Data pruning

Thesis by

Anelia Angelova

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science

California Institute of Technology

Pasadena, California

2004

(Submitted May 27, 2004)

Acknowledgements

I would like to thank my advisers Professor Pietro Perona and Professor Yaser Abu-Mostafa for their help and support. This work would not have been possible without them.

I would also like to thank my colleagues from the Vision Group and Learning Systems Group at Caltech.

This work is supported by the National Science Foundation Research Center for Neuromorphic Systems Engineering.

Finally, I thank Dragomir and my family for their love and support.

Abstract

Could a training example be detrimental to learning? Contrary to the common belief that more training data is needed for better generalization, we show that the learning algorithm might be better off when some training examples are discarded. In other words, the quality of the examples matters.

We explore a general approach to identify examples that are troublesome for learning with a given model and exclude them from the training set in order to achieve better generalization. We term this process 'data pruning'. The method is targeted as a pre-learning step in order to obtain better data to train on.

The approach consists in creating multiple semi-independent learners from the dataset each of which is influenced differently by individual examples. The multiple learners' opinions about which example is difficult are arbitrated by an inference mechanism. Although, without guarantees of optimality, data pruning is shown to decrease the generalization error in experiments on real-life data. It is not assumed that the data or the noise can be modeled or that additional training examples are available.

Data pruning is applied for obtaining visual category data with little supervision. In this setting the object data is contaminated with non-object examples. We show that a mechanism for pruning noisy datasets prior to learning can be very successful especially in the presence of large amount of contamination or when the algorithm is sensitive to noise.

Our experiments demonstrate that data pruning can be worth while even if the algorithm has regularization capabilities or mechanisms to cope with noise and has a potential to be a more refined method for regularization or model selection.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Learning and Generalization

A learning task can be defined in the following way: given a set of training examples $S = \{(x_1, y_1), ..., (x_N, y_N)\}$ which supposedly come from an unknown target function $f : X \rightarrow \{-1; +1\}$, $y_i = f(x_i)$, $i = 1 \ldots, N$ find a function g, $g : X \rightarrow \{-1; +1\}$, $g \in G = \{g\}$ which agrees with the target function f as well as possible. The learning model G is a predefined set of hypotheses from which the target function is selected.

The purpose of learning machines is to be able to classify correctly unseen examples, not only the training ones. In other words, to generalize well rather than simply memorize the training data. If the learning algorithm performs well on the training data but poorly on unseen data, overfitting is said to occur.

1.2 Difficult examples and outliers

In real-life problems it is possible that the training data is contaminated by noise, meaning $y_i = f(x_i)$ is not satisfied for all of the examples. Those wrongly labeled examples would be problematic for the learning process. An important task of a learning algorithm is to be robust to noisy or mislabeled examples.

Troublesome examples may also be ones which are difficult in a sense that in order to learn them the algorithm would be in contradiction with other training examples, or would need to increase its complexity in order to accommodate them. Learning these hard examples may lead to the algorithm being unable to generalize well or overfit.

In the presence of troublesome examples, the learning machine has to face a very difficult task: it has to learn an unknown function using a finite number of examples considering that some of the examples may be misleading or difficult in other ways.

We would like to explore if it is possible to find mechanisms to detect and eliminate examples hard for learning and improve the generalization performance by doing so.

1.3 Data pruning overview

1.3.1 What is data pruning?

Suppose we are given a set of training examples and a selected learning model. The task is to identify if there are examples in the training set such that by eliminating them one may improve generalization performance.

Many methods deal with noisy examples by querying for more data, accommodating the examples by reweighting or decreasing their influence by regularizing the solution. We will study methods that fully eliminate those 'bad' examples from training or, in other words, prune the training data.

The most challenging problem is how to define for a particular training data and selected model which examples are worth removing so that to improve generalization ability of the learner.

1.3.2 Overview of data pruning method

Our approach consists in learning diverse classifiers (learners) by randomizing the training set and then combining their output to decide on difficult examples. We use a bootstrapping method to select various semi-independent learners.

Each of the new learners would be capable of classifying the data comparably to the learner on the full-sized training set. However, each one is trained on a different subset of the training data in the hope that they will have independent or semi-independent opinions with respect to the troublesome examples. Our intuition is that most of the learners would agree on non-difficult examples. Furthermore, examples which are forcing poor generalization performance would not influence all the learners. So, combining the output of all the learners may help identify the troublesome training examples. The classifiers' opinions are combined using Bayesian reasoning to receive a final decision of whether an example has to be eliminated.

We apply the data pruning method to learn to recognize face category from very noisy datasets. As the target is visual category, multiple semi-independent learners can be received by training on slightly overlapping regions containing parts of the object.

Chapter 2

Identifying noisy examples

In this chapter we give a notion of which example is difficult and show various ways to deal with outliers, noisy or difficult examples in different areas of learning from examples.

2.1 Introduction

2.1.1 What is a difficult example?

Formal definition of difficult examples is hard to give. Below we give informal definition of outliers or difficult examples. Difficult examples are those which obstruct the learning process or mislead the learning algorithm or those which are impossible to reconcile with the rest of the examples.

Defining difficult example cannot be done without the learning model or the generating distribution.

The reliability of an observation is dependent on the other observations, i.e., defining difficult example should be done in the context of the remaining data. The relative number of outliers with the same 'weirdness' should be small, i.e., the particular properties of an outlier are not supported by many other examples.

Quite often in statistics the method advocated is through visual examination of the data or the residuals after fitting a particular model. Further elimination of values which are surprising to the investigator is done, if necessary. This makes the definition of a difficult example very subjective and prone to errors due to misspecified model.

As a summary, an outlier is discordant with the remaining data and with respect to the model.

We would use interchangeably the terms outliers, difficult, adversary, noisy and troublesome example. Alternative names common in the literature are discordant observations, contaminants, surprising values.

2.1.2 How to deal with outliers?

Defining what a difficult example is and how to cope with it are two interconnected problems.

Different approaches have been taken trying to be robust to outliers: discard them, give different weights to more influential examples, average out multiple observations so that to diminish the poor influence of outliers in some models or accommodate them in a redesigned model. Discarding examples is suitable in the case of inherently wrong observations. It can improve the accuracy of the mean. Reweighting is suggested for heavy tailed distributions, for example, give weights according to the standard deviation. Averaging out observations would reduce the variance.

In this work we will eliminate the outlying examples altogether.

2.1.3 Why defining outliers is difficult?

Outliers and difficult examples may come from various sources and may be realizations of different phenomena. So, assuming a particular model for the noise might not be always appropriate.

Difficult examples may be noisy e.g. coming from different than the assumed distribution, or may be the result of wrong measurements.

In statistics an outlier can be defined from the generating distribution. Given the distribution, an outlier is a value which deviates 'too much'. This, in the first place, is not a clear-cut definition because it is not known what deviations are tolerable. Moreover, even if a particular distribution is assumed, there is no access to the actual one.

An example may be a result of natural variation and the model should be able to handle those without discarding them as outliers.

Furthermore, the learning algorithm may react in different ways to the presence of outliers: some algorithms might be able to learn in the presence of noise, others would be more severely influenced by adversary examples, resulting in poorer decision boundary and worse generalization performance.

2.1.4 Motivation: Why study outliers?

The outliers themselves can be a main interest for practical purposes such as detection of anomalies, interesting observations, etc.

The other side of the story is that outliers may just be measurements which are wrong and subsequently would force the incorrect model parameters to be estimated or the function selected may not generalize well because of overfitting with respect to those difficult examples. In those cases the troublesome examples are again of interest but to be considered for elimination.

An example which is too influential in the model can change the estimation of the model. For example, outlying observations can result in a wrong estimate of the mean of the population.

Moreover, in data analysis, identifying and studying outliers provides important information as to how adequate the current model is and may suggest a revision of the model and its assumptions.

As we have discussed so far, defining a difficult example is quite subjective. A statistically objective method to identify and deal with outliers is still a topic of research. In the following sections we review several methods to identify and cope with troublesome examples in various machine learning settings.

2.2 Robust statistics

Robust statistics is preoccupied with how to identify outliers or noisy observations, eliminate their influence or fully discard them [5], [21]. We give some examples of how robust estimation can be done.

2.2.1 Accommodating outliers

One straightforward approach is to reweight the observations according to their influence or the confidence we have in them and re-estimate the model parameters.

Alternative one is to use more robust cost functions. For example, the least squares objective function $\sum_i (X_i - M)^2$, where M denotes the fitted model for linear regression, is notorious for being very sensitive to outliers. Huber [21] suggested to use other cost functions $\sum_i \rho(X_i - M)$ which are more robust to outlying observations, for example $\rho(t) = |t|$ [5].

Another approach is to modify the model so that it takes into consideration outlying observations.

2.2.2 Eliminating outliers

Various statistical tests have been created for particular cases of identifying one or two outlying observations. Examples are removed one at a time and the model is re-estimated. Students' tests are used to decide if an observation is deviating too much when not included in the estimation [43].

Robust estimation by examining residuals and removing examples can be done for linear models with normal distributed noise $\mathbf{Y} = \mathbf{X}\beta + \mathbf{e}$, $var(\mathbf{e}) = \sigma^2\mathbf{I}$, where $\mathbf{X}$ is a set of data points, $\mathbf{Y}$ are the responses and a linear model needs to be fit to those points so that to minimize the least-squares error of the fit [43]. β contains the parameters of the linear model and $\mathbf{e}$ is a vector of residual errors. The least-squares estimate $\hat{\mathbf{Y}}$ is computed as follows $\hat{\mathbf{Y}} = \mathbf{X}(\mathbf{X^T X})^{-1}\mathbf{X^T Y} = \hat{\mathbf{H}}\mathbf{Y}$ where $\hat{\mathbf{H}}$ is called the 'hat' matrix. Examples which have too large influence on the model can be removed. The influence of example X_i is determined by h_{ii} in the 'hat' matrix. Details are given below.

The residuals are defined as $\mathbf{e} = \mathbf{Y} - \hat{\mathbf{Y}} = [\mathbf{I} - \hat{\mathbf{H}}]\mathbf{Y}$. The residual of an example satisfies $var(e_i) = \sigma^2(1 - h_{ii})$, i.e., values of h_{ii} close to 0 mean that the example would have a large deviation of the error estimation and suggests that it might be an outlier. However, caution should be taken because this might not always be so: the h_{ii} value should be considered in the context of the other h_{ij} in the 'hat' matrix [43].

Another way to estimate the influence of examples is by perturbing the data and examining again the residuals. Examples which result in major changes of the model are considered influential [43].

Many estimators are created particularly to improve the robustness of the mean or other statistics of the data but we would not examine them here.

The practice in robust statistics is to identify outliers for further investigation. The actual elimination is done after human supervision. In this work we explore an approach which would automatically identify and eliminate outlying examples.

Generally, in statistics, it is recommended that the samples be 'routinely subjected' to outlier detection procedures before estimating the model. This diagnostic might help to validate the adequacy of the model, to guide the subsequent data analysis, to take into consideration examples which might need to be further investigated, rejected or accommodated by the model [5].

2.2.3 RANSAC

The RANSAC (RANdom SAmple Consensus) algorithm [16] is used to learn model parameters in the presence of large number of outliers. It randomly samples minimal subsets of the data to estimate the model parameters and selects the model with maximum agreement among the samples.

It is assumed that the target model is known and a fixed number of data points m (presumably small) are needed to determine the model parameters uniquely. The algorithm proceeds by selecting multiple times (T trials) a random subset of m data points, estimate the model parameters and rate the selected model correspondent to how much it complies with the rest of the data. Out of the many attempts to fit a model the one which fits the whole data best is selected. It relies on the fact that in data containing a large amount of outliers (50% or more) the model parameters selected using the outliers would be inconsistent with each other, while the correct model parameters would be consistent throughout many trials. So, in order to estimate them with high confidence a sufficient number of trials is needed.

RANSAC is often used in vision applications for example to fit geometric primitives e.g. a line or a circle to a set of noisy points, to find point correspondences, or to find a matrix for a transformation which best explains the evidence (usually very noisy). A trivial example of using RANSAC is as follows: if we need to find a circle which is consistent with a lot of noisy points, three points are sufficient to determine the circle center at each trial, the most frequent center of circles is selected as the best model.

The RANSAC algorithm requires the target model to be known and the number of examples to estimate it uniquely should be small. In our task of data pruning we are uncertain about both model complexity, and subset of data needed for estimating learning. There is a large degree of freedom introduced by the model and its complexity being unknown.

2.3 Regularization methods

Regularization methods are created to deal with problems which are ill-posed, namely, the solution may not exist, is not unique or is unstable to small perturbations in the initial data [39]. Generally, regularization methods apply penalty or restriction on the class of admissible solutions so that the problem becomes a well-posed one. There is a large body of work on regularization with applications in solving differential equations, inverse problems, linear integral equations, etc.

We would discuss only regularization methods which are connected to learning.

Suppose the problem consists in minimizing a functional $R(X)$ depending on the training data. Regularization methods suggest to minimize $R(X) + \lambda\Omega(X)$ instead, where $\Omega(X)$ is a stabilizing functional or penalty. It may express desired properties of the solution, for example, we can penalize for a function with large derivatives to prefer a smoother (less oscillatory) function. The new target function is a trade-off, controlled by the coefficient λ, between fitting the data and using too complex a function.

There are other forms of regularization. Below, we discuss some of them. Without trying to encompass all, we consider only cases which are relevant to learning and more specifically to dealing with noisy examples or outliers.

2.3.1 Regularization with penalties

The most popular form of regularization is the weight decay for Neural Networks, where the penalty is over the sum of squares of all parameters in the network, namely its weights [6]. The regularizing functional penalizes large sum of weights in the network which may lead to overly complex discrimination function: $\Omega = \frac{1}{2}\sum_i w_i^2$.

A heuristic with similar purposes is early stopping, in which smaller number of iterations of the learning process (epochs) is preferred. It has a regularization effect because more epochs are more likely to create more complex network and therefore overfit the data [6].

Theoretical justification for preferring smaller sum of weights with weight decay or early stopping in Neural Networks was given by Bartlett [3]. He showed that the generalization ability of the Neural Network depends on the sum of weights of the network. Thus networks with larger weights may be of large complexity, which would increase the generalization error.

2.3.2 Introducing slack variables

We demonstrate the method of using slack variables for regularization with the so called soft margin hyperplanes for Support Vector Machines (SVM) [36]. The essence of SVM classification for linearly separable data is to find a hyperplane $y = \langle \mathbf{w}, \mathbf{x} \rangle + b$ which achieves maximum margin measured as $\frac{1}{\|\mathbf{w}\|^2}$, for both classes. The following problem needs to be solved:

$$\begin{array}{rl} \min \ \frac{1}{2}\|\mathbf{w}\|^2 & \quad \text{subj.to} \\ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 & \quad i = 1, \ldots, N \end{array}$$

To cope with linear nonseparability, the kernel trick is applied and the dataset is transformed to high dimensional space [36]. Still, even in high dimensional space, the data may not be perfectly separable and some form of regularization is needed. Cortes and Vapnik [14] suggest to introduce slack variables which allow some data points to violate the decision boundary. This method has proved to be very successful in practice.

$$\begin{array}{rl} \min \ \frac{1}{2}\|\mathbf{w}\|^2 - C\sum_{i=1}^{N} \xi_i & \quad \text{subj.to} \\ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 & \quad i = 1, \ldots, N \end{array}$$

2.3.3 Case study: Regularizing AdaBoost

Boosting algorithms and in particular AdaBoost [18] combine weak learners, learners performing slightly better than random guessing, into a strong one. The crucial idea of boosting is to give larger weights to examples wrong with respect to the current weak learner. Thus examples which are generally difficult would get consistently larger weights.

Boosting methods are of particular interest to us, because their internal mechanism is to overemphasize difficult examples - a strategy opposite to outlier elimination or de-emphasizing. Yet, in noisy cases, this strategy of AdaBoost has shown in experiments to be suboptimal. Therefore some sort of regularization is needed.

Jiang [22] supported theoretically the claim by demonstrating that 'boosting forever' leads to suboptimal solutions and that a regularized version of boosting would give theoretically better prediction.

2.3.3.1 Applying penalties

The standard way of applying penalty to the cost function to be minimized can be straightforwardly applied to AdaBoost.

Mason et al. [29] presented an interpretation of boosting methods as a gradient descent over some cost function. By minimizing the cost function at each iteration, the gradient descent algorithm selects a best hypothesis and a coefficient, which is equivalent to modifying the weights over the examples in order to select the next best hypothesis in the original boosting algorithm.

AdaBoost algorithm can be exactly retrieved from this framework with the exponential cost function over the margin of an example, defined as $\gamma(\mathbf{x}_i) = y_i \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_i)$

$$G = \sum_{i=1}^{N} e^{-y_i \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_i)}$$

where $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$ is the training set of N examples and T is the number of iterations.

R*ä*tsch et al. [33],[34] proposed to regularize AdaBoost by modifying the cost function at each iteration through adding penalty for difficult examples. Their criterion for a difficult example is the average weight of an example of all the iterations up to the current one. The cost function to be minimized at iteration t_0 by AdaBoost becomes:

$$G^{t_0} = \sum_{i=1}^{N} e^{-y_i(\sum_{t=1}^{t_0} \alpha_t g_t(\mathbf{x}_i)) - C\mu(\mathbf{x}_i)}$$

where $\mu(\mathbf{x}_i) = (\sum_{t=1}^{t_0} \alpha_t w_t(\mathbf{x}_i))^2$ and $w_t(\mathbf{x}_i)$ is the weight of an example at iteration t and C is a fixed constant. The rationale behind this type of regularization is that examples which are hard would tend to be over-emphasized and therefore would have large sum of weights throughout previous iterations. Therefore, the algorithm might be better off excluding the most difficult examples.

2.3.3.2 Introducing slack variables

Another way to regularize AdaBoost solution is to introduce slack variables [33],[34], as with SVM. An approximation of AdaBoost, called LP-AdaBoost [20], is used. In LP-AdaBoost the hypotheses are selected using AdaBoost and then linear programming is used to find best combining coefficients in order to achieve maximum margin. Grove and Schuurmans [20] showed experimentally that maximizing the minimum margin gives a suboptimal solutions especially in noisy cases. Slack variables are introduced in the optimization problem so that to allow the algorithm to give up on some noisy examples (to allow for some difficult examples to violate the maximum margin condition). The formulation of regularized version of LP-AdaBoost is:

$$\begin{gathered}
\max \quad \gamma - C\textstyle\sum_{i=1}^{N}\xi_i \qquad\qquad \text{subj.to} \\
y_i \textstyle\sum_{t=1}^{T}\alpha_t g_t(\mathbf{x}_i) \geq \gamma - \xi_i, \qquad\quad \xi_i \geq 0, \quad i = 1,\ldots,N \\
\alpha_t \geq 0, \quad \textstyle\sum_{t=1}^{T}\alpha_t = 1, \quad t = 1,\ldots,T
\end{gathered}$$

where $g_t(\mathbf{x})$ are the hypotheses already found by AdaBoost and α_t are the coefficients we are optimizing for and C is a fixed regularization constant.

2.3.3.3 Modifying the cost function

Mason et. al. [28] suggest that a cost function of special type is used, so that not to emphasize too much examples which might be noisy. AdaBoost cost function is exponential, i.e., it would give exponentially large weights to misclassified examples. Instead they propose optimizing over a family of less steep cost functions. Manipulating the cost function so that it penalizes more difficult examples is again some sort of regularization for a class of boosting techniques.

2.3.3.4 Removing examples

Regularization by removing examples for AdaBoost, termed 'example shaving', has been demonstrated by Merler et al. [30]. A level of difficulty of an example is determined by the weights throughout AdaBoost iterations, similar to [34], in this case by the entropy of the weight distribution throughout iterations for each example [11]. A validation set is used to define which of the most difficult examples to be removed [30].

The usefulness of regularizing AdaBoost has been demonstrated experimentally in noisy datasets [28], [33], [34]. The need for regularizing AdaBoost has been theoretically explained in [22].

2.4 Learning with queries

2.4.1 Learning in the presence of classification noise

Angluin and Laird [2] investigated learning in the presence of classification noise in the context of learning with queries. In the classification noise scenario examples have their labels flipped with a certain probability β, but are otherwise not changed, i.e., they are assumed to come from the target distribution. They showed that if the noise can be modeled as independent source then the amount of noise tolerated can be very large but strictly less than 50%. For this setting they give a sample complexity bounds for effective learning in which the number of examples is polynomial in $\frac{1}{1-2\beta}$, where β is the noise level. That is, considerably more examples are needed to learn the concept if the level of noise is large. They proved that learning in the presence of noise is possible but the search for an optimal hypothesis for general problems is polynomially intractable with their approach. For some classes of concepts efficient algorithms exist which exploit the properties of the particular concept class.

2.4.2 Learning in the presence of malicious noise

Kearns and Li [23] consider an extension of learning with noise in which the noise is not assumed to have nice properties and generally cannot be modeled. The adversary creating the noise has ultimate powers such as changing examples' labels, returning examples from a wrong distribution, having access to the currently generated examples, i.e., it can generate new examples which are purposefully confusing given already learned examples. This model is called learning with malicious noise. For this model significantly smaller amounts of noise can be tolerated. The authors give hardness results for learning of arbitrary concepts in this model as well as constructive learning algorithms for learning particular concept classes.

2.4.3 Active learning

The objective of active learning is to select particularly informative examples in order to speed up training [25], [13].

In some sense active learning is also preoccupied with excluding examples from the dataset, but there, the redundant examples are the ones to be ignored. Smaller number of data points and considerably less computational resources are required in active learning than in the standard learning scenario to achieve the same generalization performance [41]. The assumption of active learning is that the training data is not noisy.

2.5 Outliers in a probabilistic setting

Finding and eliminating outliers can be cast in a probabilistic setting. A generative probabilistic models are assumed for the two classes' distributions. An appropriate outlier model is selected. A new hidden variable is introduced for an example being an outlier (and therefore need to be eliminated or ignored) and the problem can be solved with maximum likelihood approach and the EM [17].

Although the method is very powerful its disadvantages come from the strong assumptions made, namely the models of data distributions and outliers should be known. Apart from that, more parameters for the generatve models need to be estimated, i.e., more training data would be necessary, which might not always be available.

2.6 Data valuation

The methods described above introduce mechanisms to cope with noisy situations, usually by imposing penalty, by de-emphasizing (reweighting) noisy examples. The most common case is trying to ignore them by overpowering the presence of noisy examples by using more correct ones. Data valuation method, proposed by Nicholson [32], advocates removing examples from the training data and demonstrates it is useful on several classification problems with noise.

Data valuation [32] consists in analyzing the training data prior to learning and removing examples which might be adversary to learning. The examples are given a ranking according to their agreement with the remaining data and examples which disagree most are removed. Data valuation is created for the exhaustive learning scenario.

2.6.1 Exhaustive learning

The exhaustive learning algorithm [37] returns hypotheses from the learning model with a fixed prior distribution. Thus any hypothesis in the support of the prior distribution can be selected. That is, no actual learning is performed on the data. Note the distinction from the standard learning scenario where only hypotheses which perform well with respect to the training data (for example minimize the empirical risk) are selected. The distribution over hypotheses in the learning model in exhaustive learning is independent (agnostic) of the training data.

Although the analysis of [32] is done for the exhaustive learning scenario [37], we consider the data valuation method as a predecessor to our work.

2.6.2 ρ-criterion

A primary task in data elimination is to define which examples are difficult for learning and therefore candidates for elimination. Nicholson [32] proposes a heuristic for identifying those examples, namely examples whose error does not correlate well with the overall error rate π in the exhaustive learning. As π is unknown it is approximated by the leave-one-out error, the error of the dataset excluding the example in question.

$$\rho_{\mathbf{x}_i} = corr_g(e_i(g), e_{S_i}(g))) = \frac{E_g(e_i(g)e_{S_i}(g)) - E_g(e_i(g))E_g(e_{S_i}(g))}{\sqrt{Var(e_i(g))Var(e_{S_i}(g))}} \tag{2.1}$$

where *corr* is correlation of the error of the example $\mathbf{x}_i$, denoted by $e_i(g)$, with the error of the remaining set, denoted by $e_{S_i}(g)$, $S_i = S \backslash \mathbf{x}_i$ and $S = \{(x_1, y_1), ..., (x_N, y_N)\}$ is the training set.

Furthermore, because the expected value of errors with respect to all hypotheses in G cannot be computed analytically in most of the cases, (2.1) will be approximated by randomly sampling hypotheses from the learning model.

Thus a ρ value (2.1) is assigned to each example showing how much it agrees with the target function for this particular model. It gives information whether the example is contradicting on average the rest of the examples with respect to all possible hypotheses. The higher the correlation the more the example complies with the remaining training set, i.e., would not be expected to be troublesome in learning. Examples with negative correlation ρ would be expected to be difficult using this learning model. In practice Nicholson [32] has shown that ρ needs to be selected close to -1 in order to avoid deterioration in performance, which is probably due to the approximations in calculating ρ.

The author demonstrated that in noisy datasets removing examples from the training data is beneficial for learning and less so for non-noisy cases.

Chapter 3

Data pruning

3.1 Introduction

In the previous chapter we have reviewed several learning methods which have mechanisms to cope with noisy examples. Some of them apply a common penalty for all examples (regularization methods), others identify those examples which influence the model too much and reweigh them, assuming that the model is known and fixed (robust statistics), others model the noise source as independent white noise [2] and show that learning can be done, provided that sufficient examples are available. Successful identification of outliers can be done for models which are fixed and have very few parameters to estimate e.g. RANSAC [16]. In those cases a large amount of noise can be tolerated.

In many real-life learning problems noise is often present, the number of examples is insufficient and the model and its complexity are not known a priori. Our goal is to look into those more realistic settings.

We are interested in binary classification tasks in which we are given a fixed training set and a desired model for classification. In this setting we want to identify examples which are troublesome for the learning process for this particular training set and this particular model and which might potentially cause the model to overfit and therefore deteriorate the generalization performance. We term data pruning the process of eliminating examples which might be troublesome for learning.

In this chapter we first try to understand the problem of data pruning as well as the difficulties of solving it. We propose a way to identify and eliminate troublesome examples and show experiments and promising results of data pruning on real-life data.

In the next chapter we apply our method to the recognition of object categories in which the training data is contaminated, namely, the data may have large amount of wrongly labeled examples or examples which are otherwise difficult for the model at hand.

3.2 Problem formulation

Data pruning may be defined as follows: given training data and a learning model, find if there are training examples for which the learning and generalization performance would be improved after removing those examples. The problem as we define it is quite difficult to solve. Below, we examine the reasons why.

Eliminating the influence and detrimental effects of outliers is still an active area of research in statistics. The main challenges come from the difficulty of modeling outliers because their sources may be variable. In our case we do not assume a model of the noise.

We are considering a binary classification problem which is ill-posed in the first place even without any noisy examples. This is because the training data is finite and a small change in the data, for example adding or removing a few examples, can change the decision boundary. Apart from that the solution is not unique as the most appropriate model and model complexity are not specified [36].

Straightforward approaches to solve the problem are to sort the examples in increasing level of difficulty, difficult meaning closer to the decision boundary [11] or level of disagreement [32] and eliminate sequentially most difficult ones. An independent test set is used to identify which examples to be removed [30]. Although using validation set can be very useful, in this case we feel that using additional data to decide which examples to prune gives an unfair advantage of the pruned method with respect to the full one. Of course, we could supply both methods with a validation set in order to choose their best parameters but we feel we cannot allow the algorithm to decide on pruning an example or not using extra validation set.

A lot of research has been done in using unlabeled data alongside with labeled. The major conclusion being that using unlabeled data in addition to labeled is quite useful and handy when labeling is expensive. Those depend on the assumption that they have some reliably labeled data to start the learning and estimation from [8]. In our formulation we have to identify the correctly labeled examples.

And last but not least, the powerful probabilistic models can be applied very successfully to solve fully unsupervised problems modulo some technical problems with local minima and amount of data needed to estimate the parameters correctly. Those methods rely on the strong assumption that the data comes from a particular distribution which can be modeled. If the assumption is correct Bayesian methods would give the optimal solution. Nevertheless, we believe we cannot generally assume we know and can model the sources that generated the data and want to use the available training data in less requiring discriminative models.

3.3 The essence of data pruning problem

In this section we try to understand the problem of identifying troublesome examples and removing them from the data. We would give examples to get intuition of the problem at hand.

3.3.1 When is data pruning necessary?

We give characteristic examples of datasets with and without difficult examples and show where data pruning would be helpful. We show those characteristic behaviors on real datasets, figure 3.1.

(1) There are examples adversary to learning present in the dataset, difficult examples can mislead the algorithm and it creates a poor boundary or overfits. The difficult examples would rather be removed.

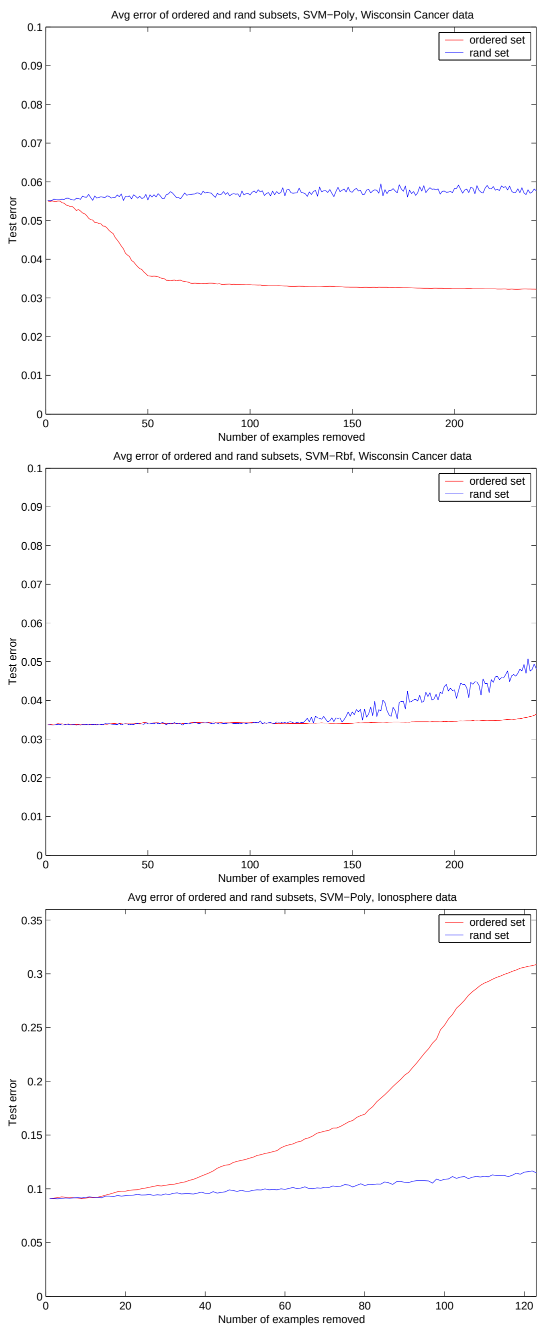


Figure 3.1: Average test errors when training on subsets of the data with increasing difficulty of examples and random subsets of the same size. The leftmost part of each plot shows the errors if no examples are removed. Only training on half of the examples is shown. Top (1) Data pruning is necessary, Wisconsin cancer data, SVM, polynomial kernel; middle (2) Data pruning is not necessary. The model can cope with troublesome examples, Wisconsin cancer data, SVM, RBF kernel; bottom (3) Data pruning would be harmful, Ionosphere data, SVM, polynomial kernel

(2) There are examples adversary to learning present in the dataset but the algorithm has mechanisms to ignore noisy examples in its optimization (for example SVM-SVC, Neural Networks). No matter that, the algorithm cannot always deal with hard examples. In those cases adversary examples can lead to poorer generalization performance, as in case (1).

(3) There are no examples adversary to learning in the dataset. Adding more difficult examples improves the test error, the difficult examples are actually useful for forming the proper boundary.

The plots of figure 3.1 show the three characteristic examples on real-life data from UCI Repository [7] in which data pruning may or may not be useful. We should note that the model in (1) has regularization capabilities but still cannot deal with all difficult examples. Conversely, on the same training data, the model in (2) can deal with difficult examples, which suggests that difficult examples are model dependent. The ordering of examples used to generate these plots would be explained in the next section.

The plots also give intuition that training on subsets of the data with increasing difficulty gives a chance to improve the generalization performance compared to training on the whole training data as is in case (1).

3.3.2 Benefits of data pruning

In this section we show that, given a learning model and a dataset, it is possible that the algorithm would perform better if training on a subset rather than on the whole set. One example has already been given in the previous section. To do that we need to search for good subsets of the data. Finding the best subset of examples to train on would need exponential number of subsets to be considered. Instead, we suggest to define a measure of difficulty of an example and train on nested subsets of examples of increasing complexity, hoping that if there are examples which are troublesome they would be identified as most difficult and training without them would decrease the generalization error.

We propose to use the margin of an example as a criterion for how difficult it is and order the examples in terms of margin. However, we do not know the exact complexity of the model according to which to measure the margin of the examples. So, we suggest to learn the model using various complexities: create nested classes of increasing capacity as in Structural Risk Minimization (SRM) [40], measure the margins of each example and average them with respect to all models to get a more reliable margin estimate.

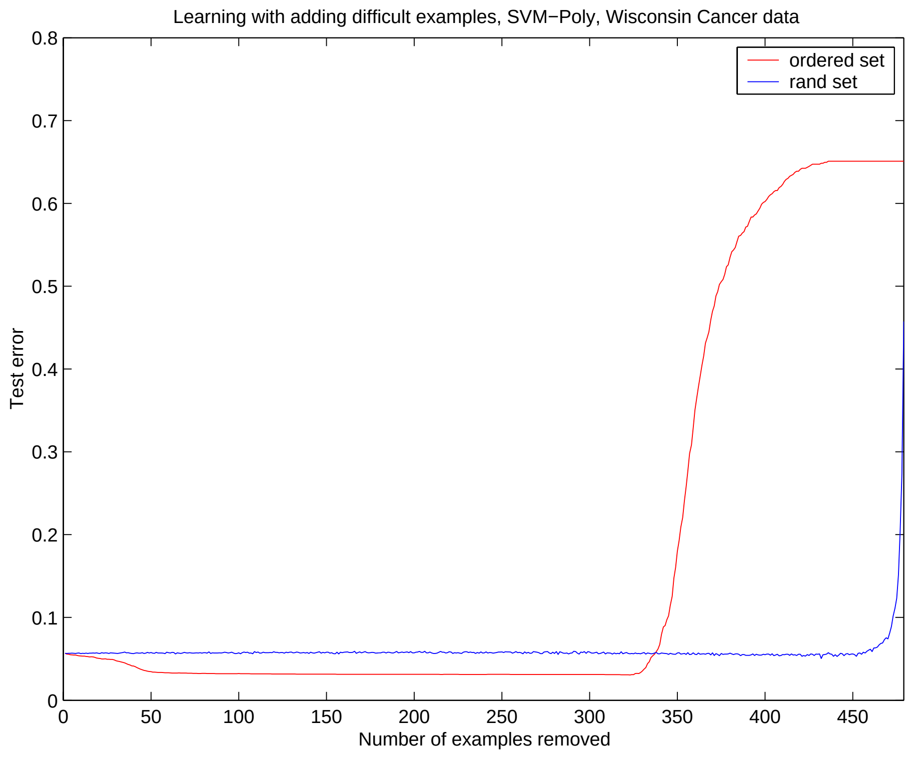
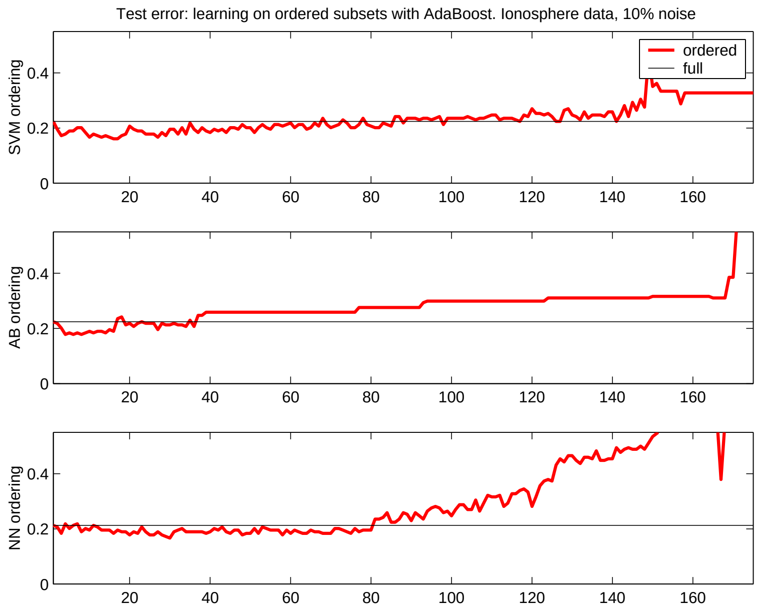


Figure 3.2: Training on subsets of the data of increasing complexity may provide for decreasing the generalization error. Learning algorithm is SVM. Wisconsin cancer data, 0% noise.

This criterion is quite natural as the margin is a measure of confidence in classification in an algorithm [35]. It is also a universal quantity because the margin of an examples can be measured no matter what learning algorithm we use.

An interesting observation is that from this heuristic measure of difficulty we can retrieve the criterion for an example being difficult defined by R*ä*tsch et al. [33], [34]



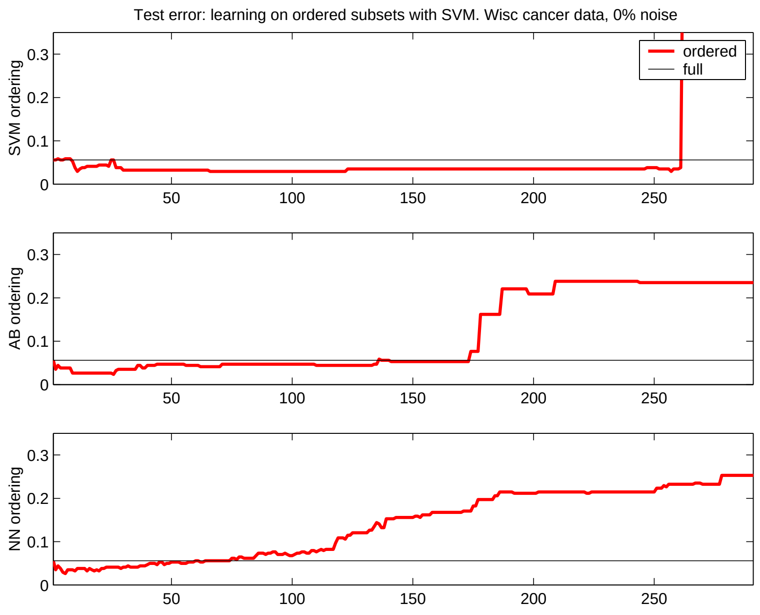


Figure 3.3: Training on subsets of the data may provide for decreasing the generalization error. Each figure shows the test error for a single run of dataset ordered in terms of difficulty. Ordering criteria are computed using the average margin for SVM, AdaBoost and Neural Network models. Top: AdaBoost learning model, Ionosphere data, 10% noise, Bottom: SVM learning model, Wisconsin cancer data, 0% noise.

in AdaBoost. The authors used the criterion to impose regularization penalty.

We could see that in some datasets it is possible to observe better generalization error using such naive and heuristic way to order examples in terms of difficulty, figures 3.2, 3.3. The plots are generated by multiple randomized runs, so we can see an estimate of the out-of-sample error, figure 3.2, and for only a single run figure 3.3 to demonstrate that this phenomenon can be observed for a particular training dataset.

We note that this is not data pruning yet. We merely demonstrate that data pruning has potential for some datasets which contain difficult examples. Note that there are datasets in which none of the examples are useless or harmful, as shown on figure 3.1.

In this paragraph we have seen that it is possible to improve the generalization error if training on a (suitably chosen) subset of the training data, for the sole reason that there are troublesome examples. In other words not only the size of the dataset matters but the quality of the examples as well.

3.3.3 Difficult examples increase model complexity

To demonstrate this point we again order the examples in terms of difficulty as in the previous section. We measure the observed complexity of the model after training on nested subsets of the training data by adding more and more difficult examples. As we noticed in the previous section we can observe better generalization error after training without the most difficult examples. In figure 3.4 we can see what the reason might be. We plot the number of Support Vectors as a measure of complexity for each nested subset of ordered training examples, as well as for random subsets of the same size. We can notice that for some datasets we observe unduly increase in the measures of complexity. For example, adding the last 15 most difficult examples for Wisconsin cancer data would significantly increase the number of support vectors used as well as the generalization error, figure 3.4. Again this results give intuition why some examples might be troublesome and causing overfitting. Alternative measure of the complexity actually used by SVM can be defined by $\frac{R^2}{\gamma^2}$ [4], where γ is the margin and R is the radius of the sphere which encompasses all the points. For Neural Networks the sum of weights on the edges can be used [3].

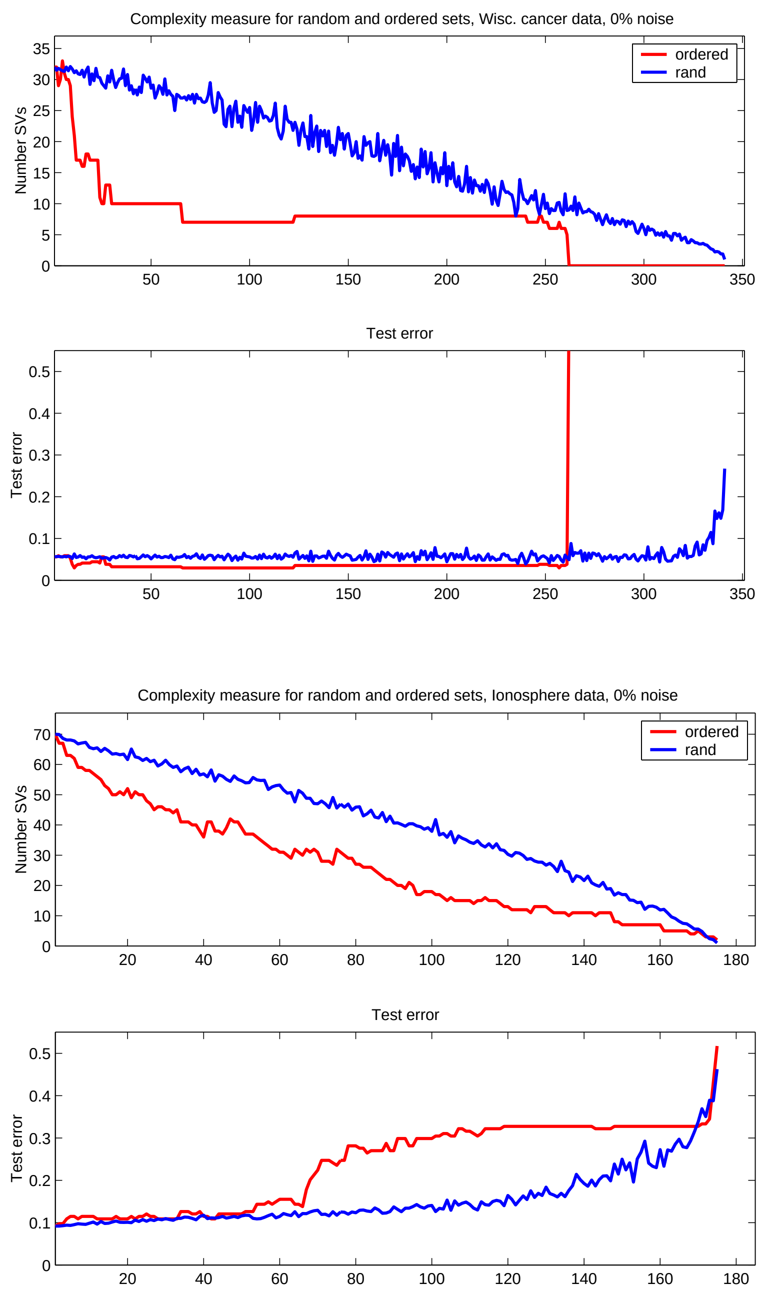


Figure 3.4: Inherent model complexity is increased by adding more difficult examples. Top plot in each figure shows the number of Support Vectors for examples entering in increased complexity and for random set of the same size. Bottom plot shows correspondent test errors. Wisconsin cancer data (top) and Ionosphere data (bottom).

Many researchers working on generalization bounds [3], [26], [35], [38] have found out that a better bound on the expected error might be estimated if some examples are ignored by the algorithm for the sake of using smaller complexity. For example, the following theorem from [3] and [38] gives a trade-off between model complexity and number of wrongly learned examples.

Theorem 3.3.1 *Let F be a class of functions. With probability (w.p.) at least $1-\delta$ over m iid samples $\mathbf{x}$ from some fixed probability distribution P the following holds: if $f \in sgn(F)$ has margin at least γ on the examples $\mathbf{x}$ then the expected error of f satisfies:*

$$P(f(X) \neq Y) \leq \tfrac{2}{m}(dln\tfrac{34em}{d}log_2(578m) + ln\tfrac{4}{\delta})$$

For any other $f \in sgn(F)$ (which may have errors on $\mathbf{x}$) w.p. at least 1-δ:

$$P(f(X) \neq Y) \leq P^{\gamma}_{m}(f(X) \neq Y) + \sqrt{\tfrac{2}{m}(dln\tfrac{34em}{d}log_2(578m) + ln\tfrac{4}{\delta})}$$

where $d = fat_F(\gamma/16)$ and $P^{\gamma}_{m}(f(X) \neq Y)$ is the portion of examples with margin less than γ.

A similar bound for combination of classifiers, for example AdaBoost, was given by Schapire et al. [35]: w.p. at least 1-δ.

$$P(f(X) \neq Y) \leq P^{\gamma}_{m}(f(X) \neq Y) + \sqrt{\tfrac{c}{m}(\tfrac{dln^2(m/d)}{\gamma^2} + ln(\tfrac{1}{\delta}))},$$

where d is the VC dimension of the base classifier.

The influence of difficult or impossible to learn examples can be diminished by imposing a simpler model or rather a trade-off of the model complexity and fitting the data well. This is again another form of regularization.

Unfortunately, for our example elimination tasks, those bounds are post factum, meaning they estimate what the generalization error bound would be after the algorithm has failed to learn some of the examples, or after certain margin is observed. They are not constructive in the sense that they do not give ways to identify the examples which are troublesome so that the algorithm can benefit from ignoring them before starting to learn on them. The bounds are loose and can be used for general penalty in model selection [3], [27], but are not precise enough to estimate best trade-off to determine which examples to be removed.

Generally, those bounds are useful because they state that it might be reasonable that some examples be removed if their presence in the training data increases unduly the expended model complexity. What is needed is to have a constructive way to find out which are exactly the examples to be removed.

3.3.4 Regularization can benefit from data pruning

Regularization methods are a very successful way to ignore or decrease the influence of some noisy examples. By penalizing overly complex models a trade-off between model complexity and number of examples not learned by the algorithm can be achieved.

However, in the figures we have seen above, figure 3.2, we have a regularized algorithm (SVM with slack variables, called SVC) which can still benefit from pruning of some noisy examples. We can observe that there are subsets of the data which would improve generalization error. Thus, there might be examples which are so difficult that even though the algorithm has intrinsic mechanisms to cope with noise, they might still influence it in an adversary way.

3.3.5 Challenges for data pruning

In this subsection we analyze important issues of what challenges and constraints we have to face when dealing with noisy examples.

Model dependent or model independent definition of a difficult example

In most of the cases in practice, algorithms, provided that they are of comparable power, would share discriminatory boundaries on the same data, so the difficult examples for one model would also be difficult for the other, see figure 3.5. So, criterions for pruning which depend on other models would also give satisfactory results. See, for example, figure 3.3, where criterions for ordering in terms of difficulty based on Neural Networks, SVM and AdaBoost can be used for decreasing the error with a different learning model, here SVM or AdaBoost. In general, however, the examples which are impossible to accommodate by one model (i.e. are difficult) might be easy to fit by another model. On figure 3.6 we show SVM model with different kernels on the same dataset: SVM model with polynomial kernel would be influenced by some examples and have larger test error if using the whole data while SVM with RBF kernel can cope with those examples.

It seems a general criterion should depend on the learning model but if there are particularly bad examples in the dataset they would be troublesome whatever model is used, so, in practice a model different from the target learning model can be used to do the pruning.

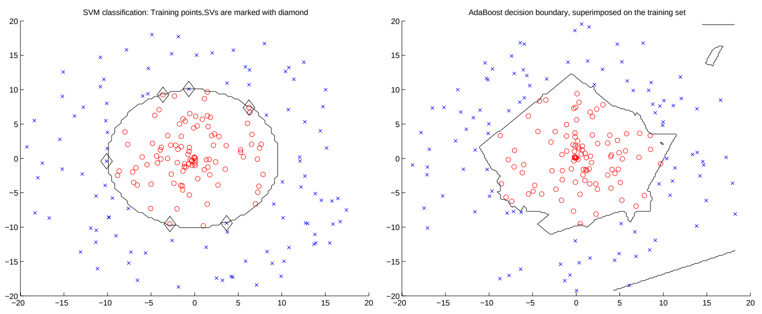


Figure 3.5: Algorithms may share boundaries on the same datasets and therefore difficult examples. Decision boundaries for Support Vector Machine (left) and AdaBoost (right) on the same data.

Definition of difficult examples depends on the whole data

As we discussed earlier, the definition of an outlier depends on the remaining examples and that a general penalty on the whole data may solve only partially the problem. So, we believe we need to consider examples individually for elimination but in the context of the rest of the data.

Training data is insufficient

Learning from examples is an ill-posed problem in the first place because of the finiteness of the training data and the fact that it is not known what class of functions the target belongs to. If the data is noisy then the problem is even less well defined. With data pruning, we would like to investigate a more complicated problem, namely, whether it is possible to find training examples which are troublesome for learning,

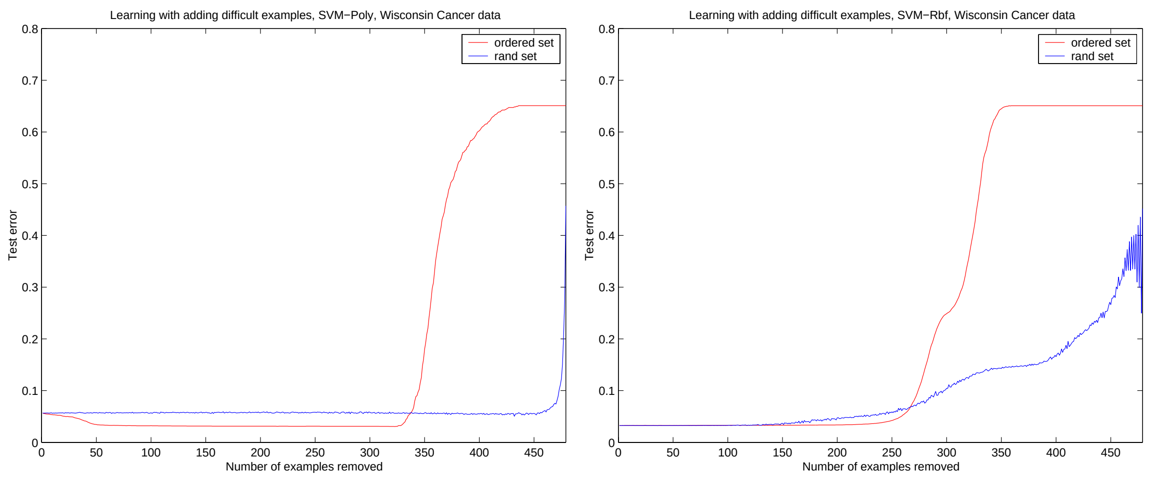


Figure 3.6: Definition of difficult example depends on the model. Test error for ordered and random sets for SVM with Polynomial kernel (left) and with RBF kernel (right) on the same data. The left plot is taken from fig. 3.2. It is given here for the sake of comparison with another learning model.

where we do not know if and how much noise is present.

In many of the cases coping with noisy examples is done by relying on large amount of data: RANSAC [16], query with noise [2], [23], etc. We are not entitled to using more examples than the ones given.

Model and model complexity is unknown

Unlike probabilistic approaches here we cannot model the data. We take discriminative approach in which the general model is fixed but its complexity is unknown. For example we may want to use Neural Networks but we have to select or learn the topology of the network as well as some parameters on it. This gives additional complications, because overly complex models would fit the training data perfectly, i.e., would overfit with respect to the difficult examples and would consequently have poor generalization. Models which are too simple would not be learning the data well enough, so many good examples would appear to be difficult.

3.4 Data pruning

3.4.1 Overview of the approach

Given the dataset and the learning model we want to find out if there are examples in the training data such that the learning algorithm would give possibly better generalization error when training without them.

Our approach consists in randomizing the data so that to select multiple classifiers which are of comparable power to the target learner and are diverse enough to have semi-independent classification. These classifiers would be affected in different ways by the troublesome examples. Probabilistic inference with naive Bayes classifier is done over the multiple classifiers' opinions to decide which examples are troublesome.

3.4.2 Multiple models and data subsets are needed

Suppose the training data has troublesome examples and we knew the appropriate model complexity. If the learning algorithm is applied to the whole data then it would be influenced in an adversary way by these examples and would overfit or create a poor discriminatory boundary. The trouble is that we do not know which examples are actually causing the poor behavior. If we measure how well the examples do with respect to the decision boundary it would not be correct because the algorithm may have overfitted and learned those outlying examples very well. This problem is well known in robust statistics [43].

One possible solution is to allow those troublesome examples to influence the solutions in different ways, i.e., have both large and small influence. This can be achieved by some sort of randomization of the training data. The correspondent learners created from randomizing the data would be diverse and would be influenced in different ways by the troublesome examples. However, assuming that the outlying examples are not the majority of the data, most of the learners would be consistent with the target function. Thus the troublesome examples can be identified as the ones which create disagreement among classifiers.

3.4.3 Collecting multiple learners' opinions

In the previous section we have argued that multiple diverse classifiers are needed to be able to retrieve wrongly labeled or otherwise difficult examples.

One way to create multiple semi-independent classifiers is through bootstrapping [15]. It is appropriate for smaller size datasets and for small number of input dimensions. Other ways of creating diversity of classifiers are through selecting different subsets of input features or selecting slightly overlapping subsets of the data or even through injecting noise in the training data [10].

In this section we would stick to bootstrapping as one possible approach. Using learners from bootstrapped data does not guarantee diversity but bootstrapping data does encourage it [9].

Why bootstrapping? It is known that the existence of outliers in the data can have detrimental effect on the final bootstrapped classifier because some resampled subsets may have higher contamination levels even than the initial set [1]. This might be a problem for creating a robust estimate by using bootstrapping, but would be an important observation for our goals. Namely, if some difficult examples would influence some of the learners in an adversary way then using many other learners which are not very badly influenced can identify this discrepancy.

Several explorations have shown that resampling with replicates gives an imbalance in influence of certain random examples. The experiments of [12] show that among multiple datasets created with bootstrapping there would be examples which may not enter any resampled data as well as examples which would enter and influence the committee multiple times. The important realization is that through bootstrapping we can achieve an imbalance of influence of some random examples, but of course we do not know which they would be. The authors of [12] as well as many others are concerned with modifiying the bootstrapping sampling scheme so that to give equal influence of each example. Conversely, we use bootstrapping to take advantage of the imbalance of influence it gives. Our hope is that through multiple resamplings we can give various opportunities of examples which are poor to influence in larger or smaller extent the training set and therefore create classifiers which would be closer or farther from the target.

Now, instead of aggregating the classifiers hoping to average out the poor effect of outliers on some classifiers, as often done in learning with ensembles [9], [18], we would apply inference machine to find out which examples have created discordant classifications.

3.4.4 Pruning the data

So far we have retrieved multiple semi-independent learners which can classify the data. The label and the confidence of classification of each learner, which we refer to as a response of a classifier, can be used as opinion of which example is difficult and its level of difficulty. The responses of the classifiers can be considered as projections of the initial data.

Now we want to combine the classifiers responses or opinions of which example is difficult in an appropriate way.

3.4.4.1 Combining classifiers' votes

Instead of simple voting we suggest to use inference with probabilistic models to determine the true label of an example. We are interested in finding the probability $p(y|\mathbf{X})$ of the label of an example y, given the data $\mathbf{X}$. In fact the label would be determined by the ratio $\frac{P(y=1|\mathbf{X})}{P(y=0|\mathbf{X})}$.

$P(y=c|\mathbf{X}) \propto P(\mathbf{X}|y=c)P(y=c)$ where $c \in \{0,1\}$ denotes a class label.

$\frac{P(y=1|\mathbf{X})}{P(y=0|\mathbf{X})} = \frac{P(\mathbf{X}|y=1)P(y=1)}{P(\mathbf{X}|y=0)P(y=0)}$

The ratio is compared to 1 and if larger or equal then the estimated label of an example is 1, otherwise 0.

The probabilities $P(y=1)$ and $P(y=0)$ are set to our prior belief of the data. They can be set to $\frac{1}{2}$ if the examples come in approximately equal proportions.

In both expressions $P(\mathbf{X}|y=0)$ and $P(\mathbf{X}|y=1)$ can be modeled and estimated from the data or in our case the projections from the multiple learners.

After estimating the new labels, the examples whose labels disagree with the original ones are pruned.

3.4.4.2 Naive Bayes classification

A simplest way to model the data is to use Naive Bayes classifier and decompose the data into several independent attributes. In our case the attributes are the projections of the input data on several classifiers: $P(\mathbf{X}|y = c) = \prod_{j=1}^{J} P(A_j|y = c)$

In order to use Naive Bayes, an assumption of the attributes being independent is crucial. Quite often, however, the rule can be successfully applied even if the attributes are not independent.

Note that we might have started with a probabilistic model in the beginning to reason about which example is wrong. But this would require us to assume a particular probabilistic model of both the data and the difficult examples which we cannot do with enough generality.

3.4.5 Why pruning the data?

In the previous section we have estimated which examples create most disagreement among classifiers and therefore are most difficult to learn. Those examples might come from various sources. The simplest assumption is that there is an oracle which flips a coin and with certain probability provides a wrong label. The examples are assumed to come from the genuine distributions. This is the classification noise scenario of Angluin and Laird [2]. In this case, the best way to proceed is to flip the labels of the examples identified as noisy. However, in real-life dataset this is not a realistic assumption: the examples may come from different distributions, may be result of measurement errors, may have errors introduced in the data component not only in the label.

So, as we do not know the source of those troublesome examples, we suggest to eliminate them from the data, i.e., do data pruning. We consider it more dangerous to flip the labels because the classification noise scenario cannot be guaranteed in practice.

We will see an example of that in the next section, where we train on data of face category from very challenging images. In this dataset we may have face examples which are correctly labeled but are inherently hard to learn because of poor illumination, pose variations, etc. In this case pruning the data to receive a better training set is more appropriate.

3.4.6 Experiments

We show results of data pruning on several datasets from UCI [7]. We apply data pruning to the original dataset, as well as to the data with artificially introduced label noise.

Results of data pruning for three datasets are shown on figure 3.7. Summarized results for more datasets are shown on figure 3.8. The following datasets from UCI [7] are used: Wisconsin breast cancer, ionosphere, twonorm, votes, pima indians diabetes, waveform, wine, mushroom, sonar and credit. For multi-class datasets, such as wine and waveform the first two classes are used. In some datasets the examples with missing attributes are removed. For each dataset, half of the available data is used for training and the rest for testing. The results show the average test errors from 100 independent runs.

In all our experiments we apply cross-validation on the training data to try to find the most appropriate model parameters. This is a practical way to solve the problem of model complexity not known. Note that only the training set is used to do that.

The results in figures 3.7 and 3.8 show advantage of data pruning method, especially if noise is present. For some datasets, e.g. votes, credit and mushroom, the pruning method can identify that no troublesome examples are present and not do pruning at all but for datasets like ionosphere, where all examples are needed, see figure 3.1, the pruning method is detrimental. This calls for the need to refine the decision of which examples to be eliminated and revision of Naive Bayes for these purposes, which we plan to address in future work.

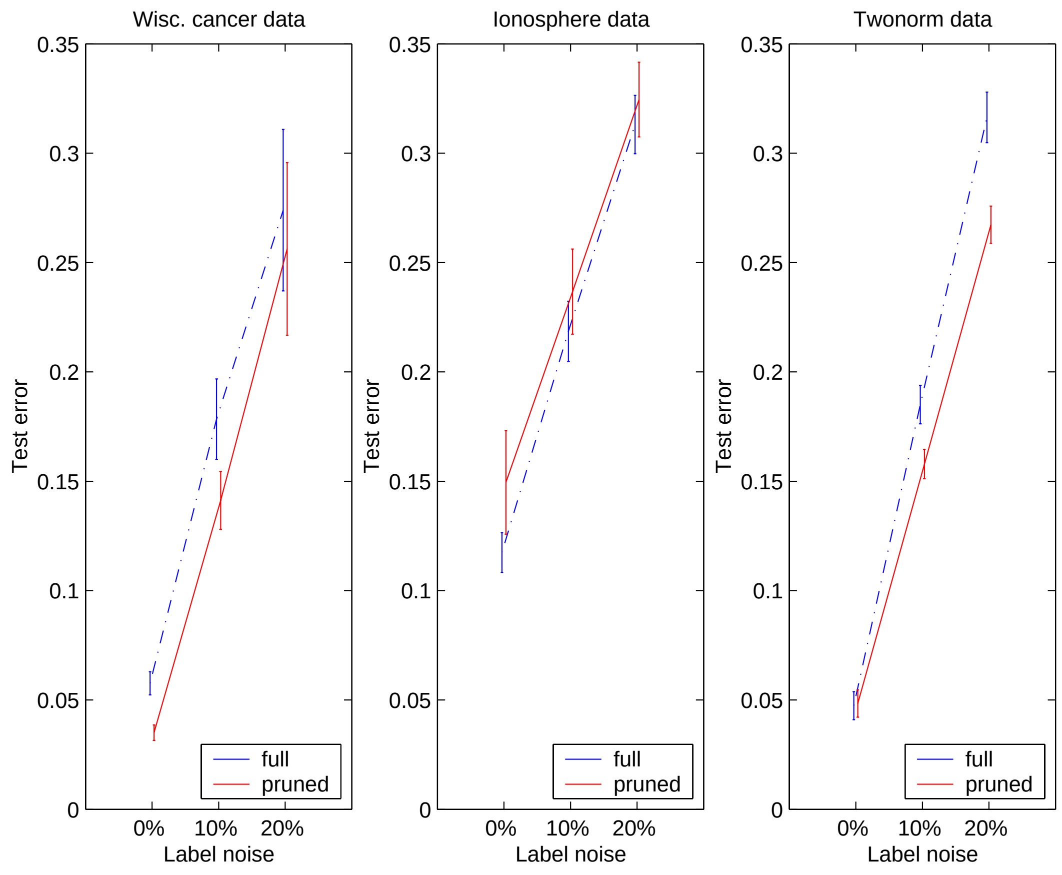


Figure 3.7: Pruning results for UCI data. SVM model. The pruning is done using 100 SVMs over bootstrapped data

3.5 Discussion

Several remarks concerning the method follow.

It is very difficult to define which examples would be troublesome and what are the cases in which those examples are harmful for the algorithm and should be removed. For, the training data is not sufficient, the best model complexity is not known and some examples may be actually forming the decision boundary and may be harder to learn but also very important in the dataset.

Data pruning is shown to be useful on several real-life datasets. Quite naturally, the data pruning method does not always select the best subset for further learning, especially if no noise is present. Regularization methods are also shown to be detrimental if applied to data with no noise [24].

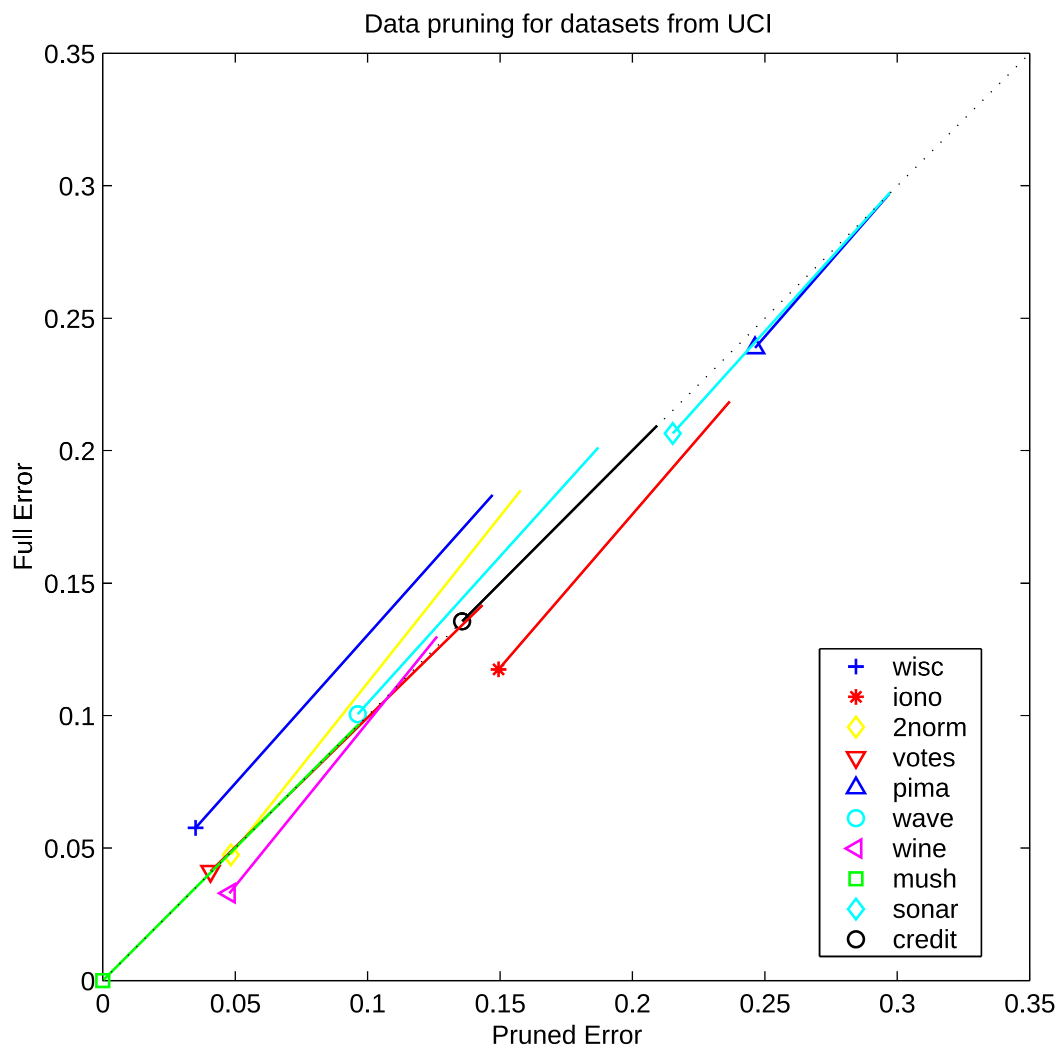


Figure 3.8: Data pruning results for UCI datasets. SVM model. The pruning is done using 100 SVMs over bootstrapped data. The character denotes the average test errors over 100 runs on the original data and the lines point to the test errors on the same data with 10% label noise

The work of Caprile, Merler and Furlanello [30], suggests estimating the examples' hardness for AdaBoost algorithm and removing hardest examples. The methodology for deciding on a cut-off threshold is based on a validation set. In data pruning we addressed a more challenging problem, i.e., to determine which examples would be troublesome for learning without using any sort of validation set, additional data or prior information. To our knowledge this is the only work that considers the problem in this more realistic setting.

The approach is different from the standard regularization methods. The regularization methods are general penalties, applied to the whole data thus can affect also the good data points, instead, we reason about each data point separately. Moreover, regularization methods are defined a priori, before seeing the data.

Using multiple classifiers and combining their decisions for more reliable prediction has been in the literature for a while [18], [9]. However, in this work we do not use combination of classifiers to learn the target, rather, we use multiple projections for the purpose of identifying difficult examples which are troublesome and remove them. The final learning is done on the pruned set using a single learner of the learning model, not a combination of learners.

It is no wonder that various methods which try to cope with noise use multiple classifiers for that purpose, because, if no other information is known about the data and there are noisy examples the only way to find out which examples are noisy or to somehow ignore them is to use multiple projections of the data, hoping that noisy observations would affect the classifiers in a discordant way and the good examples would promote consistent with one another classification.

3.6 Conclusion

In this chapter we have explored the data pruning approach which improves learning in the presence of noisy or outlying examples. Data pruning advocates eliminating examples which have been identified as troublesome because they might be inherently hard for the model.

Several important problems are left unexplored. What kind of multiple learners can be used? Can we expect to perform equally well if we use weak learners, learners slightly better than random guessing? What amount of noise can be tolerated in this case? How does the level of independence among learners affect the pruning results? We hope to be able to address them in future work.

Chapter 4

Cleaning contaminated data

4.1 Introduction

Object recognition is an easy task for humans, yet, still difficult for machines. Learning of visual object categories by humans is done by seeing examples, rather than using predefined rules or descriptions. Inspired by the learning people do, most contemporary algorithms use training examples to learn a category.

Training examples are difficult to obtain and require human interference to manually collect and label which examples belong to the category, a tedious and time consuming task. Naturally, we are aiming to acquire training data with minimal human supervision. For example, web search machines, such as Google [19], can return quite a lot of images of object categories as a response to keyword queries. Unfortunately, the image search and indexing is not content based, but uses image caption and context instead. So, among the returned images, there would be many which contain the target category but also numerous of images which are not relevant to the target. We can further assume we have access to infinite number of images which do not contain the target category or might contain it with small probability (for example if we query the web search machine with other keywords).

In this setting, we can apply the data pruning method to clean the training data so that to achieve a better training set for learning.

Another important motivation to use data pruning, even in the cases when no noise is introduced, is to identify examples which are correctly labeled but difficult for learning. This is useful, especially for target images, because there might be examples with poor illumination or extreme pose variations which the learning machine cannot accommodate and would generalize better if training without them.

We shall note the difference to active learning scenario. In active learning we can have many unlabeled examples and assume that we can query for the object label at any time, but this query is more expensive than pulling out an unlabeled example [13]. In our case we have a set of examples which contain instances of the target but may have a lot of wrongly labeled examples. We can possibly ask for more data of the same type but cannot access the true label of the examples.

In this section we apply the data pruning method to clean contaminated face data, modifying it to take advantage of the fact that we are learning visual categories. For our experiments we solve an easier problem in which the examples are aligned.

4.2 Experimental setup

4.2.1 Training data

In our setting we have training examples from one object category (human faces). Among them we might have non-face examples which have been erroneously labeled as faces. Apart from that we have available numerous images which are known not to contain the target object and we can potentially create infinite non-target examples by cutting random patches from those images. The chance that there is a target patch among them is very small, almost zero. Example training data is shown on figure 4.1. The face images, even without injected noise, may still contain very difficult examples, as mentioned before, and may be hard to learn.

In order to learn the object category we align the target patches. We plan to extend to nonaligned target objects in future work.

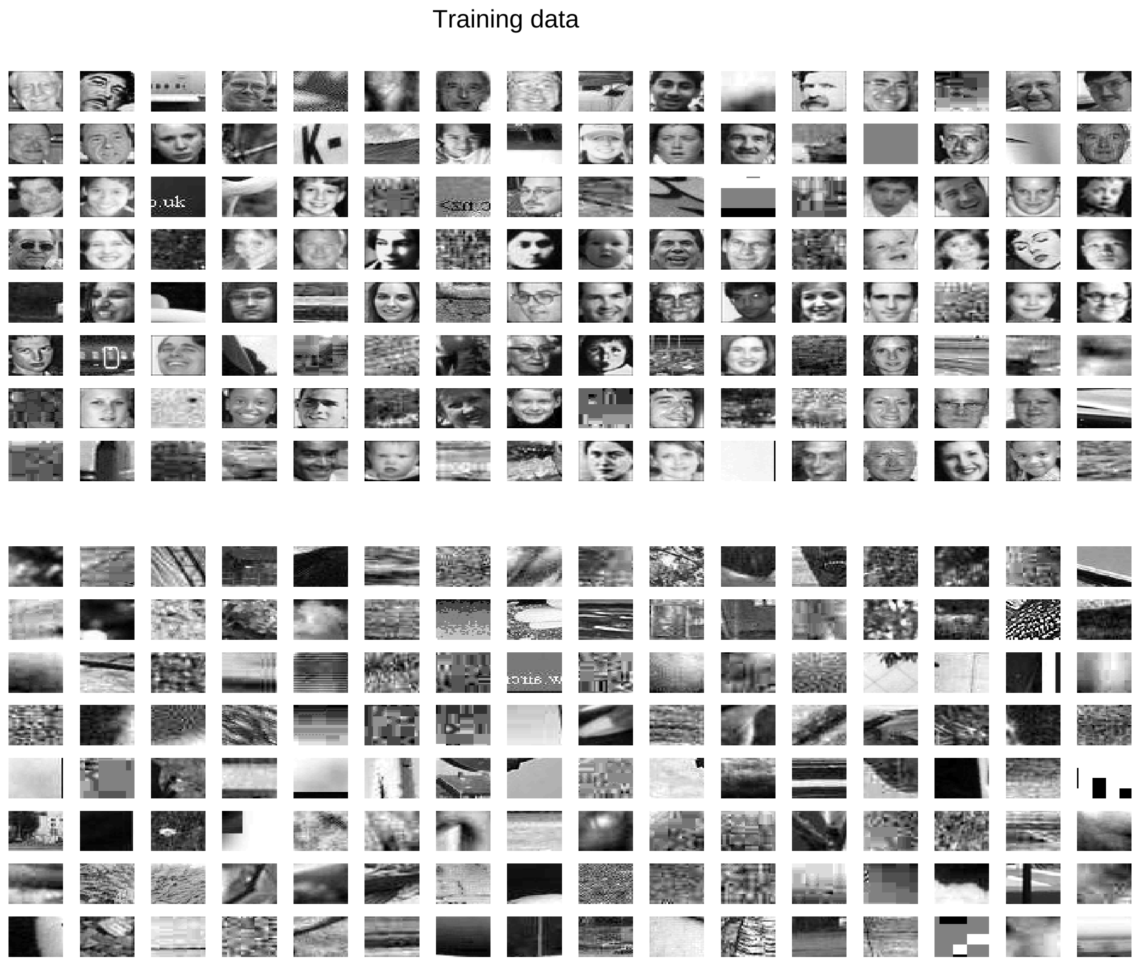


Figure 4.1: A subset of the training data, face examples (top) are contaminated with wrongly labeled non-faces, non-face examples (bottom) are random patches cut from images not containing the target.

4.2.2 Feature projections

The training data comes in patches of size 32x32. Instead of using pixel-wise correspondence we exploit the fact that we are working with visual data and select suitable projections, or features, which capture local dependencies of pixels, as well as invariances to small translations and illumination changes. Using features, rather than raw pixel values, has also the advantage of working in smaller dimension.

To ensure invariance to illumination changes, the images are preprocessed with filters, forming several channels. Some of the filters are shown on figure 4.2. We use the following filters: Gaussian 5x5 smoothing filter, Laplacian 3x3 filter, 5x3, 7x3, 3x5 and 3x7 horizontal and vertical long edge filters, four steerable filters based on Gaussians at 45°. Similar filters are proposed in [31]. From each filter channel the sums of pixels in sub-regions of the image, represented as rectangular masks of activity, figure 4.3, are taken. The object masks are also borrowed from the paper of Murphy et al. [31].

The feature projections are selected in this particular way for the following reasons:

The face dataset, as seen in figure 4.1, contains quite challenging images with large variation in terms of illumination, pose, image quality, etc. No matter that the faces are aligned, pixel-wise correspondence or even correspondence within small neighborhood is highly unlikely to give consistent response among the face examples. Thus in order to select consistent features the pixels in larger regions need to be used, so that to accommodate for variability in terms of local translations, rotations etc. The easiest way to ensure variability is to sum the pixels in the region, as proposed by Viola and Jones [42]. However, there would be large variations in illumination, so working on changes in intensity would be more informative than on raw pixel values. That is why we used the sum of pixels in several filter channels instead in the initial image. We shall note we used only the first moment statistics (sum of pixels) in the rectangular regions, while Murphy et al. [31], who created the features to discriminate among many objects, used the second and fourth moment statistics.

4.2.3 Learning algorithms

Once we have projected the training data on certain potentially more useful feature projections we can apply any algorithm for learning. The learning algorithm would receive as input only the set of extracted features. For our particular application we would be using Support Vector Machines (SVM) and AdaBoost.

4.3 Data pruning for visual data

We apply data pruning for cleaning visual data using the method described in the previous chapter. Apart from using general methods to train multiple independent

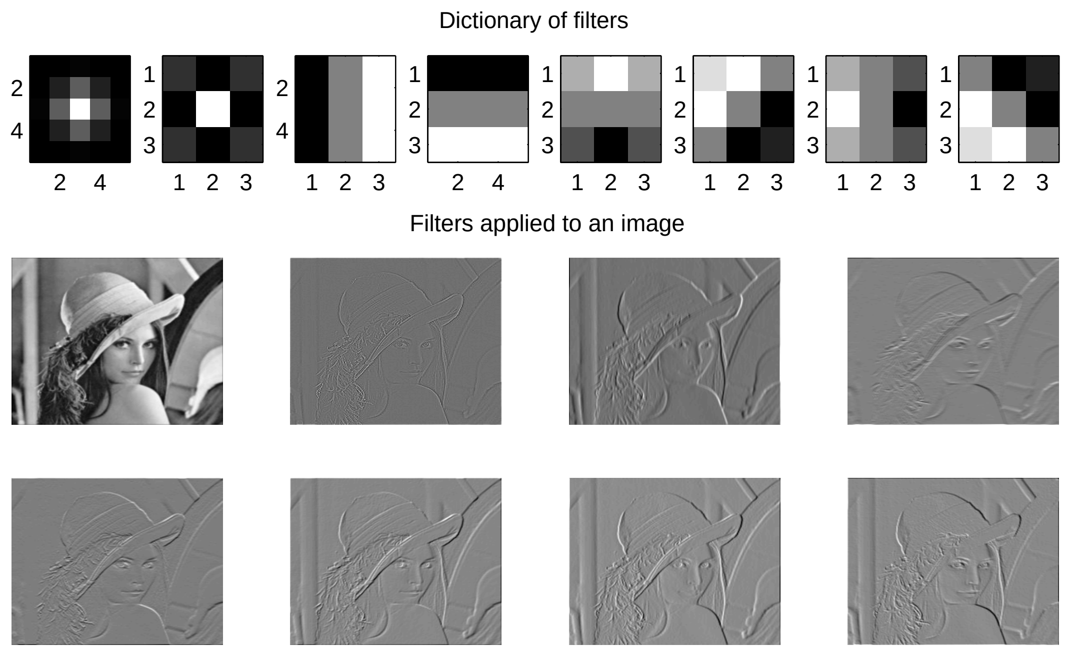


Figure 4.2: Dictionary of filters (top) and example of their application to the image of Lena (bottom). The filters (left to right) are as follows: Gaussian 5x5 smoothing filter, Laplacian 3x3 filter, horizontal and vertical long edge 5x3 and 3x5 filters, four steerable filters based on Gaussians at 45°. Similar filters are used in [31]

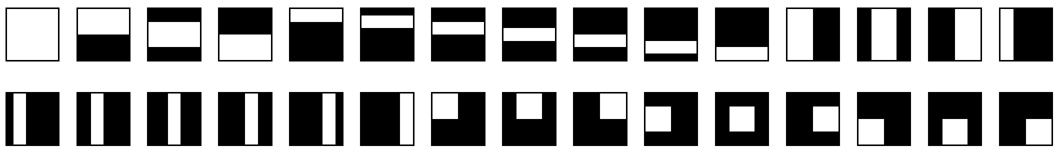


Figure 4.3: Masks selecting sub-regions of the object [31]

learners, such as bootstrapping, we exploited the fact that the training data is for a visual category and proposed more powerful and useful learners. Each of those learners gives an opinion of which examples are hard to learn and their opinions are combined with Bayesian reasoning. In this way examples are marked for elimination and further learning is done on a pruned set of the data.

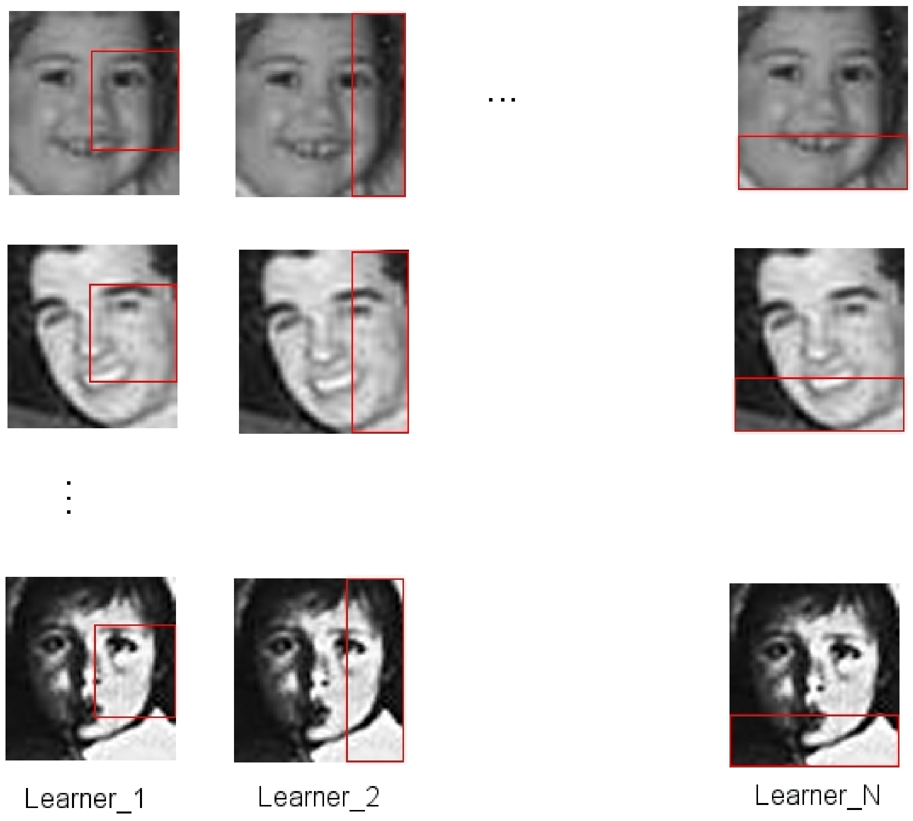


Figure 4.4: Sub-regions for training multiple independent learners for face category

4.3.1 Generating semi-independent learners

The visual object category gives more freedom in training semi-independent learners because it provides for many potential feature projections. So we can easily select large subsets of feature projections which are slightly overlapping (i.e. diverse) and which are informative enough. Note that this is harder to do with a dataset of small number of input dimensions.

To produce semi-independent learners we can split the image in slightly overlapping regions. We use regions defined similarly to the ones in figure 4.3 but drop the rectangles which take too large areas such as the first four, resulting in a total of 23 learners, so that not to receive fully dependent classifiers, see figure 4.4. We call them region learners. Each classifier receives only the visual information in one of those regions and is trained using AdaBoost with rectangular features, as in [42]. For those learners the values in the channels of figure 4.2 are used. The features allowed are rectangles of various sizes at different positions of the sub-region mask. Those features are simpler than in [42]: within each rectangle only the sum of pixels is taken. Note that linear combinations in this dictionary of features is sufficient to simulate the original ones in [42], but would need more features to do so. The images are normalized as suggested in [42].

4.3.2 Data pruning results

In our experiments we compare the results of an algorithm, say SVM or AdaBoost, on the full and pruned dataset. The algorithm works in the feature space as described above and is independent of what the data pruning procedure would be. The data pruning step only provides a pruned subset of the data to train on. In the case of face category our data pruning procedure can be model independent, for example based on the region learners, in which case it would be the same for both AdaBoost and SVM and can use potentially different feature projections, or model dependent, based on bootstrapping the data and creating multiple learners of the same type as the original algorithm.

To simulate contaminated data, we artificially flip the labels of some non-target examples and compare learning on the full data with learning on the pruned one. The data pruning module is as described above. In our experiments we have 1000 examples in which a number of background examples are labeled as face, dependent on the noise level, the remaining examples are split in half among the faces and non-faces with true labels. For example, for the 90% noise case we will have 900 non-face examples labeled as faces, 50 correctly labeled faces and 50 correctly labeled non-faces, for 0% noise we will have 500 face and 500 non-face examples all of which are correctly labeled. The test set is composed of 500 face and 500 non-face examples. It is independent of the training set and has not been contaminated. Within each run the same test set is used to report the results of learning on pruned and full sets.

The results of learning with SVM, using bootstrapped learners for pruning, are shown in table 4.1 and figures 4.5 and 4.6. As we can see pruning the data is helpful especially in the presence of large amount of noise. The test error, while training on the full set, increases steeply with increasing the noise level, much less so if we preprocess the data and prune it. Pruning is not harmful in the no noisy case. The scatter plot, figure 4.6, shows the errors in each individual run, so that we can see that pruning the data is consistently better and is not due to isolated felicitous cases. Examples which have been selected from the algorithm to be pruned can be seen on figure 4.8.

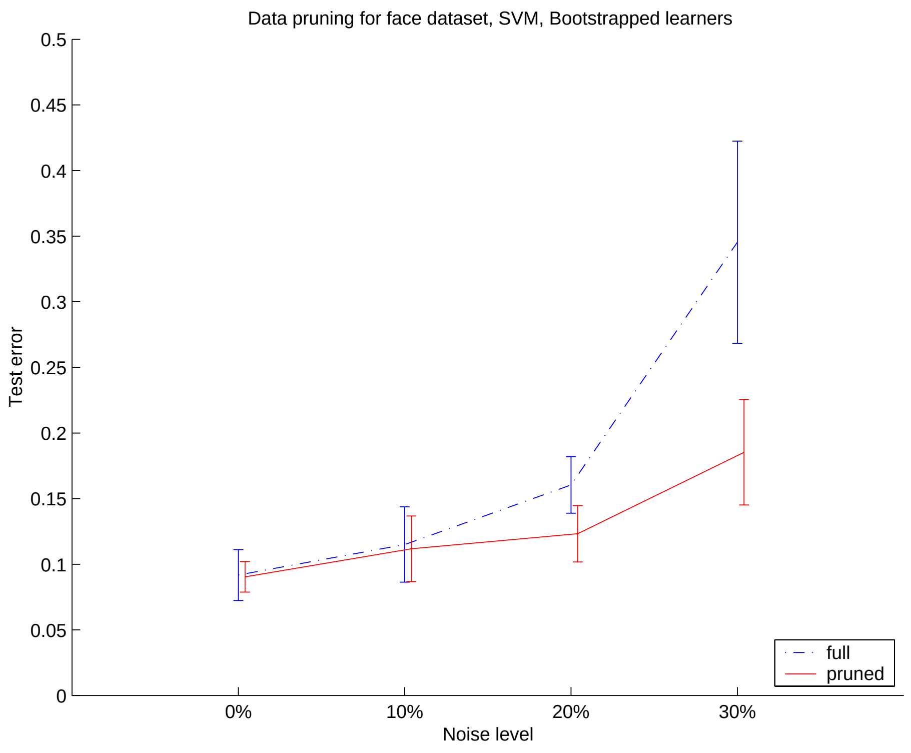


Figure 4.5: Test errors of learning on full and pruned datasets for different levels of noise. SVM algorithm. Pruning based on bootstrapped learners.

For comparison, we provide experiments with SVM algorithm where the pruning is done by region learners and with AdaBoost algorithm, again with region learners based pruning, figure 4.7. In the first comparison, the learning algorithm is the same,

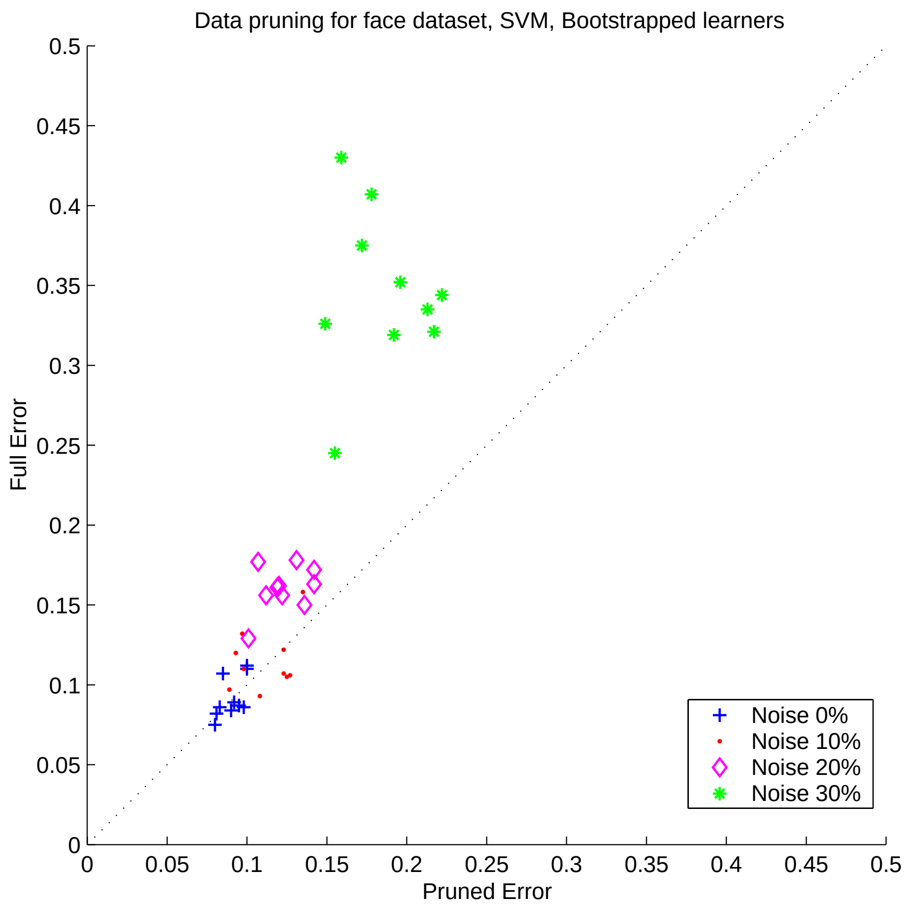


Figure 4.6: Test errors of learning on full and pruned datasets for different levels of noise. Scatter plots. SVM algorithm. Pruning based on bootstrapped learners

Table 4.1: Average test error for learning on pruned and full face data, SVM algorithm. Pruning based on bootstrapped learners

Noise	**Full**	**Pruned**
0%	0.0918	0.0904
10%	0.1150	0.1118
20%	0.1604	0.1232
30%	0.3454	0.1853

SVM, but the pruning stage is done with region learners, specifically proposed for visual data, or with bootstrapping, a more general method considered in the previous chapter. Here we have the chance to compare the pruning mechanism. Both pruning methods seem to be performing comparably well, but the bootstrapped learners, as

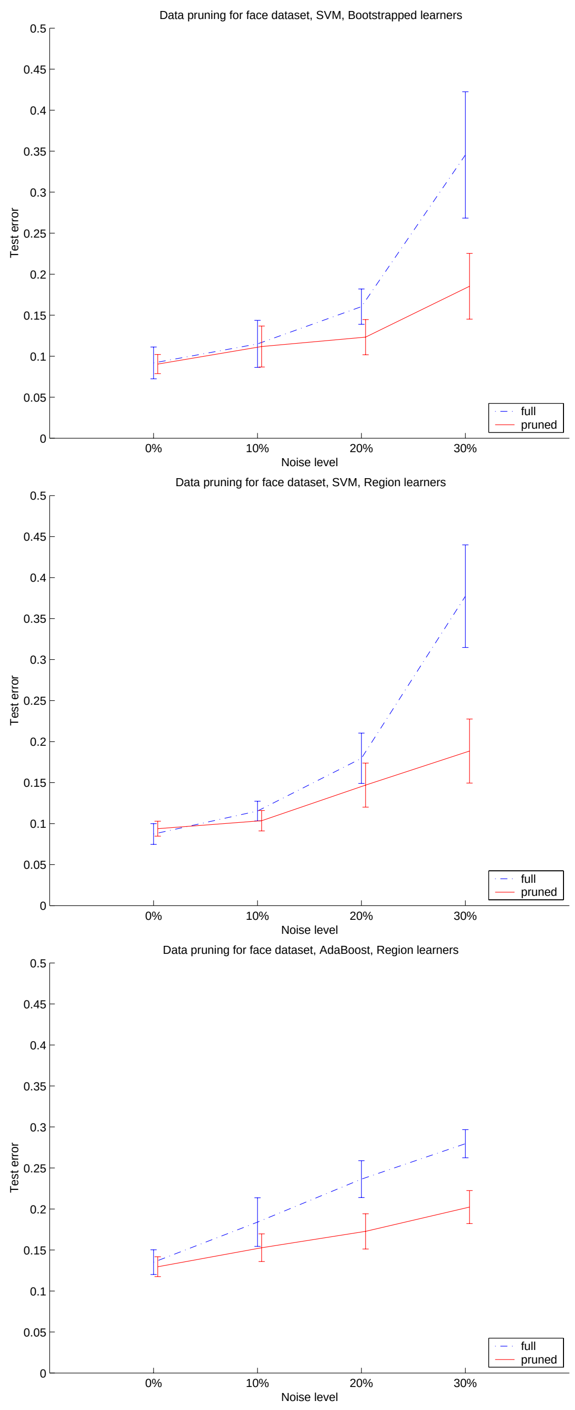


Figure 4.7: Comparison between different pruning mechanisms and different basic learning algorithms: SVM with bootstrapped learners based pruning (top), SVM with region learners based pruning (middle) and AdaBoost with region learners based pruning (bottom)

we will see later, allow for more improvement in the generalization error for large amounts of noise, see figure 4.10. In the second comparison both SVM and AdaBoost use the same pruning mechanism and we can notice only the different reactions of the algorithms to different levels of noise. Namely, AdaBoost is sensitive to even small amount of noise and would benefit from pruning in any noisy situations. SVM is robust to small amounts of noise, but would need pruning much more than AdaBoost in large noise cases.

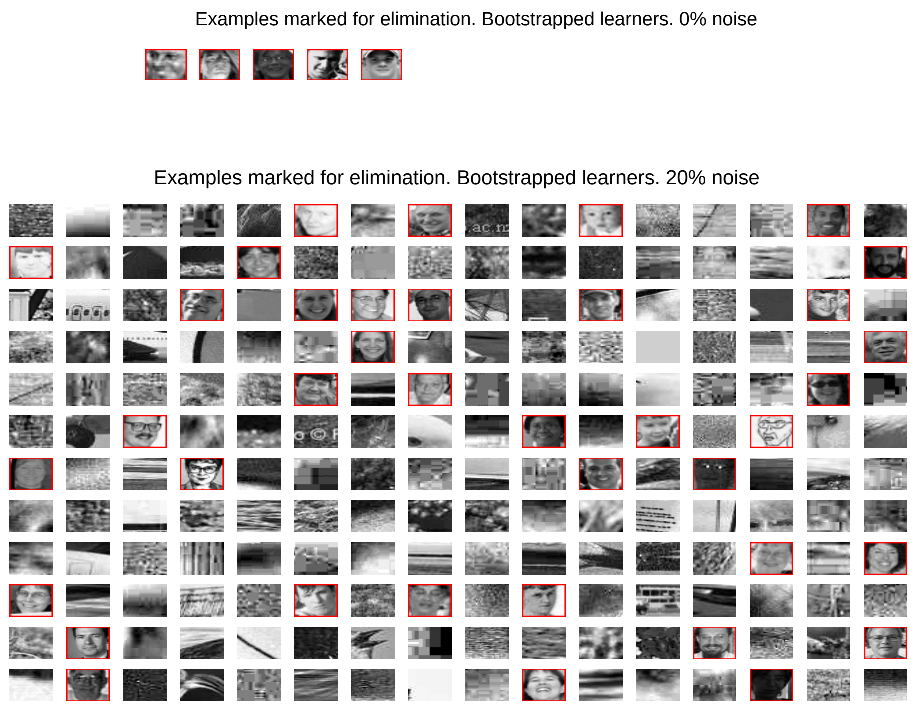
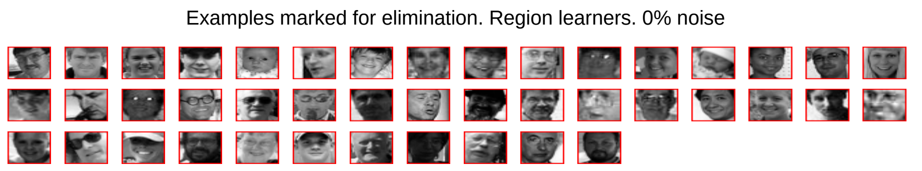

Examples marked for elimination. Bootstrapped learners. 0% noise
Examples marked for elimination. Bootstrapped learners. 20% noise

Figure 4.8: Examples selected by the algorithm for elimination by bootstrapped learners. Dataset with 0% (top) and with 20% (bottom) label noise. Examples with original face label are surrounded by a red box.

Examples which have been determined as wrongly labeled by the bootstrapped learners and by the region learners are shown in figures 4.8 and 4.9, respectively, and statistics of how well the examples with flipped labels are identified is shown in tables 4.2 and 4.3. Both pruning methods can identify many of the examples with flipped labels, except for very noisy datasets. We can notice that the pruning based on bootstrapped learners is more precise in identifying wrongly labeled examples.



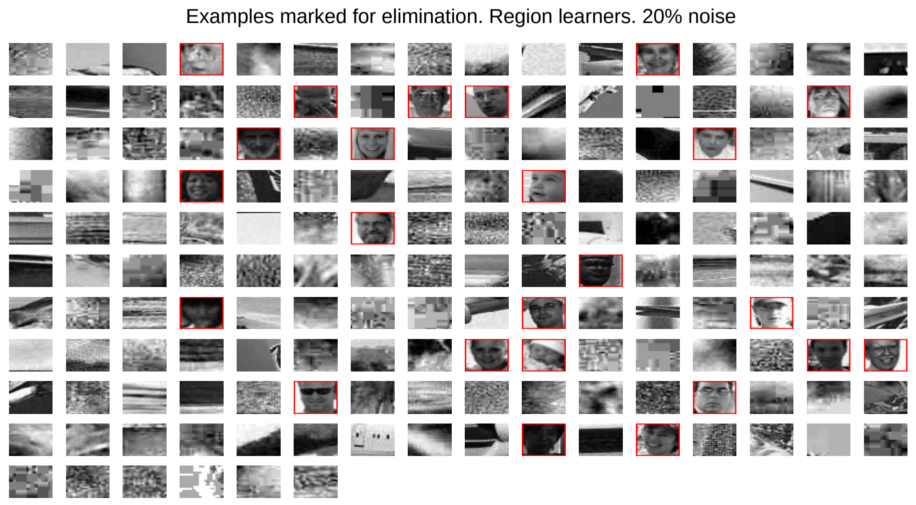


Figure 4.9: Examples selected by the algorithm for elimination by region learners. Dataset with 0% (top) and with 20% (bottom) label noise. Examples with original face label are surrounded by a red box.

The number in the first column in the tables 4.2 and 4.3 is the portion of detected wrongly labeled examples among the ones with artificially flipped labels, whereas the false alarm rate is measured against the whole data. Those numbers should be read with care because they are compared against examples whose labels were artificially flipped, whereas the algorithm may find examples which are otherwise hard to learn and therefore are worth removing. Indeed, some face examples marked for pruning (shown in red boxes in figures 4.9 and 4.8) are quite difficult and should not be present in the training data in the first place. Comparing tables 4.2 and 4.3 we can notice that the bootstrapped based learners are more precise in their decision which examples to eliminate. Apart from removing unreasonably hard examples from the dataset figures 4.8 and 4.9 show that data pruning with both mechanisms is quite successful in identifying examples which have been wrongly labeled.

Table 4.2: Statistics on the identified wrongly labeled examples. Bootstrapped learners.

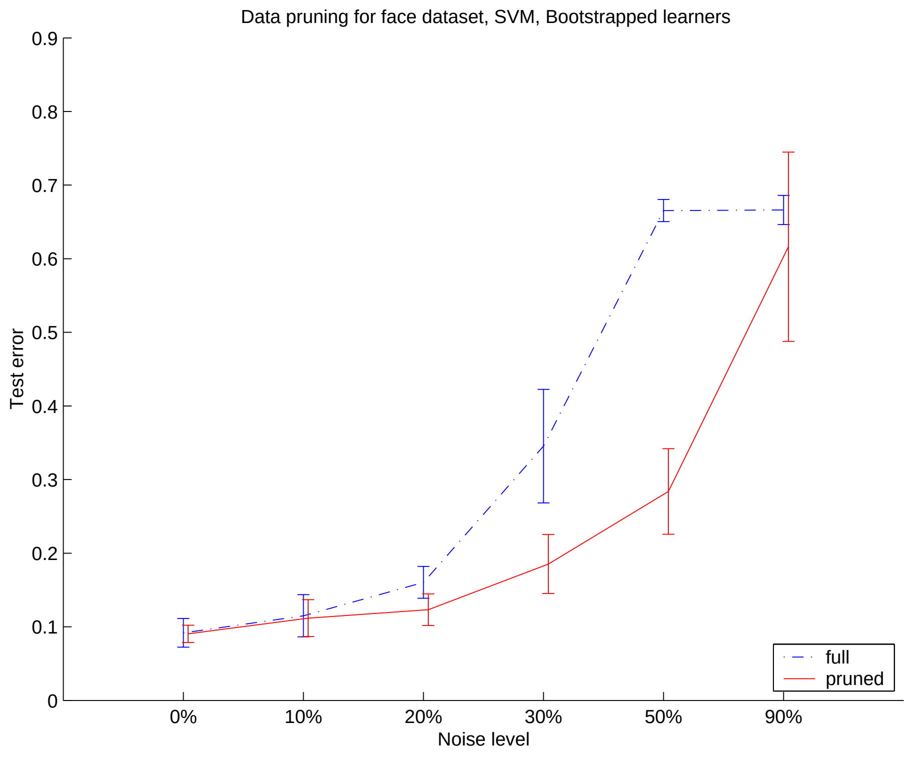
Noise	Identified	FA non-face	FA face
0%	0.0	0.010	0.006
10%	0.737	0.015	0.027
20%	0.736	0.019	0.034
30%	0.700	0.022	0.042
50%	0.620	0.023	0.028
90%	0.280	0.012	0.007

Table 4.3: Statistics on the identified wrongly labeled examples. Region learners.

Noise	Identified	FA non-face	FA face
0%	0.0	0.021	0.034
10%	0.725	0.020	0.028
20%	0.665	0.022	0.022
30%	0.589	0.017	0.020
50%	0.359	0.008	0.009
90%	0.162	0.001	0.001

4.3.3 Pruning very noisy data

In this section we wanted to try the limits of the pruning technique when adding large amounts of noise. The results are seen on figure 4.10 and in tables 4.2 and 4.3. Data pruning reduces the test error significantly for reasonable amount of noise. Not surprisingly, in the presence of large amount of noise and very little signal, e.g. with 90% noise, the pruning method gives very little advantage. In this case we have only 50 correctly labeled face examples among 1000 training examples, therefore very poor signal-to-noise ratio. Moreover, the face examples are quite difficult and probably the algorithm cannot find any consistent projections from this data so that to give reasonable opinion of which examples are wrongly labeled. Still, for relatively large amounts of noise, such as 50%, data pruning methods identify and remove primarily wrongly labeled examples and improve the generalization. This makes the data pruning technique very promising in learning from noisy datasets.



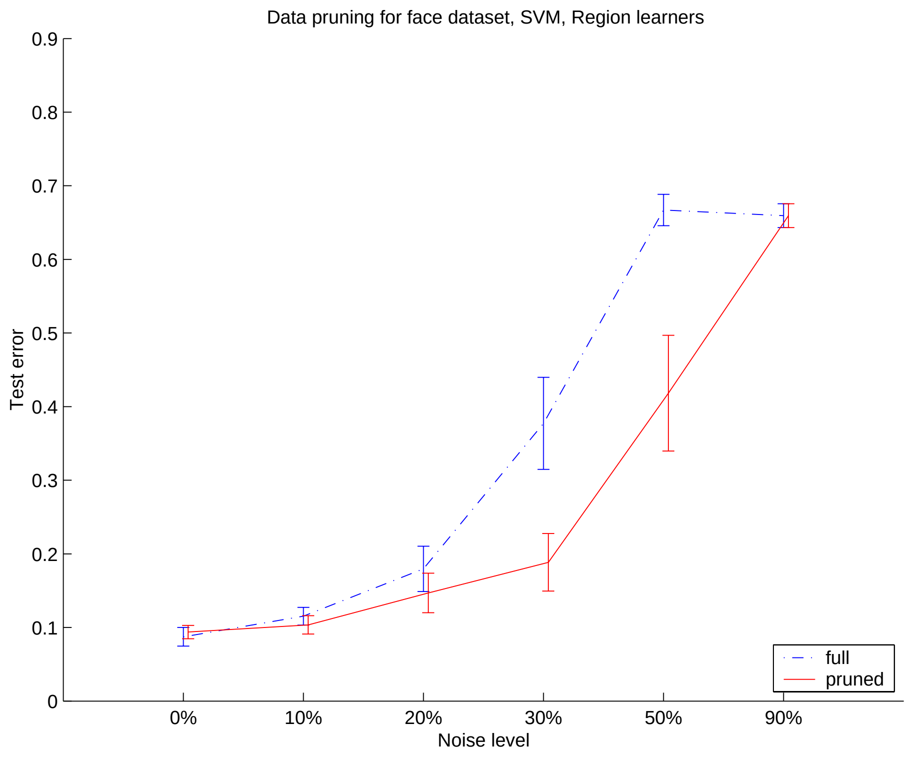


Figure 4.10: Test errors of learning on full and pruned datasets for very noisy data. SVM algorithm. Pruning based on bootstrapped (top) and region learners (bottom).

4.4 Conclusion

Our results show that in learning of object categories with a lot of label noise it pays off if we preprocess the data to automatically remove the wrongly labeled or otherwise difficult examples, prior to training, rather than leave the training algorithm to try to ignore them while learning.

Data pruning for learning face category is demonstrated to work well with both algorithms which are sensitive to noise (AdaBoost) and algorithms which have inherent regularization capabilities (SVM).

The algorithm does not deteriorate the performance when no noise is introduced, instead, it removes some examples which are inherently difficult for learning. Data pruning is very helpful and necessary for high levels of noise.

A future plan is to extend the algorithm to work on non-aligned images of the target category which, for visual categories, would require more involved feature and algorithm selection rather than conceptual change in the data pruning ideas.

Chapter 5

Conclusion

In this work we have shown the usefulness of processing the training data prior to learning, in which identification and elimination of difficult for learning examples is done.

Conversely to methods for learning in the presence of noise, where general penalties are imposed, data pruning advocates direct elimination of difficult to accommodate examples. For, there might be examples which are so hard for the model that they would influence the solution in an adversary way, even though the solution is regularized. Cases in which data pruning is helpful for algorithms which have inherent capabilities of ignoring noisy examples are shown.

Unlike methods which assume a known noise or data model, infinite amount of training data or simply resort to additional training set to do the pruning we restrict our problem to the given training data and do not model explicitly the noise or the data. To our knowledge this is the only work that considers the problem in this more realistic setting.

With this work we have shown that in learning the quality of the examples is also important for better generalization.

Important future direction is to theoretically quantify example difficulty with respect to a model and the influence of troublesome examples in the dataset on the generalization error of a learning model.

The proposed method can be used for acquiring visual category data with very little supervision. For example, images of an object category returned by search engines can be used for training, but they contain a large amount of noise. We demonstrate the usefulness of the approach for very challenging face dataset with large levels of noise. Training on the pruned data demonstrated considerably improved performance compared to training on the full data, especially for large amount of contamination.

An extension of data pruning for object recognition with minimum supervision, namely when the exact object position is not known in the image, would be a very useful and challenging research direction.

Bibliography

[1] Allende, H., Ñanculef, R., Salas, R., 'Robust bootstrapping Neural Networks', MICAI 2004: 813-822

[2] Angluin, D., Laird, P., 'Learning from noisy examples', Machine Learning, 2, 343:370, 1988

[3] Bartlett, P., 'The sample complexity of pattern classification with Neural Networks: the size of the weights is more important than the size of the size of the network', IEEE Transactions on Information Theory, 44, 525-536, 1998

[4] Bartlett, P., Shawe-Taylor, J., 'Generalization performance of Support Vector Machines and other pattern classifiers', in Schölkopf, B., Burges, C., Smola, A. (eds.), 'Advances in Kernel Methods: Support Vector Learning', The MIT Press, 1999

[5] Beckman, R., Cook, R., 'Outlier...s', Technometrics, vol. 25, no. 2, p.119-149, 1983

[6] Bishop, C., 'Neural Networks for pattern recognition', Oxford University Press, 1995

[7] Blake, C. L., Merz, C. J., 'UCI Repository of machine learning databases' [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998

[8] Blum, A., Mitchell, T., 'Combining labeled and unlabeled data with co-training', COLT: Conference on Computational Learning Theory, 1998

[9] Breiman, L., 'Bagging predictors', Machine Learning, 24(2), 123-140, 1996

[10] Breiman, L., 'Randomizing outputs to increase prediction accuracy', Technical report 518, University of California, Berkeley, May 1, 1998.

[11] Caprile, B., Furlanello C., Merler, S., 'Highlighting hard patterns via AdaBoost weights evolution', Multiple Classifier Systems 2002: 72-80

[12] Christensen, S., Sinclair, I., Reed, P., 'Designing committees of models through deliberate weighting of data points', Journal of Machine Learning Research 4, p. 39-66, 2003

[13] Cohn. D., Ghahramani, Z., Jordan, M., 'Active learning with statistical models', Journal of Artificial Intelligence Research 4, p.129-145, 1996

[14] Cortes, C., Vapnik, V., 'Support vector networks', Machine Learning, vol. 20, pp. 273–297, 1995.

[15] Efron, B., Tibshirani, R., 'An introduction to the bootstrap', Chapman and Hall, 1993

[16] Fischler, M., Bolles, R., 'Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography', CACM, 24(6):381-395, June 1981

[17] Forsyth, D., Ponce, J., 'Computer vision: a modern approach', Prentice Hall, 2003

[18] Freund, Y., Schapire, R., 'A decision-theoretic generalization of on-line learning and an application to boosting', European Conference on Computational Learning Theory, 1995

[19] [http://www.google.com]

[20] Grove, A., Schuurmans, D., 'Boosting in the limit: Maximizing the margin of learned ensembles', Fifteenth National Conference on Artificial Intelligence, 1998

[21] Huber, P., 'Robust statistics: A review', The Annals of Mathematical Statistics, vol. 43, no. 4, p.1041-1067, 1972

[22] Jiang, W., 'Is regularization unnecessary for boosting', Proc. Eighth International Workshop on Artificial Intelligence and Statistics, Morgan Kaufmann, January, 2001

[23] Kearns, M., Li, M., 'Learning in the presence of malicious errors', In Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, pages 267-280, Chicago, Illinois, 2-4 May 1988

[24] Koistinen, P., 'Asymptotic theory for regularization: One-dimensional linear case', In Jordan, M., Kearns, M., Solla, S., editors, Advances in Neural Information Processing Systems 10, pages 294-300, 1998.

[25] Kulkarni, S., Mitter, S., Tsitsiklis, J., 'Active learning using arbitrary binary valued queries', Machine Learning, Vol. 11, 1, 23-35, 1993

[26] Littlestone, N., Warmuth, M., 'Relating data compression and learnability', Technical report, University of California Santa Cruz, 1986

[27] Loy, G., Bartlett, P., 'Generalization and the size of the weights: an experimental study', Proceedings of the Eighth Australian Conference on Neural Networks, 60-64, 1997

[28] Mason L., Bartlett, P., Baxter, J., 'Improved generalization through explicit optimization of margins', Machine Learning, 0, 1-11,1999

[29] Mason, L., Baxter, J., Bartlett, P., Frean, M., 'Boosting algorithms as gradient descent in function space', Technical report, Department of Systems Engineering, Australian National University, 1999

[30] Merler, S., Caprile, B., Furlanello, C., 'Bias-variance control via hard points shaving', International Journal of Pattern Recognition and Artificial Intelligence, 2004. In Press.

[31] Murphy, K., Torralba, A., Freeman, B., 'Using the forest to see the trees: a graphical model relating features objects and scenes', Advances in Neural Information Processing Systems 15, 2003

[32] Nicholson, A., 'Generalization Error Estimates and Training Data Valuation', Ph.D. Thesis, California Institute of Technology, 2002

[33] R*ä*tsch, G., Onoda, T., Muller, K, 'Regularizing AdaBoost', Advances in Neural Information Processing Systems 11, p. 564-570, MIT Press, 1999

[34] R*ä*tsch, G., Onoda, T., Muller, K, 'Soft margins for AdaBoost', Machine Learning, p. 1-35, Kluwer Academic Publisher, Boston, 2000

[35] Schapire, R., Freund, Y., Bartlett, P., Lee, W., 'Boosting the margin: A new explanation for the effectiveness of voting methods. The Annals of Statistics, 26(5):1651-1686, 1998

[36] Sch*ö*lkopf, B., Smola, A., 'Learning with kernels', The MIT Press, 2002

[37] Schwartz, D., Samalam, V., Solla, S., Denker, J., 'Exhaustive learning', Neural Computation, 2(2):374-385, 1990

[38] Shawe-Taylor, J., Bartlett, P., Williamson,R., Anthony, M., 'Structural risk minimization over data-dependent hierarchies', Technical Report NC-TR-96-053, Department of Computer Science, Royal Holloway, University of London, Egham, UK, 1996

[39] Tikhonov, A., Arsenin, V., 'Solutions of ill-posed problems', John Wiley & Sons, Washington D.C., 1977

[40] Vapnik, V., 'The nature of statistical learning theory', Springer-Verlag, New York, 1995

[41] Vidyasagar, M., 'Learning and generalization with applications to Neural Networks', Springer-Verlag London Limited, 2003

[42] Viola, P., Jones, M., 'Rapid object detection using a boosted cascade of simple features', Conference on Computer Vision and Pattern Recognition, 2001

[43] Weisberg, S., 'Applied linear regression', John Wiley & Sons, 1985