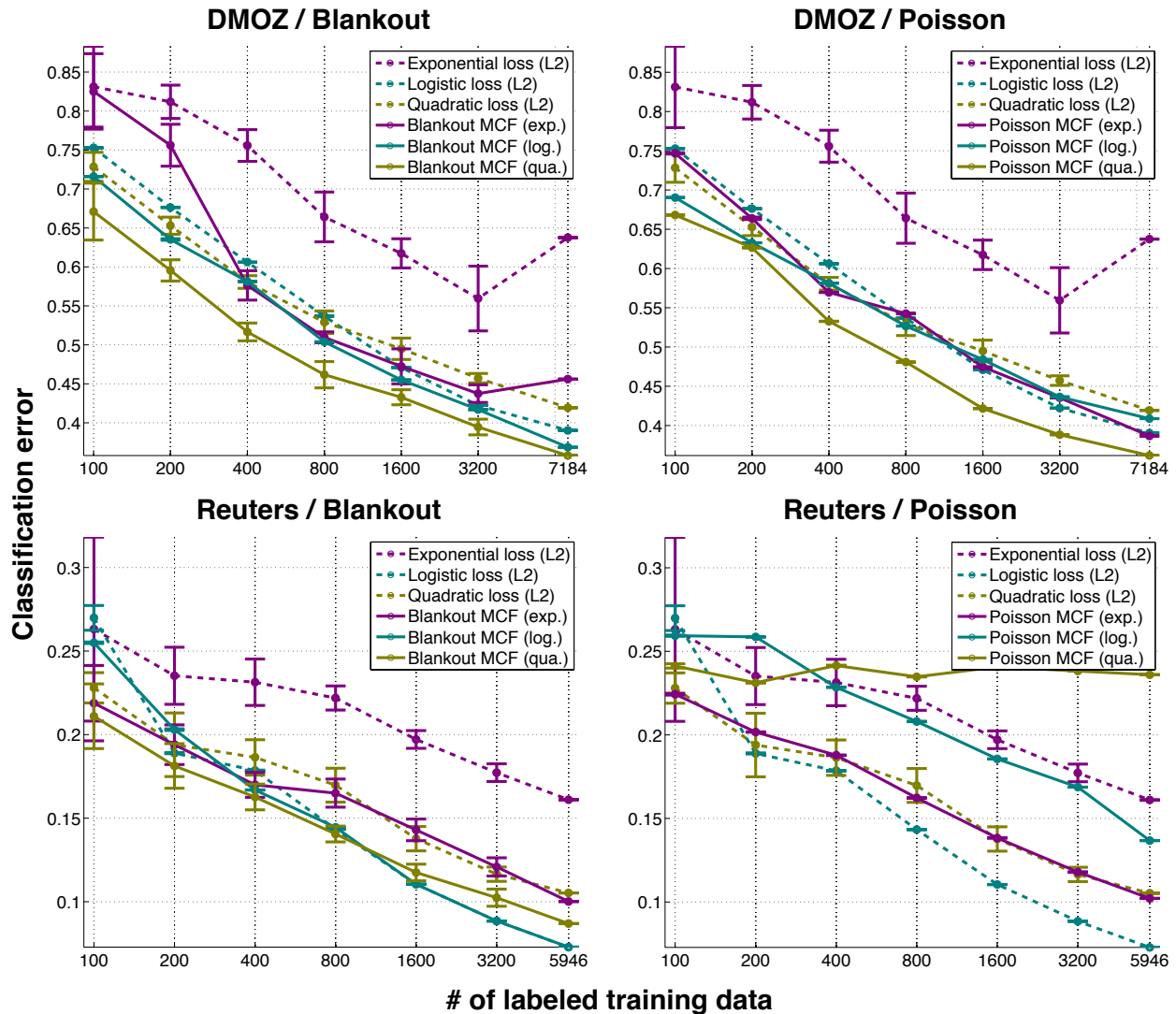


Figure 3.8: The performance of standard and MCF classifiers with blankout and Poisson corruption models as a function of training set size on the Dmoz and Reuters data sets. Both the standard and MCF predictors employ l_2 -regularization. Figure best viewed in color.

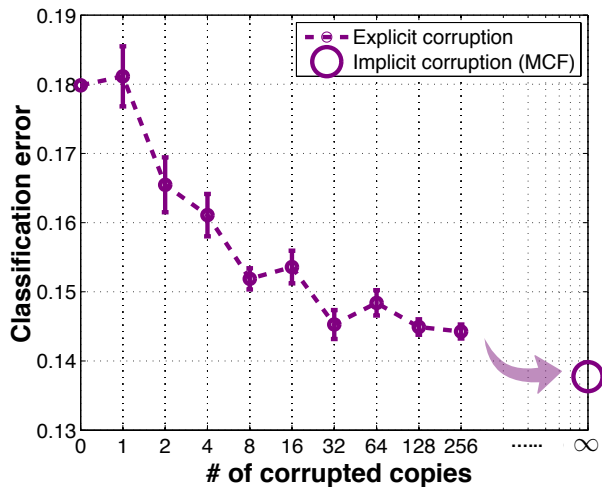


Figure 3.9: Comparison between MCF and explicitly adding corrupted examples to the training set (for quadratic loss) on the Amazon (books) data using blankout corruption. Training with MCF is equivalent to using infinitely many corrupted copies of the training data.

image descriptor¹⁶ by subdividing the image into four equally sized quadrants and counting the number of times each prototype occurs in each quadrant. This leads to a descriptor of dimensionality $D = 4 \times 2048$. We do not normalize the descriptors, because all images have the same size. We train MCF predictors with blankout and Poisson corruption on the full set of training images, cross-validating over a range of l_2 -regularization parameters. The generalization error is evaluated on the test set.

Results. The results are reported in Table 3.4. The baseline classifiers (without MCF) are comparable to the 68.8% accuracy reported by Coates et al. [44] with exactly the same experimental setup (except for exponential loss). The results illustrate the potential of MCF classifiers to improve the prediction performance on bag-of-visual-words features, in particular, when using quadratic or logistic loss in combination with a Poisson corruption model.

Although our focus in this section is to merely illustrate the potential of MCF on image classification tasks, it is worth noting that the best results in Table 3.4 match those of a

¹⁶This way of extracting the image features is referred to by Coates et al. [44] as k -means with hard assignment, average pooling, patch size 7×7 , and stride 1.

Table 3.4: Comparison of MCF and l_2 -regularization on classification errors obtained on the CIFAR-10 data set with simple spatial-pyramid bag-of-visual-word features.

	Quadr.	Expon.	Logist.
No MCF	32.6%	39.7%	32.5%
Poisson MCF	29.1%	39.5%	30.0%
Blankout MCF	32.3%	37.9%	29.4%

highly non-linear mean-covariance RBMs trained on the same data [124], despite our use of very simple visual features and of linear classifiers.

Nightmare at test time

To test the performance of our MCF predictors with blankout corruption under the “nightmare at test time” scenario, we perform experiments on the MNIST data set of handwritten digit images. The MNIST data set contains $N = 60,000$ training and 10,000 test images of size $D = 28 \times 28 = 784$ pixels, and comprises $K = 10$ classes.

Setup. We train our predictors on the full training set, and evaluate their performance on versions of the test set in which a certain percentage of the pixels are randomly blanked out, *i.e.* set to zero. We compare the performance of our MCF-predictors (using blankout corruption) with that of standard predictors that use l_1 or l_2 -regularized quadratic, exponential, logistic, and hinge loss. As before, we use cross-validation to determine the optimal value of the regularization parameter. For MCF predictors, we also cross-validate over the blankout corruption level q (again, we use the same noise level for each feature, *i.e.* $\forall d : q_d = q$). In addition to the comparisons with standard predictors, we also compare the performance of MCF with that of FDROP [67], which is a state-of-the-art algorithm for the “nightmare at test time” setting that minimizes the hinge loss under an adversarial worst-case scenario (see section 3.2.3).

The performances are reported as a function of the feature-deletion percentage in the test set, *i.e.* as a function of the probability with which a pixel in the test set is switched off. Following the experimental setting of Globerson and Roweis [67], we perform the cross-validation for

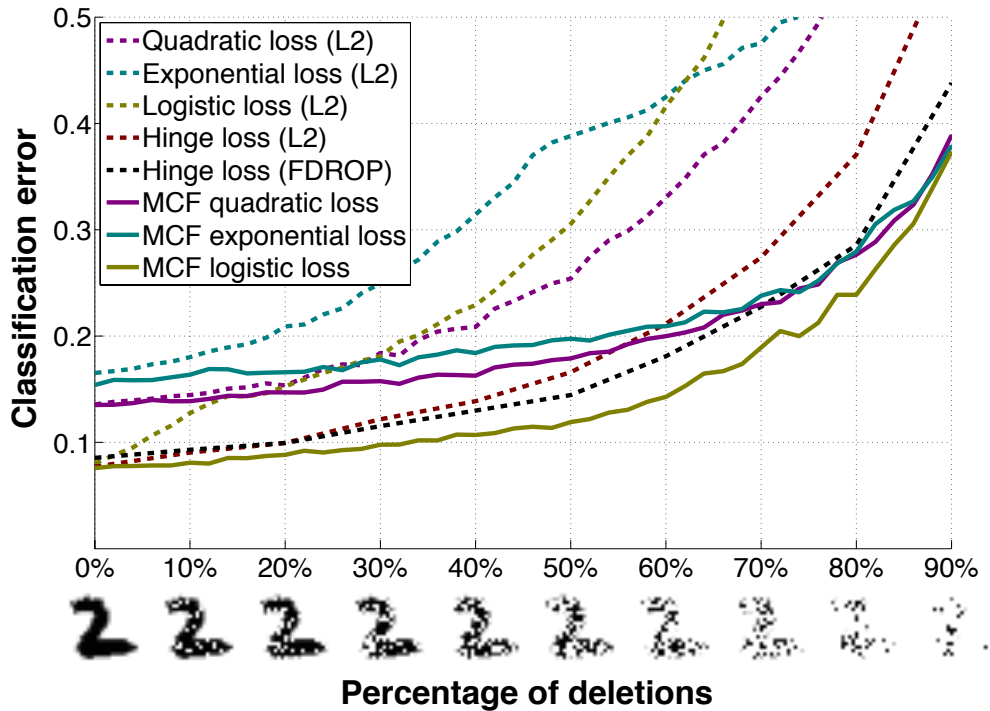


Figure 3.10: Evaluation on the “Nightmare at test-time” scenario. Classification errors of standard and MCF predictors with a blankout corruption model – trained using three different losses – and of FDROP [67] on the MNIST data set using the “nightmare at test time” scenario. Classification errors are represented on the y -axis, whereas the amount of features that are deleted out at test time is represented on the x -axis. The images of the digit illustrate the amount of feature deletions applied on the digit images that are used as test data. Figure best viewed in color.

each deletion percentage independently, *i.e.* we create a small validation set with the same feature-deletion level and use it to determine the best regularization parameters and blankout corruption level q for that percentage of feature deletions.

Results. Figure 3.10 shows the performance of our predictors as a function of the percentage of deletions in the test images. The figure shows the performance for all three loss functions with MCF (solid lines) and without MCF (dashed lines). The performance of a standard predictor using hinge loss is shown as a red dashed line; the performance of FDROP (a state-of-the-art method for this learning setting) is shown as a black dashed line. The results presented in Figure 3.10 clearly illustrate the ability of MCF with blankout corruption

to produce predictors that are more robust to the “nightmare at test time” scenario: MCF improves the performance substantially for all three loss functions considered. For instance, in the case in which 50% of the pixels in the test images is deleted, the performance improvements obtained using MCF for quadratic, exponential, and logistic loss are 40%, 47%, and 60%, respectively. Further, the results also indicate that MCF-losses may outperform FDROP; in particular, our average-case logistic loss outperforms FDROP’s worst-case hinge loss across the board¹⁷. This is particularly impressive as FDROP uses the hinge loss, which performs surprisingly better than the alternative losses on this data set (the improvement of FDROP over the generic hinge-loss is in fact relatively modest). This result suggests that it is better to consider an average-case scenario than a worst-case scenario in the “nightmare at test time” setting.

3.2.6 Conclusion

We presented an approach to learn classifiers by marginalizing corrupted features (MCF). Specifically, MCF trains predictors by introducing corruption on the training examples, which is marginalized out in the expectation of the loss function. We minimize the expected loss with respect to the model parameters. Our experimental results show that MCF predictors with blackout and Poisson corruption perform very well in the context of bag-of-words features. MCF with Poisson corruption is particularly interesting for such count features, as it improves classification performances without introducing any additional hyper-parameters. As a disclaimer, care must be taken when applying MCF with Poisson corruption on data sets with outliers. The Poisson corruption may strongly emphasize outliers in the expected loss because the variance of a Poisson distribution is identical to its mean, and because we use loss functions that are not robust to outliers. A simple solution to this problem may be to redefine the corruption distribution to $p(\tilde{x}_d|x_d) = Pois(\tilde{x}_{nd}|\min\{x_{nd}, u\})$ for some cutoff parameter $u \geq 0$.

Throughout our experiments with MCF on exponential and logistic losses, the weight of the l_2 -regularization term, which was set by cross-validation, typically ended up very close to zero. This implies that the regularizing effect of MCF corruption is sufficient even for high

¹⁷Quadratic and exponential losses perform somewhat worse because they are less appropriate for linear classifiers, but even they outperform FDROP for large numbers of feature deletions in the test data.

dimensional data. In comparison with traditional regularization, MCF corruption shines in two ways: i) it often yields superior classification performance and ii) it can be much more intuitive to set parameters about data corruption than about model hyper parameters. Further, MCF with blankout corruption also appears to prevent *weight undertraining* [148]: it encourages the weight on each feature to be non-zero, in case this particular feature survives the corruption.

The resulting redundancy is one of the key factors why MCF corruption makes classifiers so effective against the “nightmare at test-time” scenario. Consequently, MCF classifiers are particularly useful in learning settings in which features in the test data may be missing (*e.g.*, because sensors that were measuring these features temporarily broke down). Learning with MCF is quite different from previous approaches for this setting [54, 67]. In particular, prior work focuses on the worst-case scenario whereas we explicitly consider the (arguably) more common average-case scenario by considering all possible corrupted observations. This has the advantage that it is computationally much cheaper and that it allows for incorporating prior knowledge in the learning. For instance, if the data is generated by a collection of unreliable sensors, knowledge on the reliability of a sensor may be used to set the corresponding q_d -parameter.

An interesting direction for future work is to investigate extensions of MCF to structured prediction, as well as to investigate if MCF can be employed for kernel machines. We also plan to explore in more detail what corruption models $p(\tilde{\mathbf{x}}|\mathbf{x})$ are useful for what types of data. Further, MCF could be used in the training of neural networks with a single layer of hidden units: blankout noise *on the hidden nodes* can improve the performance of the networks [96, 79] and can be marginalized out analytically. A final interesting direction would be to investigate the effect of marginalizing *corrupted labels* or target values [95].

3.3 Marginalized Corrupted Labels

In this work, we focus on image annotation using partial tags. Automatic image annotation is a difficult and highly relevant machine learning task. Recent advances have significantly improved the state-of-the-art in retrieval accuracy with algorithms based on nearest neighbor

classification in carefully learned metric spaces. But this comes at a price of increased computational complexity during training and *testing*. We propose FastTag, a novel algorithm that achieves comparable results with two simple linear mappings that are co-regularized in a joint convex loss function. The loss function can be efficiently optimized in closed form updates, which allows us we to incorporate a variety of image descriptors and scale to datasets with large number of images cheaply. On several standard real-world benchmark data sets, we demonstrate that FastTag matches the current state-of-the-art in tagging quality, yet reduces the training and testing times by several orders of magnitude.

The section is organized as follows. Section 3.3.3 reviews several image annotation techniques, in particular, the metric learning for local nearest neighbor based tag transfer approach. Section 3.3.4 details our FastTag approach to image annotation, which exploits the multi-modality of the data: images with keywords. We compare FastTag to the state-of-the-art on three standard image annotation datasets in section 3.3.5.

3.3.1 Recap of Learning with Partial Supervision

Obtaining the complete list of labels is often much more expensive than getting a partial list for many tasks. For example, in document classification tasks, finding all the categories that apply to the document in a taxonomy of thousands of nodes is much harder than pinpointing a single one. In image annotation, it is easy to tag an image with a few of the most prominent visual features, but to obtain the complete list can be quite difficult. Lately, the research community has been turning to crowdsourcing as a tool for obtaining labels cheaply and at scale. However, crowd sourced labels are noisy and notoriously incomplete.

Figure 3.11 shows a Venn diagram that illustrates the setup we have in learning with partial supervision. Let $\mathcal{X} \in \mathcal{R}^d$ denote the feature space, and $\mathcal{Y} \in \{+1, -1\}^T$ the output space. Let \mathcal{D} denote a joint distribution on $\mathcal{X} \times \mathcal{Y}$. It is the testing distribution we are ultimately interested in. Let \mathcal{D}' denote the distribution of the partially labeled data. As shown in the figure, the two marginal distributions match. But since \mathcal{D}' only contains part of the labels, the joint distributions differ.

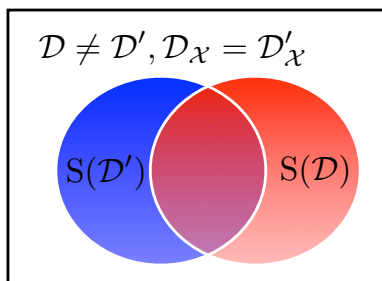


Figure 3.11: Venn diagram on learning with partial supervision.

Notation. Let $\mathcal{T} = \{\omega_1, \dots, \omega_T\}$ denote the dictionary of T possible annotation tags. We have access to a set of training data $L = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\} \subset \mathcal{R}^d \times \{0, 1\}^T$ sampled i.i.d from \mathcal{D}' , where \mathbf{y}_i contains the incomplete tags for instance \mathbf{x}_i . In other words, if the t^{th} entry of \mathbf{y}_i is positive, then the tag ω_t applies to example \mathbf{x}_i , but not necessarily the other way around. Our goal is to find a hypothesis $h \in \mathcal{H}$ to accurately predict the complete list of tags using the incompletely labeled set L .

3.3.2 Introduction

Image tag annotations are an important component of searchable image databases such as FlickrTM, PicassaTM or FacebookTM. However, a large fraction (over 50% in Flickr) of images have no tags at all and are hence never retrieved for text queries. Automatic image annotation is an essential tool towards surfacing this “dark content”. A working image annotation engine can suggest tags to users [164] and thus increase the number of tagged images, or generate relevant tags for image retrieval directly.

Automatic image annotation is a difficult machine learning task. Different type of objects require different image descriptors, *e.g.* rainbows can be identified through color histograms [76], whereas insects can be best identified through local image descriptors [102]. Similar objects can look very different across images and may only be partially visible, thus necessitating large training data sets. Training labels are typically obtained through crowd-sourcing and are noisy and notoriously incomplete. The ESP game [162] proposes a solution

to improve label quality by incentivizing pairs of labelers to match their answers. This results in tag sets with high precision but with no guarantees for high recall: each image may be tagged with only a small subset of tags that describe the most obvious visual features. As shown in figure 3.12, the annotation does not necessarily include all the relevant tags, for example, “fence” and “hand” are missing in the annotation of the leftmost image of the first row.

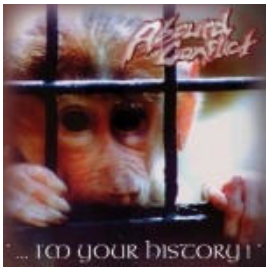
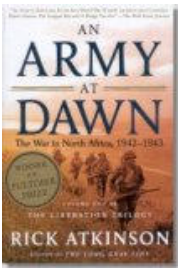






			
animal, ear	book, army	logo, green	cloud, sky, gray
			
people, pink, flower	white	film, smoke, women	face, man

Figure 3.12: Sample images with associated tags from the Espgame dataset.

Recently, Makadia et al. [104], Guillaumin et al. [75] proposed new algorithms for automatic image annotation based on nearest neighbor methods. Guillaumin et al. [75] carefully learn embeddings into metric spaces that combine a diverse set of image descriptors and assign tag-specific weights to overcome label sparsity. The resulting algorithm significantly improves over the prior state-of-the-art in both precision and recall. Although these approaches yield impressive results, they are impractical for large image databases with $n \gg 0$ images. Their training procedures scale on the order of $O(n^2)$. Moreover, the task of tagging a *single* test image is $O(n)$, *linear* with the training set size.

In real world applications, the number of images can be very large. Many million images are added every day (*e.g.* 300 Million images are uploaded to Facebook per day, with a total of 100 Billion images¹⁸), rendering these approaches impractical even to index the daily uploads.

We present a novel learning algorithm for image tag annotation that achieves comparable accuracy to Guillaumin et al. [75], but can be trained in $O(n)$ time and applied during testing in *constant time* w.r.t. the training set size. Our proposed algorithm, *FastTag*, can naturally incorporate many image descriptors and address the difficulties of label sparsity with a novel approach. It interprets its training data (images with partial tags) as *unlabeled multi-view* data and learns *two* classifiers to predict tag annotations: one attempts to reconstruct the (unknown) complete tag set from the few tags available during training; the other learns a mapping from image features to this reconstructed tag set. We propose a *joint* convex loss function that combines both classifiers via co-regularization and coerces them into agreement. Our loss function can be trained efficiently through alternating optimization with simple closed-form updates. We demonstrate on real world data sets that FastTag matches the highly competitive state-of-the-art in terms of precision and recall, but is several orders of magnitude faster during training and almost instantaneous during testing.

3.3.3 Related Work

In this section, we review some of the popular methods for automatic image annotation. We roughly divide them into four groups: parametric topic models, nonparametric mixture models, discriminative models and local nearest neighborhood based models.

Parametric Topic Models

The first group of models are based on topic models. Monay and Gatica-Perez [112] extend the probabilistic latent semantic analysis model [80], and Barnard et al. [7] extend the latent dirichlet allocation model [18] to the multi-modal data respectively.

¹⁸CNET 08/2012, <http://tinyurl.com/9jfs7ut>

Different from the traditional topic models, special treatment is required to model the co-occurrence of image features and tags. It is often assumed that the latent space is shared by the two modalities. In other words, same mixture of topics is shared by both image features and text features. Each annotated image is modeled as a mixture of topics, and each topic has a distinct distribution over the visual features and the tags. Often the case, a multinomial distribution over the tag dictionary and Gaussian distributions over the visual features are employed. where each topic generates the corresponding image features and tags. The mixture of topics, the distribution of text and the visual features given the hidden topics are the parameters that needed to be inferred.

Duygulu et al. [57] introduced a machine translation based method to automatic image annotation. The idea is to treat annotation as a translation process that translates image regions into annotation vocabulary. It can also be understood as a topic model that has one hidden topic for every annotation tag.

Although topic models offers strong explanatory power, the predictive power of these models is limited by the number of topics that could be included in the modeling. The complex parameter estimation process often limits the number of topics to be within hundreds. Also since the number of parameters grows linearly with the number of topics, the models run into overfitting problem easily as more topics are used, and Bayesian parameter estimation or other form of regularization need to be enforced.

Nonparametric Mixture Models

The second group of methods models the joint distribution of the image features and the tags with mixture models. Examples of models in this class include Continuous-space Relevance Model [85], Cross-Media Relevance Model [94], and Multiple Bernoulli Relevance Models [60].

Carneiro et al. [29] model the distribution over the image features of the entire image using gaussian mixture models with a fixed number of mixture components per keyword. Other methods [85, 94, 60] model the distribution over image patches or segmented image regions. The distribution is approximated using kernel density estimation [128]. After the model is trained, the conditional probability of the keyword given the visual features is used to annotate new images.

Discriminative Models

In generative models, the parameters are estimated to maximize the likelihood of generating the training data, which is not necessarily good for predictive performance. The third groups of methods instead train discriminative models, such as SVM [48], ranking SVM [72] and boosting [78] to predict tags from image features.

Local Nearest Neighborhood Based Models

The methods we introduced above have achieved promising annotation results. However, their complex training processes limits the number of descriptors that can be incorporated. Recently proposed models such as the Joint Equal Contribution model of Makadia et al. [104] and the TagProp model of Guillaumin et al. [75] rely on local nearest neighborhoods and work surprisingly well despite their simplicity. JEC assign equal weight to different visual descriptors when computing the distances between data points, while TagProp carefully learns the weights for different visual descriptors to maximize the predictive performance. TagProp is the current state-of-the-art method for image annotation. Its success can be attributed to three elements: 1) the ability to incorporate a large number of different visual descriptors; 2) the model grows in capacity as training data increases, alleviating the effect of sparse training tags; 3) its special treatment of rare tags.

Although Tagprop achieves superior performance on several benchmark datasets, the $O(n^2)$ training and $O(n)$ test complexity, where n is the number of examples in the training set, hinder its applicability to large scale datasets. In this work, we introduce a new model that incorporates the three elements for successful annotation much more cheaply. Our model matches the performance of TagProp in term of annotation precision and recall, but is much faster to train and test.

Most existing models assume that a complete list of relevant tags for each image is available at training time. However, in practice, this is either impractical or impossible for a large training set. It is much easier to tag an image with a few of the most prominent visual features than to obtain the complete list from a tag dictionary. To alleviate the need for complete labeling, several existing approaches [61, 136, 145] resort to semi-supervised approaches to

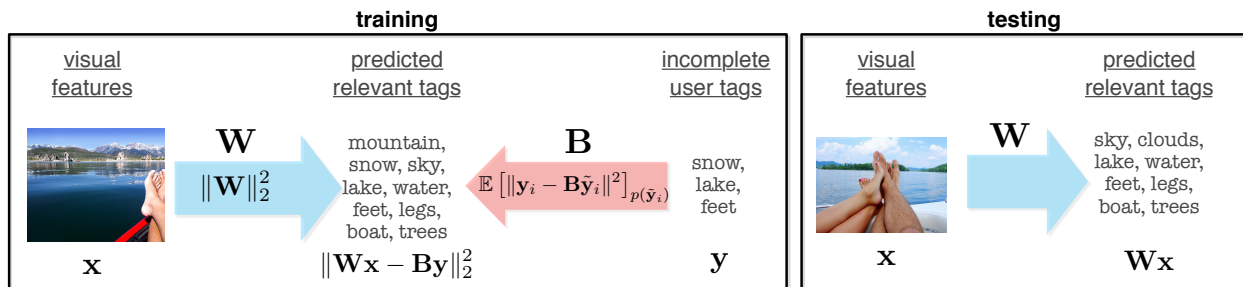


Figure 3.13: Schematic illustration of FastTag. During training two linear mappings \mathbf{B} and \mathbf{W} are learned and co-regularized to predict similar results. At testing time, a simple linear mapping $\mathbf{x} \rightarrow \mathbf{W}\mathbf{x}$ predicts tags from image features.

leverage unlabeled or weakly labeled data from the web. We adopt the same assumption of sparse training tags and incorporate partial supervision in our work.

3.3.4 Method

We assume that only a few relevant tags are provided for each image during training. Given the training images annotated with incomplete tags, our goal is to learn a model that can infer the full list of tags from image features at test time. Our proposed algorithm is fast in training and almost instant prediction during testing (only a linear transformation is required). Thus we refer to our algorithm as *FastTag*.

Notation. Let $\mathcal{T} = \{\omega_1, \dots, \omega_T\}$ denote the dictionary of T possible annotation tags. Let the training data be denoted by $D = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\} \subset \mathcal{R}^d \times \{0, 1\}^T$, where each vector $\mathbf{x}_i \in \mathcal{R}^d$ represents the features extracted from the i -th image and each \mathbf{y}_i is a small *partial* subset of tags that are appropriate for the i -th image. Our goal is to learn a linear function $\mathbf{W} : \mathcal{R}^d \rightarrow \mathcal{T}$, which maps a test image \mathbf{x}_i to its *complete* tag set.

Duo Classifier Formulation

In this section we introduce a new model for automatic image annotation from incomplete user tags. It jointly learns two classifiers on two sources, *i.e.*, image and text, to agree upon

the list of tags predicted for each image. It leads to an optimization problem which is jointly convex and has closed form solutions in each iteration of the optimization.

Co-regularized learning. As we are only provided with an incomplete set of tags, we create an additional auxiliary problem and obtain two sub-tasks: 1 training an image classifier $\mathbf{x}_i \rightarrow \mathbf{W}\mathbf{x}_i$ that predicts the complete tag set from image features, and 2 training a mapping $\mathbf{y}_i \rightarrow \mathbf{B}\mathbf{y}_i$ to *enrich* the existing sparse tag vector \mathbf{y}_i by estimating which tags are likely to co-occur with those already in \mathbf{y}_i . We train both classifiers simultaneously and force their output to agree by minimizing

$$\frac{1}{n} \sum_{i=1}^n \|\mathbf{B}\mathbf{y}_i - \mathbf{W}\mathbf{x}_i\|^2. \quad (3.32)$$

Here, $\mathbf{B}\mathbf{y}_i$ is the enriched tag set for the i -th training image, and each row of \mathbf{W} contains the weights of a linear classifier that tries to predict the corresponding (enriched) tag based on image features.

The loss function as currently written has a trivial solution at $\mathbf{B} = \mathbf{0} = \mathbf{W}$, suggesting that the current formulation is underconstrained. We next describe additional regularizations on \mathbf{B} that guides the solution toward something more useful.

Marginalized blank-out regularization. We take inspiration from the idea of marginalized stacked denoising autoencoders [43] in formulating the tag enrichment mapping $\mathbf{B} : \{0, 1\}^T \rightarrow \mathcal{R}^T$. Our intention is to enrich the incomplete user tags by turning on relevant keywords that should have been tagged but were not. Imagine that the observed tags \mathbf{y} are randomly sampled from the complete set of tags: it is a “corrupted” version of the original set. We leverage this insight and train the enrichment mapping \mathbf{B} to reverse the corruption process. To this end, we construct a further corrupted version of the observed tags $\tilde{\mathbf{y}}$ and train \mathbf{B} to reconstruct \mathbf{y} from $\tilde{\mathbf{y}}$. If this secondary corruption mechanism matches the original corruption mechanism, then re-applying \mathbf{B} to \mathbf{y} would recover the likely original pristine tag set.

More formally, for each \mathbf{y} , a corrupted version $\tilde{\mathbf{y}}$ is created by randomly removing (*i.e.*, setting to zero) each entry in \mathbf{y} with some probability $p \geq 0$ and therefore, for each user tag

vector \mathbf{y} and dimensions t , $p(\tilde{y}_t = 0) = p$ and $p(\tilde{y}_t = y_t) = 1 - p$. We train \mathbf{B} to optimize

$$\mathbf{B} = \underset{\mathbf{B}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n \|\mathbf{y}_i - \mathbf{B}\tilde{\mathbf{y}}_i\|^2.$$

Here, each row of \mathbf{B} is an ordinary least squares regressor that predicts the presence of a tag given all existing tags in $\tilde{\mathbf{y}}$. To reduce variance in \mathbf{B} , we take repeated samples of $\tilde{\mathbf{y}}$. In the limit (with infinitely many corrupted versions of \mathbf{y}), the expected reconstruction error under the corrupting distribution can be expressed as

$$r(\mathbf{B}) = \frac{1}{n} \sum_{i=1}^n \mathbb{E} [\|\mathbf{y}_i - \mathbf{B}\tilde{\mathbf{y}}_i\|^2]_{p(\tilde{\mathbf{y}}_i|\mathbf{y})}. \quad (3.33)$$

Let us denote as $\mathbf{Y} \equiv [\mathbf{y}_1, \dots, \mathbf{y}_n]$ the matrix containing the partial labels for each image in each column. Define $\mathbf{P} \equiv \sum_{i=1}^n \mathbf{y}_i \mathbb{E}[\tilde{\mathbf{y}}_i]^\top$ and $\mathbf{Q} \equiv \sum_{i=1}^n \mathbb{E}[\tilde{\mathbf{y}}_i \tilde{\mathbf{y}}_i^\top]$, then we can rewrite the loss in (3.33) as

$$r(\mathbf{B}) = \frac{1}{n} \operatorname{trace}(\mathbf{B}\mathbf{Q}\mathbf{B}^\top - 2\mathbf{P}\mathbf{B}^\top + \mathbf{Y}\mathbf{Y}^\top) \quad (3.34)$$

For the uniform “blank-out” noise introduced above, we have the expected value of the corruptions $\mathbb{E}[\tilde{\mathbf{y}}]_{p(\tilde{\mathbf{y}}|\mathbf{y})} = (1-p)\mathbf{y}$, and the variance matrix $\mathbb{V}[\tilde{\mathbf{y}}]_{p(\tilde{\mathbf{y}}|\mathbf{y})} = p(1-p)\delta(\mathbf{y}\mathbf{y}^\top)$. Note that, as we corrupt each tag independently, the variance matrix has non-zeros entries only on the diagonal. Here $\delta(\cdot)$ stands for an operation that sets all the entries except the diagonal to zero. We can then compute the two matrices as

$$\begin{aligned} \mathbf{P} &= (1-p)\mathbf{Y}\mathbf{Y}^\top \\ \mathbf{Q} &= (1-p)^2\mathbf{Y}\mathbf{Y}^\top + p(1-p)\delta(\mathbf{Y}\mathbf{Y}^\top) \end{aligned} \quad (3.35)$$

Eq. (3.34) is the regularization function for \mathbf{B} .

Joint loss function. Combining the squared loss in Eq. (3.32) with the marginalized blank-out regularization term $r(\mathbf{B})$ in Eq. (3.34) and the standard ridge regression l_2 regularizer

for \mathbf{W} , the joint loss function can be written as

$$\begin{aligned} \ell(\mathbf{B}, \mathbf{W}; \mathbf{x}, \mathbf{y}) &= \underbrace{\frac{1}{n} \sum_{i=1}^n \|\mathbf{B}\mathbf{y}_i - \mathbf{W}\mathbf{x}_i\|^2}_{\text{Co-regularization}} + \lambda \|\mathbf{W}\|_2^2 \\ &+ \underbrace{\gamma^r(\mathbf{B})}_{\text{Marginalized blank-out}}. \end{aligned} \quad (3.36)$$

The first term enforces that the tags enriched through co-occurrence with existing labels agree with the tags predicted by the content of the image. A regularizer on \mathbf{W} is included to reduce complexity and avoid overfitting. The last term ensures that the enrichment mapping \mathbf{B} reliably predicts tags if they were to be removed from the training label set.

Optimization and Extensions

The loss in Eq. (3.36) can be efficiently optimized using block-coordinate descent. When \mathbf{B} is fixed, the mapping \mathbf{W} reduces to standard ridge-regression and can be solved for in closed form:

$$\mathbf{W} = \mathbf{B}\mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + n\lambda I)^{-1}, \quad (3.37)$$

where \mathbf{X} and \mathbf{Y} respectively contain the training image features and labels in columns.

Similarly, when \mathbf{W} is fixed, the solution to Eq. (3.36) can be expressed as the well-known closed-form solution for ordinary least squares [43]:

$$\mathbf{B} = (\gamma\mathbf{P} + \mathbf{W}\mathbf{X}\mathbf{Y}^T)(\gamma\mathbf{Q} + \mathbf{Y}\mathbf{Y}^T)^{-1}.$$

where \mathbf{P} and \mathbf{Q} can be computed analytically following eq. (3.35). In other words, we can derive the optimal mapping \mathbf{B} under closed form without explicitly creating any corruptions. The conclusion holds for any corrupting models of which the expected value and variance can be computed analytically. The loss is jointly convex with respect to \mathbf{B} and \mathbf{W} and consequently coordinate descent converges to the global minimum. Fig. 3.13 contains a depiction of this algorithm.

Tag bootstrapping. The enrichment mapping \mathbf{B} is trained to predict missing tags based on pairwise co-occurrence patterns. We would like to also reconstruct tags that do not co-occur together but tend to appear within similar contexts. As an example, the tag “pond” might rarely co-occur with “lake”, as both describe similar things and annotators tend to use one or the other. However, it would be good to give the predictor \mathbf{W} the flexibility to predict both from similar image features. We can achieve this via stacking: starting with the enriched vector $\mathbf{B}\mathbf{y}_i$ as the tag representation for the i -th image¹⁹, we optimize another layer of $\ell(\mathbf{B}', \mathbf{W}'; \mathbf{x}, \mathbf{B}\mathbf{y})$ to obtain new mappings \mathbf{B}', \mathbf{W}' . We can have an arbitrary number of layers, each resulting in a new linear mapping \mathbf{W}^t from image features to tags. To find the right trade-off between too much bootstrapping and too little, we perform model selection on a hold-out set, adding layers until it no longer improves the F1 measure.

Rare tags. Eq. (3.36) solves for the linear predictors \mathbf{W} for all T tags simultaneously. This is computationally efficient in that it requires only one matrix inversion per iteration. However, it has the disadvantage that the prediction loss for each tag is weighed equally, which leads to the overall loss to be dominated by contributions from more frequent tags, sacrificing the prediction accuracy of rare tags. This is a known problem in tag prediction. Other approaches also find that dealing with rare tags is the key to improving tagging performance [75]. We introduce several re-optimization stages, where at each stage we solve a sub-problem of Eq. (3.36). That is, we identify a subset of tags with a recall below a certain threshold (in our experiments we choose the ones with zero recall). We re-optimize (3.36) restricted to only the rows of \mathbf{B} and \mathbf{W} corresponding to such tags. We iterate until we no longer improve the F1 measure on a hold-out set.

In addition, we weigh each example in a tf-idf-like fashion so that losses from rare tags are given more weight during training. Specifically, each tag ω is assigned a cost $c_\omega = \frac{1}{n_\omega}$, where n_ω is the number of times tag ω appears in the training set. Thus, rarer tags are given a higher cost than the more frequent ones. We then assign each example an weight by simply summing over the costs of its active tags, so that examples with rarer tags contribute more to the loss in eq. (3.36). Let Λ denote an $n \times n$ diagonal matrix containing the weight for each training example, we can then solve for the optimal mapping as $\mathbf{W} = \mathbf{B}\mathbf{Y}\Lambda\mathbf{X}^T(\mathbf{X}\Lambda\mathbf{X}^T + n\lambda I)^{-1}$. The tag enrichment mapping \mathbf{B} can be generalized to the weighted version in the same fashion.

¹⁹The enriched tags $\mathbf{B}\mathbf{y}_i$ are real numbers. When stacking, we truncate $\mathbf{B}\mathbf{y}_i$ to be within $[0, 1]^T$.

High F-1 score							
	bug, green, insect, tree, wood	blue, cloud, ocean, sky, water	black, computer, drawing handle, screen	baby, doll, dress, green, hair	blue, earth, globe, map, world	fish, fishing, fly, hook, orange	fly, plane, red, sky, train
Random							
	asian, boy, gun, man, white	anime, comic, people, red, woman	feet, flower, fur, red, shoes	blue, chart, diagram internet, table	gray, sky, stone, water, white	black, dark, game, man, night	plane, red, sky, train, truck
Low F-1 score							
	brown, ear, painting, woman, yellow	board, lake, man wave, white	blue, circle, feet round, white	drawing, hat, people red, woman	blue, dot, feet, microphone, statue	hair, ice, man, white, woman	black, moon, red, shadow, woman

Figure 3.14: Predicted keywords using FastTag for sample images in the Espgame dataset.

Test time. At test time, given an image \mathbf{x} , the final mapping \mathbf{W}^* is used to score the dictionary of tags.

Homogeneous feature mapping

Local kNN methods [75, 104] enjoy the advantage of naturally identifying non-linear decision boundaries based on multiple feature spaces from different image features. In our work, we adopt linear image feature classifiers for their simplicity and speed, and instead incorporate non-linearity into the feature space as a pre-processing step. To this end, we adopt the homogeneous feature mapping method of Vedaldi and Zisserman [157]. For each visual descriptor $\mathbf{f}_m(\mathbf{x}) \in \mathcal{R}^{d_m}$ extracted from the input image, it uses an explicit feature mapping $\Psi_m : \mathcal{R}^{d_m} \rightarrow \mathcal{R}^{d_m(2r+1)}$ to project it to a slightly higher-dimensional feature space, in which the inner product approximates the kernel distance well. In other words, $\langle \Psi_m(\mathbf{f}_m(\mathbf{x})), \Psi_m(\mathbf{f}_m(\mathbf{x}')) \rangle \approx K_m(\mathbf{f}_m(\mathbf{x}), \mathbf{f}_m(\mathbf{x}'))$. For the family of additive kernels, such as the l_1 -distance and χ^2 -distance used in our experiments, the mapping $\Psi(\cdot)$ can be computed analytically and approximates the kernel well even with small r (in our experiment, we set

$r = 1$). After projecting each visual descriptor independently, we further apply random projection [158] to reduce the dimensionality²⁰.

3.3.5 Experimental Results

We evaluate FastTag on three image annotation benchmark datasets. All data sets (with pre-extracted features) were obtained from <http://lear.inrialpes.fr/people/guillaumin/data.php>.

Experimental Setup

We begin with a detailed description of the data sets, the visual feature descriptors and the evaluation metrics.

Corel5K. The dataset was first introduced in [57], and has thereafter become a staple benchmark set for evaluating keyword based image retrieval and image annotation. It contains 5,000 images collected from the larger Corel CD set. Each image is manually annotated with keywords from a dictionary of 260 distinct terms. On average, each image was annotated with 3.5 tags.

ESP game. The dataset consists of 20,770 images²¹ of a wide variety, such as logos, drawings, and personal photos, collected for the ESP collaborative image labeling task [162]. Overall, the images are annotated with a total of 268 tags. Each image is associated with a maximum of 15 and on average 4.6 tags.

IAPRTC-12.²² The dataset consists of 19,627 images of sports, actions, people, animals, cities, landscapes and many other aspects of contemporary life [74]. Tags are extracted from the free-flowing text captions accompanying each image. Overall, 291 tags are used.

²⁰The dimension k is roughly cross-validated using a least squares baseline.

²¹To be comparable to prior work, we use the same subset (out of the total 60,000 images) selected by Guillaumin et al. [75], Makadia et al. [104], available at <http://hunch.net/?p=23>.

²²We used the same annotations as in [75, 104]

Table 3.5: Comparison of FastTag and existing work on the Corel5K dataset.

Name	P	R	F1	N+
leastSquares	29	32	30	125
CRM [94]	16	19	17	107
InfNet [109]	17	24	20	112
NPDE [169]	18	21	19	114
SML [29]	23	29	26	137
MBRM [60]	24	25	24	122
TGLM [100]	25	29	27	131
JEC [104]	27	32	29	139
TagProp [75]	33	42	37	160
FastTag	32	43	37	166

For all these datasets, we follow the training/test split used in previous work [75, 104]. Please refer to Guillaumin et al. [75] for more detailed statistics on the datasets.

Feature extraction. We use the 15 different visual descriptors, extracted by Guillaumin et al. [75] for each dataset. These include one Gist descriptor [119], six global color histograms, and eight local bag-of-visual-words features. As described in section 3.3.4, we adopt the explicit feature mapping of Vedaldi and Zisserman [157] to obtain a non-linear feature transformation. Here we use the l_1 approximation (*i.e.* the Euclidean distance after the mapping approximates the l_1 distance) for the global color descriptors, and the approximated χ^2 distance for the local bag-of-visual-words features. Finally, we apply random projection after each feature mapping to reduce the dimensionality.

Evaluation metric. For full comparability, we adopt the same evaluation metrics as in Guillaumin et al. [75]. First, all image are annotated with the five most relevant tags (*i.e.* tags that have the highest prediction value). Second, precision (P) and recall (R) are computed for each tag. The reported measurements are averaged across all tags. For easier comparability, both factors are combined in the F1-score ($F1 = 2 \frac{P*Q}{P+Q}$), which is reported separately. We also report the number of keywords with non-zero recall value (N+). In all metrics a higher value indicates better performance.

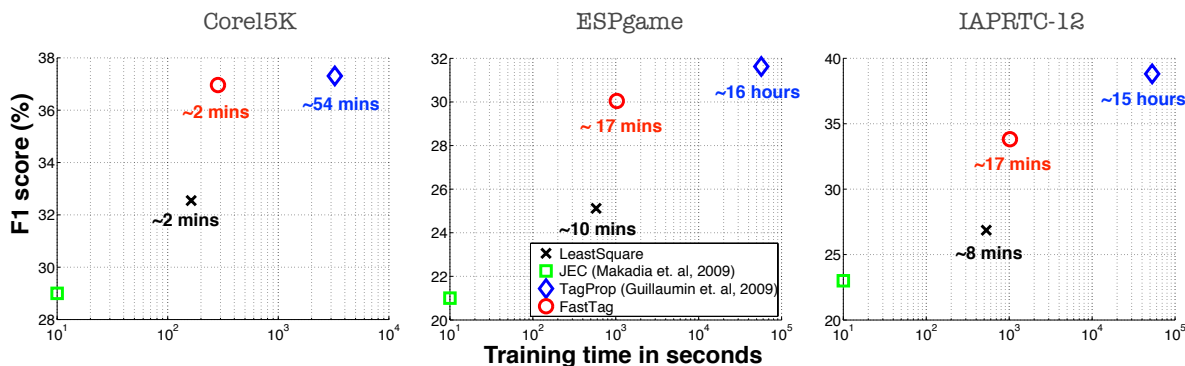


Figure 3.15: Comparison of FastTag and existing work in terms of F1 score vs. training time on the Corel5K, Espgame and IAPRTC-12 dataset.

Table 3.6: Comparison of FastTag and existing work on the Espgame and IAPRTC-12 dataset.

	Espgame				IAPR			
	P	R	F1	N+	P	R	F1	N+
leastSquares	35	19	25	215	40	19	26	198
MBRM	18	19	18	209	24	23	23	223
JEC	24	19	21	222	29	19	23	211
TagProp	39	27	32	238	45	34	39	260
FastTag	46	22	30	247	47	26	34	280

Baselines. We compare against *leastSquares*, a ridge regression model which uses the partial subset of tags $\mathbf{y}_1, \dots, \mathbf{y}_n$ as labels to learn \mathbf{W} , i.e., FastTag without tag enrichment. We also compare against the *TagProp* algorithm [75], a local kNN method combining different distance metrics through metric learning. It is the current best performer on these benchmark sets. Most existing work do not provide publicly available implementations. As a result, we include their previously reported results for reference [94, 109, 169, 29, 60, 100, 104].

Comparison with related work

Table 3.5 shows a detailed comparison of FastTag to the leastSquares baseline and eight published results on the Corel5K dataset. We can make three observations: 1. The performance of FastTag aligns with that of TagProp (so far the best algorithm in terms of accuracy on

this dataset), and significantly outperforms the other methods; 2. The leastSquares baseline, which corresponds to FastTag without the tag enricher, performs surprisingly well compared to existing approaches, which suggests the advantage of a simple model that can extend to a large number of visual descriptor, as opposed to a complex model that can afford fewer descriptors. One may instead more cheaply glean the benefits of a complex model via non-linear transformation of the features. 3. The duo classifier formulation of FastTag, which adds the tag enricher, alleviates the intrinsic label sparsity problem of image annotation. It leads to a 10% improvement on precision, 28% on recall, and an overall 20% improvement on F1 score over the leastSquares baseline. We also increase the number of tags with positive recall by 34.

Table 3.6 compares the performance of FastTag over leastSquares and three existing methods on the Espgame and IAPRTC-12 datasets. Similar trends can be observed. First, FastTag significantly outperforms the baseline, MBRM (a generative mixture model) of Feng et al. [60], and JEC (a local NN method) of Makadia et al. [104] on both datasets. FastTag performs slightly worse than TagProp on these datasets. However, as we next demonstrate, FastTag achieves enormous speedup over TagProp during both training and testing.

Computational time. All experiments were conducted on a desktop with dual 6-core Intel i7 cpus with 2.66Ghz.

Figure 3.15 shows the F1 score vs. the training time required for different methods on these three datasets. The time is plotted in log scale. We can make three observations: 1. TagProp outperforms all other related work in terms of F1 measure, but is also the slowest to train. It takes close to one hour to train on the relatively small Corel5K dataset, which has around 4,500 training examples. For the larger datasets (ESPgame and IAPRTC-12) with close to 17,000 examples, the training time blows up to 16 hours. 2. The JEC method of [104] falls into the same category of local NN method as TagProp, with the difference that it uses the simple average of the 15 distance metrics to define neighbors. JEC does not require training. However, we can see that it cannot compete in terms of accuracy performance. Note that, it still has $O(n)$ test-time complexity, where n is the number of training examples, because each query example requires a neighbor-lookup during testing. 3. The training time of FastTag is over 50x faster than that of TagProp. Note the time reported in the figure for FastTag also includes the feature preprocessing time, *i.e.*, performing homogeneous feature mapping

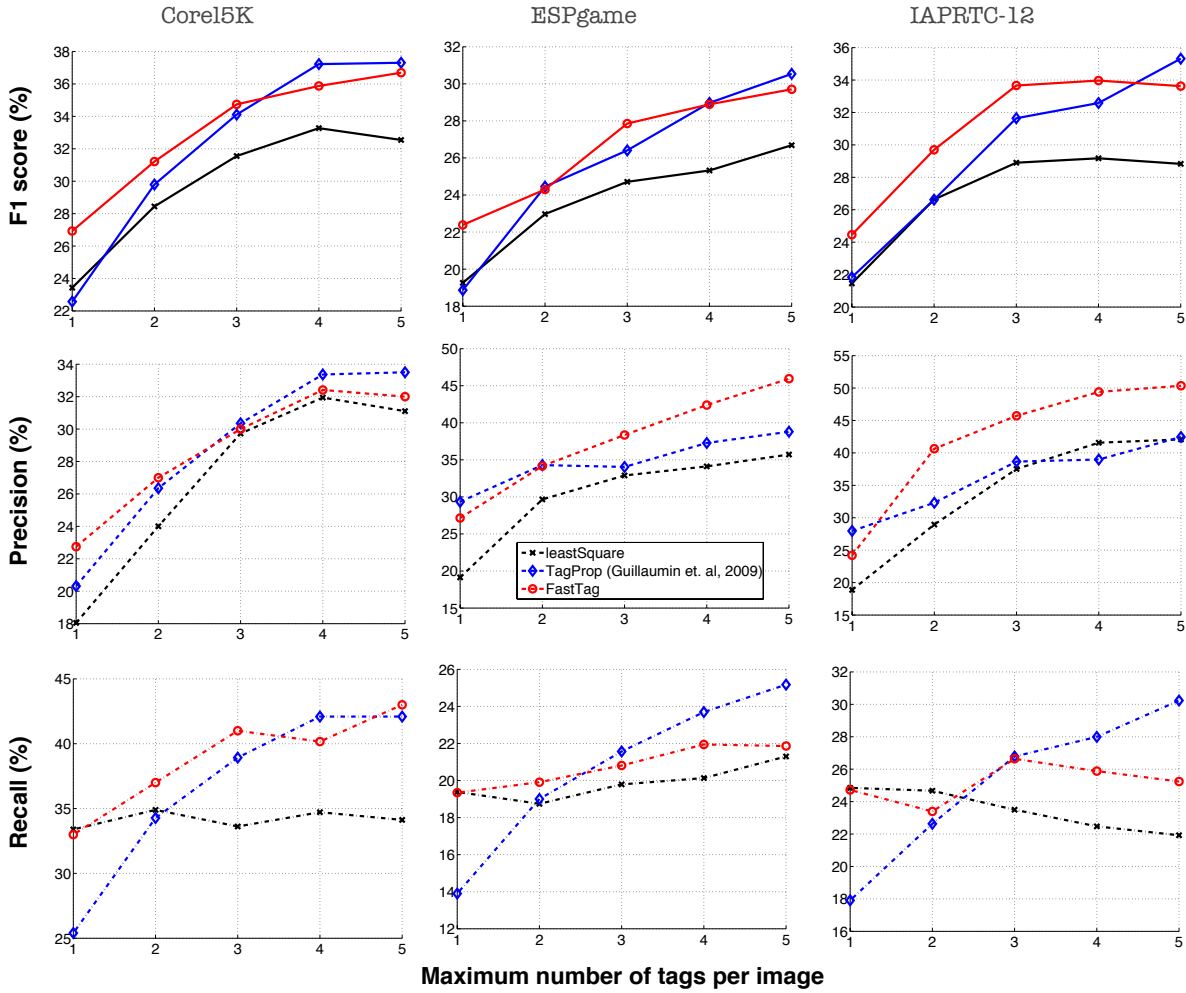


Figure 3.16: Comparison of FastTag and existing work at different levels of tag sparsity.

and random projection, which takes up the majority of the computation time. For a total of 16,748 training examples (dimensionality $d = 15,000$) and 268 tags, FastTag takes on average 34 seconds to train for one bootstrap iteration. The optimal number of bootstrap iterations ranges from 1 to 8 in different re-optimization iterations (The number of iterations is usually very small at the beginning, but gradually increases in the later re-optimization stages as it needs bootstrapping to recover rare tags.). The algorithm converges within a few re-optimization stages.

Sample annotations

Figure 3.14 shows example images from the ESP game data set and their tag annotations obtained with FastTag. The figure shows three rows of results. The top row consists of images with high $F1$ score, *i.e.* these are images on which FastTag reliably retrieves relevant tags. The middle row shows images that are picked uniformly at random. Although not perfect, the vast majority of tags are relevant to the particular image. The bottom images have low $F1$ score, and represent examples where FastTag fails to retrieve relevant tags. Note that, even among the ones with low $F1$ score, we can find some very relevant tags.

Further analysis

The graphs compare the results of FastTag with the TagProp algorithm at different levels of tag sparsity.

While these benchmark data sets are appropriate for algorithm comparisons, they may not be representative of the quality of training image tags found in the wild. In practice, most of the images are annotated with far fewer tags. We run the algorithms on images with down-sampled sparse tags in order to gauge their performance in this more realistic setting. Figure 3.16 depicts the comparison of FastTag and TagProp at different levels of training set tag sparsity. We “stage” the training data into successively larger tag sets, starting by giving each image only one tag (down sampled from the full set if more tags are available), then up to two tags, and so on. Performance in terms of Precision (P), Recall (R) and F1 score (F1) is plotted as a function of the maximum number of tags provided for each training image on the three benchmark datasets. We can see that FastTag out-performs TagProp when the maximum number of provided tags is small. In general, FastTag performs comparably to Tagprop across different tag sparsity levels. In other words, the tag enrichment mapping of FastTag indeed helps to alleviate the intrinsic tag sparsity problem.

3.3.6 Conclusion

We present an automatic image tagging method, FastTag, that performs on-par with current state-of-the-art algorithms, at a fraction of the computation cost. We recast a supervised multi-label classification problem as unlabeled multi-view learning. We define two classifiers, one for each view of the data, and coerce them into agreement via co-regularization in a joint loss function. We trade off complexity in the classifiers with non-linear mapping of the features and demonstrate that such a choice pays off. FastTag is computationally efficient during training and testing yet maintains tagging accuracy. It can effectively deal with sparsely tagged training data and rare tags that are often obstacles in such large-scale learning problems. We hope our research will lead to interesting future work in many related learning scenarios.

Chapter 4

Conclusion and Future Directions

Most canonical machine learning algorithms assume that: 1) a sufficiently large quantity of training data is available; 2) the training and testing data come from some common distribution. Although these assumptions are often met in practice, there are also many tasks for which obtaining a large amount of high quality labels (training data) is hard and the first assumption is violated. In this dissertation, we focus on the scenarios in which training data from the relevant distribution is scarce and exert trading off the second assumption to compensate for the first one. In other words, we utilize additional data, which is readily available or can be obtained easily but comes from a different distribution than the testing data. Generalizing to the testing distribution is hopeless if the data we used for training follows any arbitrary distribution. We have to enforce certain constraints on the distribution of the external data to enable learning.

We present five learning scenarios based on the kinds of constraints we imposed on the distribution that is used to sample the additional training data and the testing distribution: 1) learning with weak supervision; 2) domain adaptation; 3) learning from multiple domains; 4) learning from corrupted data; 5) learning with partial supervision. For learning with weak supervision, we assume the support of the data distribution is a superset of the support of the testing distribution. We proposed *Pseudo Multi-view Co-training* (PMC), an algorithm to cherry pick the instances that belong to the overlapped support from the noisy set to help learning. For domain adaptation and learning from multiple domains, we assume the support of the data (*source*) distribution(s) overlaps with that of the testing (*target*) distribution. we proposed *Co-training for Domain Adaptation* (CODA), an algorithm to slowly adapt the training data from source to target; and *marginalized Stacked Denoising Autoencoders*

(mSDA), an algorithm to reduce the difference between distributions of different domains via unsupervised feature learning. For learning from corrupted data, we introduced *Marginalized Corrupted Features* (MCF), an algorithm that learns more robust classifiers implicitly on “infinitely many” data created by corrupting the original small set of training data. For learning with partial supervision, we assume that only a subset of the labels is available for multi-label tasks. We proposed FastTag, an automatic image annotation algorithm that recovers missing labels by corrupting the labels. We demonstrated the applicability of these algorithms on many real datasets.

The proposed algorithms can be divided into two groups based on the basic building blocks.

Single View Co-training. The first two algorithms, PMC and CODA, are inspired by the co-training algorithms of Blum and Mitchell [23]. While Co-training itself is restricted to multi-view data, the automatic feature decomposition proposed in PMC generalizes co-training to the more common single view cases, making it applicable to most real datasets. In contrast to previous works [116, 171, 2, 33], which decompose the feature space either in a preprocessing step or greedily, PMC explicitly models the necessary conditions for successful co-training. It decomposes the feature space along with training the classifiers to satisfy these condition and the decomposition is adapted to the distribution of the training data in each PMC iteration, making it more powerful and reliable. PMC achieves record performance on the challenging Caltech-256 object recognition task, using web retrieved images as weakly labeled data. Comparable performance has been achieved before [65, 146], but much more sophisticated features and classifiers have to be employed. The ability of PMC to effectively select high quality examples from large collection of noisy search results opens the door to future work on diverse sets of web-specific applications across different domains, such as web-spam filtering, sentiment analysis or information retrieval.

CODA adds to PMC another feature selection component. While PMC shifts the distribution of the training data from source to target, the new feature selection component slowly shifts the active feature distribution from source to target, which address the problem of incompatible features used in different domains. In contrast to previous works [82, 105, 167, 21, 22], which either weighs the source instances, or learns a new feature representation as a preprocessing step, CODA changes the weights for source and target instances in each iteration, as

well as the features employed in the classifiers for best performance on the target domain. CODA achieves state-of-the-art classification results with impressive consistency across a wide range of available target labels. As CODA does not make any explicit assumption on the features or classifiers, it opens up opportunities for future work on domain adaptation in many different areas, such as natural language processing, speech and vision.

Marginalized Dropout. The second group of algorithms, mSDA, MCF and FastTag, all incorporate marginalized dropout into learning. Another common point of these three algorithms is that they all achieve state-of-the-art performance with orders of magnitude speedup at training and testing. mSDA is inspired by the Stacked Denoising Autoencoders (SDA) [161, 69], which learn hidden representation to reconstruct clean inputs from corruption. In contrast to SDA, mSDA: 1) does not have hidden nodes in each layer – this allows a closed-form solution with substantial speed-ups but might entail limitations; 2) only has two free meta-parameters, which greatly simplifies the model selection. 3) trains the parameters in each layer to optimally denoise all possible corrupted training inputs – arguably *infinitely many* by leveraging the analytic tractability of linear regression and marginalizing out corruption. This is practically infeasible for SDAs. mSDA achieves state-of-the-art performance. The fast training time, the capability to scale to large and high-dimensional data and implementation simplicity make it a promising method with appeal to a large audience within and beyond machine learning. As for its limitation, we found that sparsity in the original inputs is important for successful application of mSDA. For text data, it is natural to start from the sparse bag-of-word representation or its variant. For dense input, such as vision data, it would help to first transform the original input into some sparse representation, *e.g.*, bag-of-visual-word before applying mSDA. In future work, we would like to investigate whether we can relax this requirement while still preserving the computational efficiency.

MCF deals with overfitting when training data is insufficient. It offers an effective and efficient alternative to regularization or learning with priors. Specifically, MCF trains predictors by introducing corruption on the training examples, which is marginalized out in the expectation of the loss function. In comparison with traditional regularization approaches, MCF corruption shines in three ways: 1) it often yields superior classification performance and 2) it can be much more intuitive to set parameters about data corruption than about

model hyper parameters; 3) while regularization encourages sparsity, MCF with blank-out corruption appears to prevent weight under-training [158], encouraging the weight on each feature to be non-zero, in case this particular feature survives the corruption. The resulting redundancy is one of the key factors why MCF corruption makes classifiers so effective against the nightmare at test-time scenario. In comparison with previous work on incorporating corruption for better generalization [28, 35, 67], which either uses explicit corruption or focus on worst-case scenarios, MCF has several advantages: 1) it considers the (arguably) more common average-case scenario by training on all possible corrupted observations, which results in more robust classifiers; 2) it is computationally much cheaper; we can derive plug in solutions for a wide range of loss functions and corrupting distributions by marginalizing out the corruption. The training and testing cost of MCF is equivalent to learning with regularizations; 3) it is easy to incorporate prior knowledge of the data variance by setting the parameters in the corrupting distributions. We test MCF on a variety of tasks, including text and image classifications, and the “nightmare at test-time” scenario, and achieve promising results. We investigated blank-out, Poisson and Gaussian corruption in our experiments. As future work, we plan to explore in more detail what corrupting distributions is useful for what types of data. Another interesting direction is to extend MCF to structured prediction, kernel machines, as well as different learning scenarios, such as the learning with weak supervision, or cross-domain generalization.

FastTag is an image tagging method we introduced to learn from overly-sparse supervision, *i.e.*, incomplete user tags. Different from mSDA and MCF, in which the corruption is applied to the inputs, FastTag corrupt the partial list of labels in order to recover missing labels. Similarly, we consider all possible corrupted labels and marginalize out the corruption. In comparison with previous work on automatic image annotation [112, 7, 57, 85, 94, 60, 48, 72, 104, 75], FastTag employes a much simpler model. In particular, it finds two mappings, one from the visual descriptors to the complete list of tags, and one from the partial tags to the complete ones using the corrupted labels, by coercing them into agreement. The jointly convex loss function of FastTag can be efficiently optimized with closed form updates – this allows for including a diverse set of visual descriptors as well as non-linear feature mappings. The tradeoff of model complexity with feature expressiveness pays off as FastTag performs on-par with current state-of-the-art algorithms, at a fraction of the computation cost. Sparsely tagged training data and rare tags, which are prevalent in large scale problems, are often obstacles for learning. We have demonstrate a way to deal with such partial supervision for

image annotation. As future works, we would like to explore the work in many related tasks, such as multi-label text classification, and biology-related problems, *e.g.*, protein and gene function classifications.

References

- [1] J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *Proceedings of the Conference on Learning Theory*, pages 263–274, 2008.
- [2] S. Abney. Bootstrapping. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 360–367, 2002.
- [3] D.M. Allen. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16:125–127, 1974.
- [4] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. *NIPS*, 19:41, 2007.
- [5] M.F. Balcan, A. Blum, and K. Yang. Co-training and expansion: Towards bridging theory and practice. *NIPS*, 17:89–96, 2004.
- [6] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.
- [7] Kobus Barnard, Pinar Duygulu, David Forsyth, Nando De Freitas, David M Blei, and Michael I Jordan. Matching words and pictures. *The Journal of Machine Learning Research*, 3:1107–1135, 2003.
- [8] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.
- [9] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, page 137. The MIT Press, 2007.
- [10] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and Jenn Wortman. A Theory of Learning from Different Domains. *Machine Learning*, 2009.
- [11] A. Bergamo and L. Torresani. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *Neural Information Processing Systems (NIPS)*, 2010.
- [12] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU Math Expression Compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

- [13] D.P. Bertsekas, W.W. Hager, and O.L. Mangasarian. *Nonlinear programming*. Athena Scientific Belmont, MA, 1999.
- [14] C. Bhattacharyya, K.S. Pannagadatta, and A.J. Smola. A second order cone programming formulation for classifying missing data. In *Advances in Neural Information Processing Systems*, pages 153–160, 2004.
- [15] S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, page 161. The MIT Press, 2007.
- [16] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [17] D.M. Blei and M.I. Jordan. Modeling annotated data. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 127–134. ACM, 2003.
- [18] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.
- [19] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics*, Prague, Czech Republic, 2007.
- [20] John Blitzer. *Domain Adaptation of Natural Language Processing Systems*. PhD thesis, University of Pennsylvania, 2007.
- [21] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia, 2006.
- [22] John Blitzer, Dean Foster, and Sham Kakade. Domain adaptation with coupled subspaces. In *Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, 2011.
- [23] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.
- [24] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. *Advances in neural information processing systems*, 2001.
- [25] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [26] Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004, 2003.

- [27] U. Brefeld and T. Scheffer. Co-EM support vector learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 16. ACM, 2004.
- [28] C.J.C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector machines. *Advances in Neural Information Processing Systems*, 9:375–381, 1997.
- [29] Gustavo Carneiro, Antoni B Chan, Pedro J Moreno, and Nuno Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(3):394–410, 2007.
- [30] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.
- [31] N. Cesa-Bianchi, S. Shalev-Shwartz, and O. Shamir. Online learning of noisy data with kernels. In *Proceedings of the Conference on Learning Theory*, pages 218–230, 2010.
- [32] N. Cesa-Bianchi, S. Shalev-Shwartz, and O. Shamir. Online learning of noisy data. *IEEE Transactions on Information Theory*, 57(12):7907–7931, 2011.
- [33] J. Chan, I. Koprinska, and J. Poon. Co-training with a single natural feature set applied to email classification. In *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 586–589, 2004.
- [34] C.C. Chang and C.J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
- [35] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik. Vicinal risk minimization. In *Advances in Neural Information Processing Systems*, pages 416–422, 2000.
- [36] O. Chapelle, B. Scholkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.
- [37] O. Chapelle, P. Shivaswamy, S. Vadrevu, K.Q. Weinberger, Y. Zhang, and B. Tseng. Boosted multi-task learning. *Machine Learning*, pages 1–25, 2010. ISSN 0885-6125. 10.1007/s10994-010-5231-6.
- [38] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. 2004.
- [39] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster kernels for semi-supervised learning. *Advances in neural information processing systems*, 15:585–592, 2002.
- [40] G. Chechik, G. Heitz, G. Elidan, P. Abbeel, and D. Koller. Max-margin classification of data with absent features. *Journal of Machine Learning Research*, 9(Jan):1–21, 2008.

- [41] M. Chen, K.Q. Weinberger, and J.C. Blitzer. Co-training for domain adaptation. In *Advances in Neural Information Processing Systems (NIPS 2011)*, 2011.
- [42] M. Chen, K.Q. Weinberger, and Y. Chen. Automatic Feature Decomposition for Single View Co-training. In *International Conference on Machine Learning*, 2011.
- [43] M. Chen, Z. Xu, K.Q. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 767–774, 2012.
- [44] A. Coates, H. Lee, and A.Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the International Conference on Artificial Intelligence & Statistics, JMLR W&CP 15*, pages 215–223, 2011.
- [45] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 189–196, 1999.
- [46] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [47] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.
- [48] Claudio Cusano, Gianluigi Ciocca, and Raimondo Schettini. Image annotation using svm. In *Electronic Imaging 2004*, pages 330–338. International Society for Optics and Photonics, 2003.
- [49] R. Datta, D. Joshi, J. Li, and J.Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (CSUR)*, 40(2):5, 2008.
- [50] H. Daume III, A. Kumar, and A. Saha. Co-regularization based semi-supervised domain adaptation. In *Neural Information Processing Systems*, volume 2010, 2010.
- [51] Hal Daume III. Frustratingly easy domain adaptation. In *Annual meeting-association for computational linguistics*, page 256, 2007.
- [52] Hal Daumé III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1):101–126, 2006.
- [53] Bengio Y. Dauphin Y., Glorot X. Large-Scale Learning of Embeddings with Reconstruction Sampling. In *ICML*, 2011.
- [54] O. Dekel and O. Shamir. Learning to classify with missing and corrupted features. In *Proceedings of the International Conference on Machine Learning*, pages 216–223, 2008.

- [55] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1): 31–71, 1997.
- [56] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley Interscience Inc., 2001.
- [57] P. Duygulu, K. Barnard, J. De Freitas, and D. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. *Computer Vision ECCV 2002*, pages 349–354, 2006.
- [58] Andreas Argyriou Theodoros Evgeniou and Massimiliano Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, page 41. MIT Press, 2007.
- [59] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.
- [60] SL Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–1002. IEEE, 2004.
- [61] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. 2009.
- [62] A. Flaxman, A. Kalai, and H. McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 385–394, 2005.
- [63] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, 1995.
- [64] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics, 2001.
- [65] Peter Gehler and Sebastian Nowozin. On feature combination for multiclass object classification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 221–228. IEEE, 2009.
- [66] R. Ghani. Combining labeled and unlabeled data for text classification with a large number of categories. In *Proceedings of the IEEE International Conference on Data Mining*, 2001.

- [67] Amir Globerson and Sam Roweis. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on Machine learning*, pages 353–360. ACM, 2006.
- [68] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the International Conference on Machine Learning*, pages 513–520, 2011.
- [69] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the Twenty-eight International Conference on Machine Learning, ICML, 2011*.
- [70] Sally Goldman and Yan Zhou. Enhancing supervised learning with unlabeled data. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 327–334, 2000.
- [71] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. 2005.
- [72] David Grangier and Samy Bengio. A discriminative kernel-based approach to rank images from text queries. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(8):1371–1384, 2008.
- [73] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.
- [74] M. Grubinger, P. Clough, H. Müller, and T. Deselaers. The iapr tc-12 benchmark: A new evaluation resource for visual information systems. In *International Workshop OntoImage*, pages 13–23, 2006.
- [75] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 309–316. Ieee, 2009.
- [76] J. Hafner, H.S. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(7):729–736, 1995.
- [77] R. Herbrich and T. Graepel. Invariant pattern recognition by semidefinite programming machines. In *Advances in Neural Information Processing Systems*, volume 16, page 33, 2004.
- [78] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall. Learning distance functions for image retrieval. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–570. IEEE, 2004.

- [79] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.
- [80] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.
- [81] J. Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.
- [82] J. Huang, A.J. Smola, A. Gretton, K. M. Borgwardt, and B. Scholkopf. Correcting Sample Selection Bias by Unlabeled Data. In *NIPS 19*, pages 601–608. MIT Press, 2007.
- [83] J. Huang, A.J. Smola, A. Gretton, K.M. Borgwardt, and B. Scholkopf. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601, 2007.
- [84] A. Hyvärinen, J. Hurri, and P.O. Hoyer. Independent component analysis. *Natural Image Statistics*, pages 151–175, 2001.
- [85] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 119–126. ACM, 2003.
- [86] J. Jiang. A literature survey on domain adaptation of statistical classifiers. URL: <http://sifaka.cs.uiuc.edu/jiang4/domainadaptation/survey>, 2008.
- [87] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P07-1034>.
- [88] T. Joachims. Transductive inference for text classification using support vector machines. In *Machine Learning International Workshop*, pages 200–209. Citeseer, 1999.
- [89] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.
- [90] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W Moore. Reinforcement learning: A survey. *arXiv preprint cs/9605103*, 1996.
- [91] K. Kavukcuoglu, M.A. Ranzato, R. Fergus, and Y. Le-Cun. Learning invariant features through topographic filter maps. In *CVPR 2009. IEEE Conference on*, pages 1605–1612. IEEE, 2009.

- [92] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.
- [93] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: one-sided selection. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 179–186. Citeseer, 1997.
- [94] V. Lavrenko, R. Manmatha, and J. Jeon. A model for learning the semantics of pictures. NIPS, 2003.
- [95] N.D. Lawrence and B. Schölkopf. Estimating a kernel Fisher discriminant in the presence of label noise. In *Proceedings of the International Conference in Machine Learning*, pages 306–313, 2001.
- [96] Y. LeCun, J.S. Denker, and S.A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pages 598–605, 1990.
- [97] Honglak Lee, Yan Largman, Peter Pham, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in neural information processing systems*, 22:1096–1104, 2009.
- [98] A. Levin, P. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. In *Proc. ICCV*, volume 2, pages 626–633. Citeseer, 2003.
- [99] Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. *Machine Learning*, 46(1):191–202, 2002.
- [100] J. Liu, M. Li, Q. Liu, H. Lu, and S. Ma. Image annotation via graph learning. *Pattern Recognition*, 42(2):218–228, 2009.
- [101] T. Lorenzo, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *European Conference on Computer Vision (ECCV)*, pages 776–789, September 2010.
- [102] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157. Ieee, 1999.
- [103] F. Maillet, D. Eck, G. Desjardins, and P. Lamere. Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 345–350, 2009.
- [104] A. Makadia, V. Pavlovic, and S. Kumar. A new baseline for image annotation. In *ECCV*, volume 8, pages 316–329, 2008.

- [105] T. Mansour, M. Mohri, and A. Rostamizadeh. Domain Adaptation with Multiple Sources. In *NIPS 21*, pages 1041–1048. MIT Press, 2009.
- [106] D. McClosky, E. Charniak, and M. Johnson. Reranking and self-training for parser adaptation. In *Proceedings of the 44th Association for Computational Linguistics*, pages 337–344, 2006.
- [107] Geoffrey McLachlan and David Peel. *Finite mixture models*, volume 299. Wiley-Interscience, 2000.
- [108] G. Mesnil, Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley, P. Vincent, A. Courville, and J. Bergstra. Unsupervised and transfer learning challenge: a deep learning approach. *JMLR: Workshop and Conference Proceedings*, 7:1–15, 2011.
- [109] D. Metzler and R. Manmatha. An inference network approach to image retrieval. *Image and video retrieval*, pages 2130–2131, 2004.
- [110] Tom M Mitchell. Generalization as search. *Artificial intelligence*, 18(2):203–226, 1982.
- [111] Tom M Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 1997.
- [112] Florent Monay and Daniel Gatica-Perez. Plsa-based image auto-annotation: constraining the latent space. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 348–351. ACM, 2004.
- [113] V. Nair and G.E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. 27th International Conference on Machine Learning*, 2010.
- [114] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.
- [115] A.Y. Ng and M.I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *NIPS*, 2:841–848, 2002.
- [116] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93. ACM, 2000.
- [117] K. Nigam, A.K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2):103–134, 2000.
- [118] G. Obozinski, B. Taskar, and M. Jordan. Multi-task feature selection. In *the workshop of structural Knowledge Transfer for Machine Learning in the 23rd International Conference on Machine Learning (ICML 2006)*. Citeseer, 2006.

- [119] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.
- [120] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.
- [121] J.C. Platt et al. Probabilities for sv machines. *Advances in Neural Information Processing Systems*, pages 61–74, 1999.
- [122] Elijah Polak. *Computational methods in optimization: a unified approach*, volume 77. Academic Pr, 1971.
- [123] Qian Liu and Aaron Mackey and David Roos and Fernando Pereira. Evigan: a hidden variable model for integrating gene evidence for eukaryotic gene prediction. *Bioinformatics*, 2008.
- [124] M. Ranzato and G.E. Hinton. Modeling pixel means and covariances using factorized third-order Boltzmann machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2551–2558, 2010.
- [125] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, MA, 2006.
- [126] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the Twenty-eight International Conference on Machine Learning, ICML*, 2011.
- [127] Jo Bo Rosen. The gradient projection method for nonlinear programming. part i. linear constraints. *Journal of the Society for Industrial & Applied Mathematics*, 8(1): 181–217, 1960.
- [128] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, pages 832–837, 1956.
- [129] A. Rostamizadeh, A. Agarwal, and P. Bartlett. Learning with missing features. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 635–642, 2011.
- [130] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.
- [131] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [132] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. *Computer Vision–ECCV 2010*, pages 213–226, 2010.

- [133] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.
- [134] S. Satpal and S. Sarawagi. Domain adaptation of conditional probability models via feature subsetting. *Knowledge Discovery in Databases: PKDD 2007*, pages 224–235, 2007.
- [135] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.
- [136] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [137] B. Settles. Active learning literature survey. *Machine Learning*, 15(2), 1994.
- [138] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.
- [139] F. Sha and F. Pereira. Shallow parsing with Conditional Random Fields. In *Proceedings of Human Language Technology – NAACL 2003*, pages 213–220, 2003.
- [140] G. Shafer and V. Vovk. A tutorial on conformal prediction. *The Journal of Machine Learning Research*, 9:371–421, 2008.
- [141] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.
- [142] P.K. Shivaswamy, C. Bhattacharyya, and A.J. Smola. Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7:1283–1314, 2006.
- [143] Vikas Sindhwani and S Sathiya Keerthi. Large scale semi-supervised linear svms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 477–484. ACM, 2006.
- [144] Alexander Smola and Risi Kondor. Kernels and regularization on graphs. *Learning theory and kernel machines*, pages 144–158, 2003.
- [145] R. Socher and L. Fei-Fei. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 966–973. IEEE, 2010.

- [146] Kihyuk Sohn, Dae Yon Jung, Honglak Lee, and Alfred O Hero. Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2643–2650. IEEE, 2011.
- [147] M. Sugiyama and K.R. Müller. Input-dependent estimation of generalization error under covariate shift. *Statistics & Decisions*, 23(4/2005):249–279, 2005.
- [148] C. Sutton, M. Sindelar, and A. McCallum. Feature bagging: Preventing weight undertraining in structured discriminative learning. Technical Report IR-402, University of Massachusetts, 2005.
- [149] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998.
- [150] Martin Szummer and Tommi Jaakkola. Information regularization with partially labeled data. In *Proc. NIPS*, 2002.
- [151] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
- [152] C.H. Teo, A. Globerson, S. Roweis, and A. Smola. Convex learning with invariances. *Advances in Neural Information Processing Systems*, 20:1489–1496, 2008.
- [153] A. Torralba, R. Fergus, and W.T. Freeman. 80 million tiny images: A large dataset for non-parametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.
- [154] T. Trafalis and R. Gilbert. Robust support vector machines for classification and computational issues. *Optimization Methods and Software*, 22(1):187–198, 2007.
- [155] L.J.P. van der Maaten, M. Chen, S. Tyree, and K.Q. Weinberger. Learning by marginalizing corrupted features. In *Proceedings of the International Conference on Machine Learning (to appear)*, 2013.
- [156] V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 2000.
- [157] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):480–492, 2012.
- [158] Santosh S Vempala. *The random projection method*, volume 65. Amer Mathematical Society, 2005.
- [159] P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML 25*, pages 1096–1103. ACM, 2008.

- [160] P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the International Conference on Machine Learning*, pages 1096–1103, 2008.
- [161] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.
- [162] L. Von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM, 2004.
- [163] Kilian Q Weinberger and Lawrence K Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–988. IEEE, 2004.
- [164] Kilian Q. Weinberger, Malcolm Slaney, and Roelof Van Zwol. Resolving tag ambiguity. In *Proceeding of the 16th ACM international conference on Multimedia*, MM '08, pages 111–120, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-303-7.
- [165] K.Q. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.
- [166] H. Xu, C. Caramanis, and S. Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10:1485–1510, 2009.
- [167] G. Xue, W. Dai, Q. Yang, and Y. Yu. Topic-bridged PLSA for cross-domain text classification. In *SIGIR*, 2008.
- [168] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.
- [169] A. Yavlinsky, E. Schofield, and S. Rüger. Automated image annotation using global features and robust nonparametric density estimation. *Image and video retrieval*, pages 593–593, 2005.
- [170] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114. ACM, 2004.
- [171] W. Zhang and Q. Zheng. TSFS: A Novel Algorithm for Single View Co-training. In *Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on*, volume 1, pages 492–496, 2009.

- [172] Yan Zhou and Sally Goldman. Democratic co-learning. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 594–602. IEEE, 2004.
- [173] J. Zhu, S. Rosset, H. Zou, and T. Hastie. Multi-class AdaBoost. Technical Report 430, Department of Statistics, University of Michigan, 2006.
- [174] X. Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2006.
- [175] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, volume 20, page 912, 2003.

Learning with Marginalized Dropout, Chen, Ph.D. 2013