Maximum Margin Learning Under Uncertainty

Christos Tzelepis

Submitted in partial fulfillment of the requirements of the Degree of Doctor of Philosophy

Supervisors: Dr. Ioannis Patras & Dr. Vasileios Mezaris

School of of Electronic Engineering and Computer Science
Queen Mary University of London
United Kingdom

January 2018

Statement of originality

I, Christos Tzelepis, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Signature: Christos Tzelepis

Date: January 10, 2018

Details of collaboration and publications:

- Tzelepis, Christos, Vasileios Mezaris, and Ioannis Patras, "Linear Maximum Margin Classifier for Learning from Uncertain Data", IEEE Transactions on Pattern Analysis and Machine Intelligence, accepted for publication.
- Mou, Wenxuan, Christos Tzelepis, Vasileios Mezaris, Hatice Gunes, and Ioannis Patras. "Generic to Specific Recognition Models for Membership Analysis in Group Videos." In Automatic Face and Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on, pp. 512-517. IEEE, 2017.
- Tzelepis, Christos, Zhigang Ma, Vasileios Mezaris, Bogdan Ionescu, Ioannis Kompatsiaris, Giulia Boato, Nicu Sebe, and Shuicheng Yan. "Event-based media processing and analysis: A survey of the literature." Image and Vision Computing 53 (2016): 3-19.
- Tzelepis, Christos, Damianos Galanopoulos, Vasileios Mezaris, and Ioannis Patras. "Learning to detect video events from zero or very few video examples." Image and vision Computing 53 (2016): 35-44.
- Tzelepis, Christos, Eftichia Mavridaki, Vasileios Mezaris, and Ioannis Patras. "Video aesthetic quality assessment using kernel Support Vector Machine with isotropic Gaussian sample uncertainty (KSVM-IGSU)." In Image Processing (ICIP), 2016 IEEE International Conference on, pp. 2410-2414. IEEE, 2016.
- Tzelepis, Christos, Vasileios Mezaris, and Ioannis Patras. "Video event detection using kernel support vector machine with isotropic gaussian sample uncertainty (KSVM-iGSU)." In International Conference on Multimedia Modeling, pp. 3-15. Springer, Cham, 2016.
- Markatopoulou, Fotini, Anastasia Moumtzidou, Christos Tzelepis, Kostas Avgerinakis, Nikolaos Gkalelis, Stefanos Vrochidis, Vasileios Mezaris, and Ioannis Kompatsiaris. "ITI-CERTH participation to TRECVID 2013." In TRECVID 2013 Workshop, Gaithersburg, MD, USA. 2013.
- Tzelepis, Christos, Nikolaos Gkalelis, Vasileios Mezaris, and Ioannis Kompatsiaris. "Improving event detection using related videos and relevance degree support vector machines." In Proceedings of the 21st ACM international conference on Multimedia, pp. 673-676. ACM, 2013.

Abstract

In this thesis we study the problem of learning under uncertainty using the statistical learning paradigm. We first propose a linear maximum margin classifier that deals with uncertainty in data input. More specifically, we reformulate the standard Support Vector Machine (SVM) framework such that each training example can be modeled by a multi-dimensional Gaussian distribution described by its mean vector and its covariance matrix – the latter modeling the uncertainty. We address the classification problem and define a cost function that is the expected value of the classical SVM cost when data samples are drawn from the multi-dimensional Gaussian distributions that form the set of the training examples. Our formulation approximates the classical SVM formulation when the training examples are isotropic Gaussians with variance tending to zero. We arrive at a convex optimization problem, which we solve efficiently in the primal form using a stochastic gradient descent approach. The resulting classifier, which we name SVM with Gaussian Sample Uncertainty (SVM-GSU), is tested on synthetic data and five publicly available and popular datasets; namely, the MNIST, WDBC, DEAP, TV News Channel Commercial Detection, and TRECVID MED datasets. Experimental results verify the effectiveness of the proposed method. Next, we extended the aforementioned linear classifier so as to lead to non-linear decision boundaries, using the RBF kernel. This extension, where we use isotropic input uncertainty and we name Kernel SVM with Isotropic Gaussian Sample Uncertainty (KSVM-iGSU), is used in the problems of video event detection and video aesthetic quality assessment. The experimental results show that exploiting input uncertainty, especially in problems where only a limited number of positive training examples are provided, can lead to better classification, detection, or retrieval performance. Finally, we present a preliminary study on how the above ideas can be used under the deep convolutional neural networks learning paradigm so as to exploit inherent sources of uncertainty, such as spatial pooling operations, that are usually used in deep networks.

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisors Dr Ioannis Patras and Dr Vasileios Mezaris for the continuous support of my PhD study, their motivation and patience, but mostly for the great opportunity they gave me to start my postgraduate studies. Besides my main supervisors, I would like to thank the rest of the members of my supervisory team, Dr Tao Xiang and Dr Fabrizio Smeraldi, for their insightful guidance during conducting my thesis. I owe special thanks to Dr Anastasios Delopoulos and Dr Christos Diou who supervised my diploma thesis in the School of Electrical and Computer Engineering, Aristotle University of Thessaloniki, in 2011. I thank them for their constant support, wise advice, and our always-stimulating discussions since then. Last but not the least, I would like to thank my family, my mother and sister, for supporting me spiritually throughout this challenging period. And Fenia, for making this possible by exhibiting remarkable tolerance. It has been vital to have her by my side.

I never expected hell to have so much light.

Miltos Sachtouris (1919-2005)[1]

[1] Miltos Sachtouris was a Greek poet who has created, through the development of a style as spare and lucid as Baudelaire's, a surrealist world of ordinary horror, where the most bizarre flowerings of intolerable anxiety unfold with dreamlike clarity at your elbow as you walk down the street.

Contents

List of Figures

List of Tables

Nomenclature

$\mathbb{R}$	the set of real numbers
$\mathbb{R}_+$	the set of non-negative real numbers
$\mathbb{R}^n$	the n-dimensional Euclidean space of column vectors
$\mathbb{R}^{m\times n}$	the space of $m \times n$ matrices with real entries
$\mathbb{S}^n_{++}$	the space of symmetric positive definite $n \times n$ matrices with real entries
a	a scalar that typically belongs to $\mathbb{R}$
$\mathbf{x}$	a column vector that typically belongs to $\mathbb{R}^n$
A	a matrix that typically belongs to $\mathbb{R}^{m\times n}$
I_n	the identity matrix of order n
$\mathcal{N}(\mu, \sigma^2)$	the uni-variate normal distribution with mean value $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}_+$
$\mathcal{N}(0, 1)$	the uni-variate standard normal distribution
$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	the multi-variate normal distribution with mean vector $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\Sigma \in \mathbb{S}^n_{++}$
$\mathcal{N}(\mathbf{0}, I_n)$	the n-dimensional standard normal distribution

List of abbreviations

SLT	Statistical Learning Theory
SVM	Support Vector Machine
SVM-GSU	Support Vector Machine with Gaussian Sample Uncertainty
SVM-iGSU	Support Vector Machine with Isotropic Gaussian Sample Uncertainty
RD-SVM	Relevance Degree SVM
RD-KSVM-iGSU	Relevance Degree Kernel SVM-iGSU
RBF	Radial Basis Function
SGD	Stochastic Gradient Descent
PDF	Probability Density Function
CDF	Cummulative Distribution Function
DL	Deep Learning
DCNN	Deep Convolutional Neural Network

Chapter 1

Introduction

Contents

1.1 Learning under uncertainty

Uncertainty is ubiquitous in almost all fields of scientific studies [66, 31], which it is roughly divided into two general categories: *aleatory uncertainty* and *epistemic uncertainty*. While aleatory uncertainty refers to the inherent randomness in nature, derived from natural variability of the physical world (e.g., random show of a flipped coin), epistemic uncertainty origins from human's lack of knowledge of the physical world, as well as ability of measuring and modeling the physical world. Uncertainty distinguishes from certainty in the degree of belief or confidence. If certainty is referred to as a perception or belief that a certain system or phenomenon can experience or not, uncertainty indicates a lack of confidence or trust in an article of knowledge or decision [127]. The US National Research Council [20] gave the following general definition of uncertainty: *"Uncertainty is a general concept that reflects our lack of sureness about something or someone, ranging from just short of complete sureness to an almost complete lack of conviction about an outcome"*. In this thesis, however, we use the term of uncertainty as referred to by the dominant probability theory [60, 50, 101].

Concerning the supervised machine learning field, to which this thesis lies, uncertainty has been studied in many different aspects [24, 7, 54]. More specifically, the research community has studied learning problems where uncertainty is present either in the labels or in the representation of the training data. The former case concerns the problem where the truth labels of the training examples of a supervised learning problem are not certain or are corrupted. The latter case, to which this thesis falls, concerns the problem of supervised learning using data representations that are not

certain. Examples of such problems include image classification, video event detection, emotional analysis using electroencephalogram signals, etc.

In this thesis, we focus on the supervised learning problem by introducing input uncertainty under the Statistical Learning Theory (SLT) paradigm. SLT, established by Vladimir Vapnik [111] more than three decades ago, aims at providing a framework for studying –using a statistical framework– the problem of inference; that is, the problem of gaining knowledge from data and making decisions about them. Vapnik once said that *nothing is more practical than a good theory*. Indeed, the SLT paradigm has been proven to be an extremely practical for decades. The Support Vector Machine (SVM), probably the most popular learning algorithm that is based on SLT, has been shown to be very powerful for pattern classification, among other relevant tasks. Vapnik established the standard linear regularized SVM algorithm for computing a linear discriminative function that optimizes the margin between the so-called support vectors and the separating hyperplane. Despite the fact that the standard linear SVM algorithm is a well-studied and general framework for statistical learning analysis, it is still an active research field (e.g., [99, 94]).

Although SVM has been shown to be a powerful learning paradigm, its classical formulation, as well as the majority of classification methods, do not explicitly model input uncertainty. In standard SVM, each training datum is a vector, whose position in the feature space is considered certain. This does not model the fact that measurement inaccuracies or artifacts of the feature extraction process contaminate the training examples with noise. In several cases the noise distribution is known or can be modeled; e.g., there are cases where each training example represents the average of several measurements or of several samples whose distribution around the mean can be modeled or estimated. Finally, in some cases it is possible to model the process by which the data is generated, for example by modeling the process by which new data is generated from transforms applied on an already given training dataset.

1.2 Problem definition – Challenges and Assumptions

The main problem we wish to address in this thesis is the classical supervised classification problem using input uncertainty. The majority of the learning methods (e.g., those employed in video understanding and indexing applications, such as the video event detection and aesthetic quality video assessment problems) do not address the uncertainty in the training data explicitly. In these problems each training example is typically described by a fixed position in some vector space (feature representation). However, such an approach does not account for the fact that the underlying process of extracting the feature representation may be imperfect or noisy, hence introducing some degree of uncertainty to the generated features.

For this, we model input uncertainty in training example level using the well-studied and ubiquitously-used multi-variate Gaussian distribution. More specifically, we define an input training entity/example[1] as a multi-variate Gaussian distribution, or, equival-

[1]We choose to interchangeably use the term *training example* or *training entity* over the usually

ently, as a pair of a mean vector and a covariance matrix (since a Gaussian distribution is defined uniquely by its first- and second-order statistics). Each training example is allowed to have a distinct covariance matrix. Consequently, we formally define the problem we address as follows: *Given an annotated set of Gaussian distributions, we optimize for the soft margin using the expected value of the hinge loss, where the expectation is taken under the given Gaussians.*

For developing our basic learning algorithm we choose to use and extend a very popular maximum-margin learning algorithm, the soft-margin linear Support Vector Machine (SVM). More specifically, we extend the standard linear SVM using the standard hinge loss function and optimize for the soft margin using the *expected* value of the hinge loss. This essentially means that the loss that is potentially introduced by a training example is measured not solely using a single feature vector (e.g., the mean feature vector of the input Gaussian), but rather using the knowledge of the discrepancy of the respective distribution, i.e., its (co)variance.

The fundamental assumption that we made during the development of our learning algorithms concerns the normality of the input uncertainty. That is, as discussed above, each training entity is considered as a multi-variate Gaussian distribution. Modeling the uncertainty of each training example as above comes with a number of virtues. More specifically:

- In a process (e.g., feature extraction) where there is limited knowledge of the factors that may introduce uncertainty (usually not even the number of possible noise or error sources), a Gaussian distribution is expected to serve as a good model, since it is the limit of the sum of a large number of unknown (but reasonably bounded) uncertainty sources (Central Limit Theorem).

- A Gaussian distribution is completely described by its first- and second-order statistics. This is a significant advantage of the Gaussian, which is absent in other random distributions. Measuring or modeling mean and variance is relatively easy compared to measuring higher-order moments (which may be necessary if the random vectors are not Gaussian).

- Mathematical manipulation of Gaussian provides greater convenience compared to other, even conceptually simpler, distributions. For instance, even in the case of the multivariate uniform distribution, which is conceptually simpler than an anisotropic Gaussian, the analytical evaluation of the loss introduced by an input random vector, i.e., the intersection between the n-ball and a hyperplane, introduces extra complexity, which also makes the differentiation more strenuous than the Gaussian ellipsoidal case (where tails decay exponentially).

- Finally, compared to other distributions that are inherently isotropic over the input dimensions, or rigid over input dimensions, a Gaussian distribution provides

used *training sample* so as not to cause confusion between the term *sample*, which is typically used for a datum drawn from a distribution, and the training entity of our algorithm, which is itself a Gaussian distribution.

flexibility in modeling anisotropic uncertainty for arbitrary number of input dimensions. That is, one can model the anisotropic uncertainty solely on a set of input dimensions of interest. That is, one can introduce constraints on the covariance matrices, such as them being diagonal, block diagonal, or multiples of the identity matrix. In this way one can model different types of uncertainty.

For the above reasons, we chose to adopt the multi-variate Gaussian distribution as the most appropriate candidate for modeling input uncertainty.

The challenges that arose during the development of our learning algorithms, along with the solutions that we proposed for addressing them, are described below. First of all, the introduction of input uncertainty in the loss function of the classifier should be done analytically, in contrast to other relevant methods that address the problem by generating a large number of uniformly distributed points that lie in each input Gaussian's ellipsoid, and computing the ratio of the number of points on the wrong side of the hyperplane to the total number of generated points. Instead, we addressed the problem by analytically computing the loss introduced by each input Gaussian (Appendix A), arriving at a closed-form loss along with its derivatives with respect to the optimization parameters. In addition, we proved (in Appendix B) that the above loss is convex with respect to the optimization parameters and, thus, we were allowed to obtain the global optimal solution using an appropriate iterative gradient descent algorithm that is linear with respect to the number training data. For this purpose, we modified and used a popular SGD algorithm, namely the Pegasos algorithm [96].

Next, an important challenge that is inherent in our method is the modeling or the estimation of input uncertainty, since this amount of information is considered as prior knowledge; that is, it is data-driven and is not computed or estimated during training. In general, modeling of the uncertainty is a domain- and/or dataset-specific problem, and in this respect, similarly to all of the other methods in the literature that model/use uncertainties, we do not offer a definitive answer on how this can or should be done on any existing dataset. We addressed this important issue by providing a number of methodologies that mainly depend on the specific problems, but can also be applied in similar problems/datasets.

Extending the proposed linear classifier so as to result in non-linear boundaries was a special challenging task that needed to be addressed. This is because the well-known *kernel trick* [105], which is used –among other cases– in the kernelization of the standard SVM cannot be applied directly. In the classical SVM formulation (in its dual form), the training data appear only through inner products and, thus, the application of the kernel trick is straight-forward. That is, a so-called kernel function is first used in order to map input data in a higher-, even infinite-dimensional (implicit) feature space, where one does not need to explicitly compute the aforementioned inner products, but only to evaluate the kernel function on the various combinations of the input data. In our case, where we work in the primal form, in the general case of anisotropic covariance matrices, training data do not appear in the objective function only through inner products, and thus the kernel trick could be applied directly. To resolve this limitation, we needed to assume only isotropic input covariance matrices.

Then, we proceeded to the kernel version of our classifier by appealing to a semi-parametric version [93] of the representer theorem [57].

Finally, we attempt to investigate the application of the idea of exploiting input uncertainty under the Deep Convolutional Neural Network (DCNN) framework. The renaissance of convolutional neural networks (CNNs) has given rise to sophisticated architectures for Computer Vision problems producing state-of-the-art results compared to other learning paradigms. Most of the existing CNN architectures for image classification use a set of structured convolution layers, a pooling operation before the classification layer, and they typically employ the cross-entropy loss function for measuring the miss-classification cost. While pooling has proven to be extremely effective by reducing input dimensionality and providing feature maps robust to spatial transformations, it also results to loss of information as it summarizes the responses with the pooling area by their mean (or max value).

1.3 Contributions

In this section we list the main contributions of the thesis. In the first main chapter (Chapter 3), we develop the linear variant of the proposed classifier, i.e., the linear SVM with Gaussian Sample Uncertainty (LSVM-GSU). The main contributions of that chapter can be listed as follows:

- We generalize the standard soft-margin linear SVM algorithm so as input training data are given as multi-variate Gaussian distributions (in the form of mean vector-covariance matrix pairs). Each input Gaussian distribution corresponds to an annotated training entity. We arrive at our loss function, which is based on the expectation of the hinge loss, and its derivatives in closed forms.

- We prove that the aforementioned loss function is convex with respect to the optimization parameters. Thus, the global optimal solution, i.e., the optimal separating hyperplane is guaranteed to be obtained using the appropriate optimization algorithm. For this, we modify and use a popular SGD algorithm, namely the Pegasos algorithm [96] for solving our optimization problem. This allows for great scalability (its complexity is linear to the number of training data).

- It is worth noting that one would arrive at the same decision border with the classical SVM trained on a dataset containing samples drawn from the Gaussians in question, as the number of samples tend to infinity. However, we show that as the dimensionality of the input space increases, one needs to generate more samples from the Gaussians in order to preserve a desired approximation of the loss and, thus, of the optimal decision function. We also show that for spaces of high dimensionality the number of samples needed can be prohibitively high. In addition, our method degenerates to a classical SVM in the case that all of the Gaussians are isotropic with a variance that tends to zero.

- We propose a linear subspace learning approach in order to address the situation where most of the mass of the training Gaussians lie in a low dimensional manifold that can be different for each Gaussian and subsequently solve the problem in lower-dimensional spaces.

- An important contribution of our basic linear learning algorithm is that, in contrast to previous works that model uncertainty in the SVM framework either by considering isotropic noise or by using expensive sampling schemes to approximate their loss functions, our formulation allows for full covariance matrices that can be different for each example. This allows dealing, among others, with cases where the uncertainty of only a few examples, and/or the uncertainty along only a few of their dimensions, is known or modeled. In the experimental results section we show several real-world problems in which such modeling is beneficial. More specifically, we show cases, in which the variances along (some) of the dimensions are part of the dataset – this includes medical data where both the means and the variances of several measurements are reported, and large scale video datasets, where the means and the variances of some of the features that are extracted at several time instances in the video in question are reported. We then show a case in which means and variances are a by-product of the feature extraction method, namely the Welch method for extracting periodograms from temporal EEG data. And finally, we show a case in which, for an image dataset (MNIST) we model the distribution of images under small geometric transforms as Gaussians, using a first-order Taylor approximation to arrive in an analytic form. In particular, the Taylor expansion method (Appendix B) that is behind the modeling used in Sect. 3.5.2, has been used to model the propagation of uncertainties due to a feature extraction process in other domains; for instance, in [26] (Sect. II.B) this is used to model as Gaussian the uncertainty in the estimation of illumination invariant image derivatives.

In the following chapter (Chapter 4), we proceed to the kernelization of our linear classifier, using the popular RBF kernel function. In this chapter, we assume isotropic input uncertainty, i.e., we assume that each training example is described by a (distinct) multi-variate Gaussian distribution with a covariance matrix that is a multiple of the identity matrix, and we arrive at Kernel SVM with Isotropic Gaussian Sample Uncertainty (KSVM-iGSU). The main contributions of that chapter can be listed as follows:

- We recast the optimization problem of the linear SVM-GSU into a variational calculus problem; that is, the original optimization problem is rewritten as a problem of minimizing an equivalent (objective) functional, and, thus, instead of looking for a separating hyperplane (i.e., its parameters) in the original input feature space, we look for a minimizer function that lives in a richer, higher-dimensional (in our case infinite-dimensional) space. We prove that the above functional is such that its minimizer can be represented as a finite linear combination of kernel products (in our case using the RBF kernel function). Additionally,

due to the convexity of our objective functional, we can efficiently solve the problem using an appropriate SGD algorithm (similarly to the linear case), i.e., the Pegasos algorithm [96].

- We combine KSVM-iGSU with the previously proposed Relevance Degree SVM (RD-SVM) [108, 107]. In RD-SVM each training example is associated with a confidence value (called relevance degree) indicating the degree of relevance of the respective training example with the class that it is related. This is essentially a method that handles uncertainty in the truth labels, and combined with the proposed KSVM-iGSU provide a methodology for handling uncertainty both in label and feature representation.

- We apply the above kernel classifiers in two challenging multimedia understanding problems, namely the video event detection and aesthetic quality video assessment. Especially in the visual understanding domain, the majority of the learning methods employed in video understanding and indexing applications do not address the uncertainty in the training data explicitly. That is, firstly, each training example is assumed to be described by a fixed position in some vector space (feature representation). However, such an approach does not account for the fact that the underlying process of extracting the feature representation may be imperfect or noisy, hence introducing some degree of uncertainty to the generated features. Secondly, each training example is typically annotated with a binary ground-truth label. This is essentially the result of a quantization process, where different pieces of data that may be perfect or not-so-perfect examples of a class have to be assigned a binary label, and this inevitably introduces some form of quantization error. For instance, in some cases the annotation process could naturally lead to three types of labels, i.e., positive, negative, and "near-miss" or "related", the latter expressing the fact that the example is closely related with the positive class but does not meet the exact requirements for being characterized as a positive instance; yet, for training a binary classifier, these annotations need to be subsequently quantized to just two classes: positive and negative. Similarly, in many cases the ground-truth annotation of training data is carried out by a number of experts who decide on the label of each given example, and a final binary assignment of each sample to the positive or negative class is made by averaging and binarizing, or aggregating in another similar way, the responses of the different annotators. In both the above examples, the ground truth annotation process endows every binary annotation with some level of confidence on it, but this is typically ignored in the subsequent training of a binary classifier. In thesis, we address the above using the proposed KSVM-iGSU and RD-KSVM-iGSU.

Finally, in the last main chapter of this thesis (Chapter 5), we apply the idea of introducing and exploiting input uncertainty in the DCNN framework. For this purpose we choose a state-of-the-art architecture, namely the Wide Residual Network (WRN) [124], which has been proven to be very successful in image classification tasks. More specifically, we try to address this by a) modifying the last pooling layer of a

CNN so as to compute both the first- and second-order statistics of its output, and b) modifying a maximum-margin loss function (i.e., the squared hinge loss) so as to take the aforementioned uncertainty into account during training. The proposed modifications can be applied to all networks that have pooling and classification layers at the last stages (i.e., most state of the art methods, including AlexNet, VGG, Inception, ResNet, and WRN) – the resulting network has practically the same training/testing time complexity, and no additional parameters at test time. We apply the proposed method in a state-of-the-art CNN architecture, i.e., the Wide Residual Network and propose an efficient approach for training the proposed maximum-margin classification layer. Experimental results show that using the maximum-margin hinge loss function improves classification results in comparison to soft-max layer and that the proposed methods for exploiting uncertainty increases further the classification accuracy.

1.4 Outline of the thesis

The rest of the thesis is structured as follows. We start by discussing related works in Chapter 2. We follow by introducing the linear variant of the proposed learning algorithm, i.e., the Linear SVM with Gaussian Sample Uncertainty (LSVM-GSU) in Chapter 3, where we also present the experimental evaluation of the proposed classifier on synthetic data and five publicly available and popular datasets; namely, the MNIST, WDBC, DEAP, TV News Channel Commercial Detection, and TRECVID MED datasets. Chapter 4 deals with extending the proposed linear classifier in its RBF-kernelized version, i.e., the kernel SVM with Isotropic Gaussian Sample Uncertainty (KSVM-iGSU) Moreover, in this chapter we combine the proposed kernel classifier (KSVM-iGSU) with the previously proposed Relevance Degree SVM (RD-SVM) for handling training examples with variable reliability in their truth labels. We evaluate the proposed kernel classifiers in the problems of video event detection and video aesthetic quality assessment. In Chapter 5, we provide a preliminary study on the extension of the ideas of exploiting input data uncertainty in the so-called deep learning (DL) framework; more specifically in the Deep Convolutional Neural Networks (DCNNs) for the problem of image classification. Finally, we draw our conclusions in Chapter 6.

Chapter 2

Related work

Contents

In this chapter we briefly review the works that are closely related to the developments of this thesis. More specifically, we will first discuss works that study the problem of supervised learning under uncertainty. That is, uncertainty either in the training labels, or in the feature representation. However, we will mostly focus on the latter category, since this is the case where this thesis falls. Then, since this thesis, among others, also addresses two challenging visual understanding problems, i.e., video event detection and video aesthetic quality assessment, we will briefly discuss the relevant literature. Finally, we will briefly review the literature of deep convolutional neural networks (DCNNs) for image classification, since we will attempt to apply the idea of introducing and exploiting input uncertainty in the training of a DCNN for image classification.

2.1 Learning under uncertainty

Uncertainty is ubiquitous in almost all fields of scientific studies [66]. Exploiting uncertainty in learning has been studied in many different aspects [24, 7, 54]. More specifically, the research community has studied learning problems where uncertainty is present either in the labels or in the representation of the training data.

Jaakkola and Haussler [48] introduced the Fisher kernel (named in honour of Sir Ronald Fisher), in an attempt to create a generic mechanism for incorporating generative probability models into discriminative classifiers such as SVMs. The Fisher kernel combines the benefits of generative and discriminative approaches to pattern classification by deriving a kernel from a generative model of the data. In brief, it consists in characterizing a sample by its deviation from the generative model. The

deviation is measured by computing the gradient of the sample log-likelihood with respect to the model parameters. This leads to a vectorial representation which we call Fisher Vector (FV). In the image classification case, the samples correspond to the local patch descriptors and we choose as generative model a Gaussian Mixture Model (GMM) which can be understood as a "probabilistic visual vocabulary".

Moreover, learning distance or similarity metrics has been an emerging field in machine learning, with various applications in computer vision. The goal of metric learning algorithms is to take advantage of prior information in form of labels over simpler though more general similarity measures. A particular class of distance functions that exhibits good generalization performance for many machine learning problems is Mahalanobis metric learning. The goal is to find a global, linear transformation of the feature space such that relevant dimensions are emphasized while irrelevant ones are discarded. As there exists a bijection between the set of Mahalanobis metrics and the set of multivariate Gaussians one can think of it in terms of the corresponding covariance matrix. The metric adapts to the desired geometry by arbitrary linear rotations and scalings. After projection the plain Euclidean distance is measured. In [59], Koestinger et al. proposed KISS (Keep It Simple and Straightforward) metric learning method. The authors proposed to learn a distance metric from equivalence constraints. Based on a statistical inference perspective they provide a solution that is very efficient to obtain and effective in terms of generalization performance. For this purpose, they considered two independent generation processes for observed commonalities of similar and dissimilar pairs. The dissimilarity is defined by the plausibility of belonging either to one or the other.

In [71], Liu and Tao studied a classification problem in which sample labels are randomly corrupted. In this scenario, there is an unobservable sample with noise-free labels. However, before being observed, the true labels are independently flipped with a probability $p \in [0, 0.5)$, and the random label noise can be class-conditional. Tzelepis et al. [108, 107] proposed an SVM extension where each training example is assigned a relevance degree in $(0, 1]$ expressing the confidence that the respective example belongs to the given class. Li and Sethi [65] proposed an active learning approach based on identifying and annotating uncertain samples. Their approach estimates the uncertainty value for each input sample according to its output score from a classifier and selects only samples with uncertainty value above a user-defined threshold. In [92], the authors used weights to quantify the confidence of automatic training label assignment to images from clicks and showed that using these weights with Fuzzy SVM and Power SVM [126] can lead to significant improvements in retrieval effectiveness compared to the standard SVM. Finally, the problem of confidence-weighted learning is addressed in [21, 29, 46], where uncertainty in the weights of a linear classifier (under online learning conditions) is taken into consideration.

Assuming uncertainty in data representation has also drawn the attention of the research community in recent years. Different types of robust SVMs have been proposed in several recent works. Bi and Zhang [12] considered a statistical formulation where the input noise is modeled as a hidden mixture component, but in this way the "iid" assumption for the training data is violated. In that work, the uncertainty is

modeled isotropically. Second order cone programming (SOCP) [2] methods have also been employed in numerous works to handle missing and uncertain data. In addition, Robust Optimization techniques [6, 9] have been proposed for optimization problems where the data is not specified exactly, but it is known to belong to a given uncertainty set $\mathcal{U}$, yet the optimization constraints must hold for all possible values of the data from $\mathcal{U}$.

Lanckriet et al. [63] considered a binary classification problem where the mean and covariance matrix of each class are assumed to be known. Then, a minimax problem is formulated such that the worst-case (maximum) probability of misclassification of future data points is minimized. That is, under all possible choices of class-conditional densities with a given mean and covariance matrix, the worst-case probability of misclassification of new data is minimized.

Shivaswamy et al. [97], who extended Bhattacharyya et al. [11], also adopted a SOCP formulation and used generalized Chebyshev inequalities to design robust classifiers dealing with uncertain observations. In their work uncertainty arises in ellipsoidal form, as follows from the multivariate Chebyshev inequality. This formulation achieves robustness by requiring that the ellipsoid of every uncertain data point should lie in the correct halfspace. The expected error of misclassifying a sample is obtained by computing the volume of the ellipsoid that lies on the wrong side of the hyperplane. However, this quantity is not computed analytically; instead, a large number of uniformly distributed points are generated in the ellipsoid, and the ratio of the number of points on the wrong side of the hyperplane to the total number of generated points is computed.

Several works [11, 97, 63] robustified regularized classification using box-type uncertainty. By contrast, Xu et al. [119, 120] considered the robust classification problem for a class of non-box-typed uncertainty sets; that is, they considered a setup where the joint uncertainty is the Cartesian product of uncertainty in each input. This leads to penalty terms on each constraint of the resulting formulation. Furthermore, Xu et al. gave evidence on the equivalence between the standard regularized SVM and this robust optimization formulation, establishing robustness as the *reason* why regularized SVMs generalize well.

In [87], motivated by GEPSVM [77], Qi et al. robustified a twin support vector machine (TWSVM) [56]. Robust TWSVM [87] deals with data affected by measurement noise using a SOCP formulation. In their work, the input data is contaminated with isotropic noise (i.e., spherical disturbances centred at the training examples), and thus cannot model real-world uncertainty, which is typically described by more complex noise patterns. Power SVM [126] uses a spherical uncertainty measure for each training example. In this formulation, each example is represented by a spherical region in the feature space, rather than a point. If any point of this region is classified correctly, then the corresponding loss introduced is zero.

2.2 Exploiting uncertainty in multimedia understanding problems

High-level video event detection is concerned with determining whether a certain video depicts a given event or not [129, 30, 70]. Typically, a high-level (or complex) event is defined as an interaction among humans, or between humans and physical objects. Some typical examples of complex events are those provided in the Multimedia Event Detection (MED) task of the TRECVID benchmarking activity [84]. For instance, indicative complex events defined in MED 2014 include "Attempting a bike trick", "Cleaning an appliance", or "Beekeeping", to name a few. There are many works dealing with event detection in video (e.g., [129, 30, 70, 18, 28, 35, 42, 41, 51, 52, 67, 39, 79, 13, 100]), several of them being developed in the context of the TRECVID MED task. Despite the attention that video event detection has received, though, there is only a limited number of studies that have explicitly examined the problem of learning event detectors from very few (e.g. 10) positive training examples [41],[108], and developed methods for addressing this exact problem. In [41], for instance, Habibian et al. present VideoStory, a video representation scheme for learning event detectors from a few training examples by exploiting freely available Web videos together with their textual descriptions. Several other works (e.g. [13])) treat the few-example problem in the same way that they deal with event detection when more examples are available (e.g. training standard kernel SVMs). Learning video event detectors from a few examples is a problem that is simulated in the TRECVID MED task [84] by the 10Ex subtask, where only 10 positive samples are available for training.

In the case of learning from very few positive samples, it is of high interest to further exploit video samples that do not exactly meet the requirements for being characterized as true positive examples of an event, but nevertheless are closely related to an event class and can be seen as "related" examples of it. This is simulated in the TRECVID MED task [84] by the "near-miss" video examples provided for each target event class. Except for [108], none of the above works takes full advantage of these related videos for learning from few positive samples; instead, the "related" samples are either excluded from the training procedure [39],[13], or they are mistreated as true positive or true negative instances [28]. In contrast, in [108] the authors exploit related samples by handling them as weighted positive or negative ones, applying an automatic weighting technique during the training stage. To this end, a relevance degree in $(0, 1]$ is automatically assigned to all the related samples, indicating the degree of relevance of these observations with the class they are related to. It was shown that this weighting resulted in learning more accurate event detectors.

Video aesthetic quality assessment is about the automatic assessment of a given video's aesthetic value and the corresponding ranking of the videos within a dataset, such that videos of higher aesthetic value can be ranked higher. For this problem, only a few methods have been proposed so far. The first methods in this domain tried to estimate the videos' aesthetic value by extracting mostly low-level features from video frames. For instance, in [81], a set of low-level features, such as sharpness, colorfulness, luminance and blockiness quality, and a few motion features, are extracted.

Then, a SVM using the Radial Basis Function (RBF) kernel is trained for assessing the aesthetic quality of videos. In [76], the authors treat the video as a sequence of still images to from which they extract a set of visual-based features together with two additional motion-based features, i.e., the length of subject region motion and motion stability, so as to distinguish professional videos from amateurish ones. They also tried different learning approaches, such as kernel SVM, Bayesian classification, and Gentle AdaBoost. A more elaborate method that introduces a set of features ranging from low- and mid-level attributes to high-level style descriptors, combined with a kernel SVM learning stage, is presented in [114]. In [121], an RBF kernel SVM is applied to a set of "semantically independent" features, such as camera motion and stabilization, and frame composition, along with a set of "semantically dependent" features, such as motion direction entropy, color saturation, and lightness. Semantic dependency of a feature, according to [121], refers to whether this feature relates or not to the semantic content of each frame. Moreover, in [10], low- and high-level visual and motion features are extracted at cell-, frame-, and shot-level and a Low Rank Late Fusion (LRLF) scheme is used for fusing the scores produced by a set of SVMs, each of which was trained with one specific aesthetic feature. More motion features are introduced in [122], where the authors evaluate the effectiveness of motion space, motion direction entropy and hand shaking (i.e., camera stabilization) on VAQ assessment tasks. They also use naive Bayesian, SVM, and AdaBoost classification techniques. Finally, in [82], a variety of aesthetic-related features for video are designed, such as visual continuity and shot length, and their performance in retrieving professional videos in conjunction with a kernel SVM classifier is examined.

2.3 Deep Convolutional Neural Networks

During the last years, Deep Convolutional Neural Networks (DCNNs) have achieved state-of-the-art performance on various computer vision tasks, such as image classification [44, 124, 128]. For this purpose, many network architectures have been proposed and used successfully for the problem of image classification. Two of the most widely used ones include the Residual Network (ResNet) and the Wide Residual Network (WRN). He et al. [44] proposed ResNets, which have shortcut connections parallel to their normal convolutional layers, as a solution to the problems of vanishing/exploding gradient and hard optimization when increasing the model's parameters (i.e. adding more layers). Zagoruyko and Komodakis [124] showed that wide residual networks (named WRN) could outperform Deep ResNets with hundreds of layers, shifting the interest of the community to increasing the number of each layer's filters.

Besides the various network architectures that have been proposed in recent years, the research community has also directed its efforts in the loss function used in a CNN [5, 8, 104]. Most of these works are motivated by a specific problem domain and are inspired by similar losses that had been proposed for other learning algorithms (e.g., the SVM classifier). Liu et al. [73] proposed a generalized large-margin softmax (L-Softmax) loss which explicitly encourages intra-class compactness and inter-class separability between learned features. Furthermore, in [86], the authors proposed the contrastive-center loss, which learns a center for each class. In [113] the authors pro-

posed Large Margin Cosine Loss (LMCL) for the face recognition problem, to guide the deep CNNs to learn highly discriminative features by extending the cosine margin between decision boundaries. Wen et al. [113] proposed the center loss function, which combined with the softmax loss to jointly supervise the learning of CNNs improved the discriminative power of the deeply learned features for robust face recognition. In [72], the authors presented a deep hypersphere embedding approach for face recognition. For this purpose, they proposed the angular softmax (A-Softmax) loss for CNNs to learn discriminative face features (SphereFace) with angular margin. A-Softmax loss renders nice geometric interpretation by constraining learned features to be discriminative on a hypersphere manifold, which intrinsically matches the prior that faces also lie on a non-linear manifold. Finally, in [116], Wen et al. proposed the center loss function for the problem of face recognition. More specifically, the center simultaneously learns a center for deep features of each class and penalizes the distances between the deep features and their corresponding class centers. With the joint supervision of softmax loss and center loss, the authors achieve to robustify CNNs so as to obtain the deep features with the two key learning objectives; namely, inter-class dispension and intra-class compactness as much as possible.

While there has been significant research in the design of the architecture and of the filters in the convolutional layers, there is less work on the pooling layers. Xie et al. [117] considered the neurons in the hidden layer as neural words, and constructed a set of geometric neural phrases on top of them, borrowing the idea from the Bag-of-Visual-Words (BoVW) model. Subsequently, they proposed the Geometric Neural Phrase Pooling (GNPP) algorithm in order to efficiently encode these neural phrases. GNPP acts as a new type of hidden layer, which is inserted into a CNN between a convolution and a pooling layer, that punishes isolated neuron responses after convolution.

2.4 Conclusions

In this chapter we have mentioned and briefly discussed some of the most important works concerning the problem of learning under uncertainty. We began by discussing works that address the problem of learning with uncertain training labels, and we focused on works that assume uncertainty in input data representation. We followed by discussing works on two challenging visual understanding problems, namely video event detection and video aesthetic quality assessment, and works that use DCNNs for image classification.

The main proposed classifiers of this thesis, i.e., the linear SVM-GSU and kernel SVM-iGSU, in contrast to [12], do not violate the "iid" assumption for the training input data. Especially for the basic linear variant, we can model the uncertainty of each input training example using an arbitrary covariance matrix; that is, it allows anisotropic modeling of the uncertainty analytically in contrast to [97, 87, 126]. Moreover, we define a cost function that is convex and whose derivatives with respect to the optimization parameters can be expressed in closed form. Therefore, we can find their global optimal using an iterative gradient descent algorithm whose complexity is linear

with respect to the number of training data. For this purpose we used and modified the well-studied Pegasos algorithm [96]. Finally, in LSVM-GSU, we applied a linear subspace learning approach in order to address the situation where most of the mass of the Gaussians lies in a low dimensional manifold that can be different for each Gaussian, and subsequently solve the problem in lower-dimensional spaces. Learning in subspaces is widely used in various statistical learning problems [23, 74, 75].

Concerning the problem of video event detection, it is clear that regardless of whether the works discussed above address the problem of learning from a few positive examples or assume that an abundance of such examples is available, they all treat the training video representations as noise-free observations in the SVM input space. This is also the case for the problem of video aesthetic quality assessment. To the best of our knowledge, there has been no study dealing with uncertainty in the above problems, except for the published works that relate to this thesis.

Finally, concerning the extension of our method into the DCNN framework, although learning under uncertainty has been extensively studied in other learning paradigms [110, 87, 126, 65], the deep learning community has mostly directed its efforts into studying uncertainty in the output predictions of a network [15, 40, 4, 64], in an attempt to provide a degree of confidence of those predictions. In a similar spirit, Dorta et al. [27] proposed a network in order to predict a structured uncertainty distribution for a reconstructed image. More specifically, the authors proposed a model that learns to predict a full Gaussian covariance matrix for each reconstruction, which permits efficient sampling and likelihood evaluation. Finally, in [32] the authors study the a model's uncertainty due to the use of dropout. That is, they built a probabilistic interpretation of dropout which allowed for obtaining model uncertainty out of existing deep learning models.

Although our work shares some similarity with studies that adopt a large-margin loss function (e.g., [104, 113, 72]), it is different from them since it proposes a maximum-margin loss function and an efficient learning approach that allows its application in state-of-the-art architectures, like the WRN, in contrast to [104], for instance, that is applicable mostly in swallow architectures. Moreover, the maximum-margin loss that we present is more general than other large-margin losses that are mainly motivated by and applied to a specific problem domain (i.e., face recognition), usually introducing limiting constraints (as in [72]). Finally, the proposed method for exploiting the uncertainty that rises when using a pooling operation before the classification stage, i.e., the uncertainty in the position of the network classifier's training data, is different from – but also complementary to – works that study uncertainty at the output of a CNN (e.g., [40, 27, 32]).

Chapter 3

Linear Maximum Margin Classifier for Learning from Uncertain Data

Contents

In this chapter, we present the linear version of our classifier, for which we consider that training examples are given multivariate Gaussian distributions; that is, each training example is given as a pair of a mean vector and a covariance matrix (since a Gaussian distribution is uniquely defined by its first two moments). Moreover, each training example has a different covariance matrix expressing the uncertainty around its mean. An illustration is given in Fig. 3.1, where the shaded regions are bounded by iso-density loci of the Gaussians, and the means of the Gaussians for examples of the positive and negative classes are located at $\times$ and $\circ$ respectively. A classical SVM formulation would consider only the means of the Gaussians as training examples and, by optimizing the soft margin using the hinge loss and a regularization term, would arrive at the separating hyperplane depicted by the dashed line. In our formulation, we optimize for the soft margin using the same regularization but the *expected* value of the hinge loss, where the expectation is taken under the given Gaussians. By doing so, we take into consideration the various uncertainties and arrive at a drastically different decision border, depicted by the solid line in Fig. 3.1. It is worth noting that one would arrive at the same decision border with the classical SVM trained on a dataset containing samples drawn from the Gaussians in question, as the number of samples tend to infinity. In addition, our method degenerates to a classical SVM in the case that all of the Gaussians are isotropic with a variance that tends to zero.

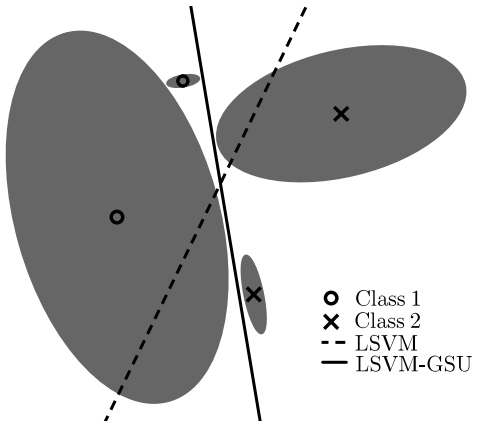


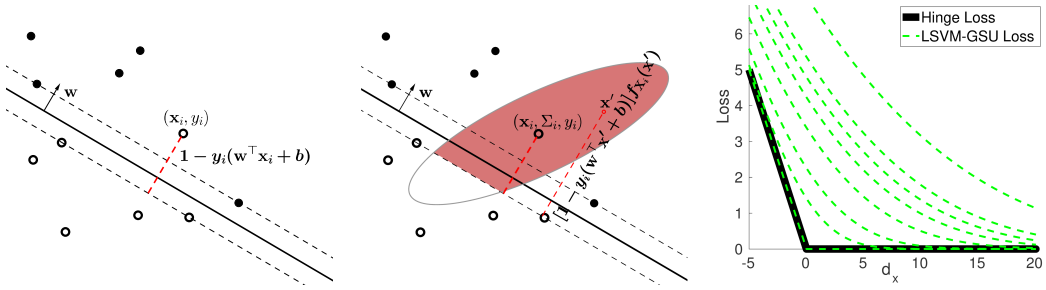
Figure 3.1: Linear SVM with Gaussian Sample Uncertainty (LSVM-GSU). The solid line depicts the decision boundary of the proposed algorithm, and the dashed line depicts the decision boundary of the standard linear SVM (LSVM).

In the rest of this chapter, we will develop a new classification algorithm whose training set is not just a set of vectors $\mathbf{x}_i$ in some multi-dimensional space, but rather a set of multivariate Gaussian distributions; that is, each training example consists of a mean vector $\mathbf{x}_i \in \mathcal{D}$ and a covariance matrix $\Sigma_i \in \mathbb{S}^n_{++}$; the latter expresses the uncertainty around the corresponding mean[1]. In Sect. 3.1, we first briefly review the linear SVM and then describe in detail the proposed linear SVM with Gaussian Sample Uncertainty (LSVM-GSU). In Sect. 3.2 we motivate and describe a formulation that allows learning in linear subspaces. In the general case we arrive at different subspaces for the different Gaussians – this allows, for example, dealing with covariance matrices that are of low rank. In Sect. 3.3 we discuss how the proposed algorithm relates to standard SVM when the latter is fed with samples drawn from the input Gaussians. In Sect. 3.4 we describe a SGD algorithm for efficiently solving the SVM-GSU optimization problem. Finally, in Sect. 3.5, we provide the experimental results of the application of SVM-GSU to synthetic data and to five publicly available and popular datasets. In the same section, we provide comparisons with the standard SVM and other state of the art methods.

3.1 Linear SVM with Gaussian Sample Uncertainty (LSVM-GSU)

We begin by briefly describing the standard SVM algorithm. Let us consider the supervised learning framework and denote the training set with $\mathcal{X} = \big\{(\mathbf{x}_i, y_i) \colon \mathbf{x}_i \in \mathbb{R}^n, y_i \in \{\pm 1\}, i = 1, \ldots, \ell\big\}$, where $\mathbf{x}_i$ is a training example and y_i is the corresponding class label. Then, the standard linear SVM learns a hyperplane $\mathcal{H} \colon \mathbf{w}^\top \mathbf{x} + b = 0$ that

[1] $\mathcal{D}$ is typically a subset of the n-dimensional Euclidean space of column vectors, while $\mathbb{S}^n_{++}$ denotes the convex cone of all symmetric positive definite $n \times n$ matrices with entries in $\mathcal{D} \subseteq \mathbb{R}^n$.



(a) (b) (c)

Figure 3.2: Illustrative example of calculating (a) the standard linear SVM's hinge loss, and (b) the proposed linear SVM-GSU's loss. In (c), the hinge loss is compared with the proposed linear SVM-GSU's loss for various quantities of uncertainty.

minimizes with respect to $\mathbf{w}$, b the following objective function:

$$\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{\ell}\sum_{i=1}^{\ell} \max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right), \tag{3.1}$$

where $h(t) = \max(0, 1-t)$ is the "hinge" loss function [43]. An illustrative example of the hinge loss calculation is given in Fig. 3.2, where in Fig. 3.2a the red dashed line indicates the loss introduced by the misclassified example $(\mathbf{x}_i, y_i)$ and in Fig. 3.2c the hinge loss is shown in the black bold line.

In this work we assume that, instead of the i-th training example in the form of a vector, we are given a multivariate Gaussian distribution with mean vector $\mathbf{x}_i$ and covariance matrix Σ_i. One could think of this as that the covariance matrix, Σ_i, models the uncertainty about the position of training samples around $\mathbf{x}_i$. Formally, our training set is a set of ℓ annotated Gaussian distributions, i.e.,

$$\mathcal{X}' = \left\{(\mathbf{x}_i, \Sigma_i, y_i) \colon \mathbf{x}_i \in \mathbb{R}^n,\ \Sigma_i \in \mathbb{S}^n_{++},\ y_i \in \{\pm 1\},\ i = 1, \ldots, \ell\right\},$$

where $\mathbf{x}_i \in \mathbb{R}^n$ and $\Sigma_i \in \mathbb{S}^n_{++}$ are respectively the mean vector and the covariance matrix of the i-th example, and y_i is the corresponding label. Then, we define ℓ random variables, $\mathbf{X}_i$, each of which we assume that follows the corresponding n-dimensional Gaussian distribution $\mathcal{N}(\mathbf{x}_i, \Sigma_i)$ and define an optimization problem where the misclassification cost for the i-th example is the expected value of the hinge loss for the corresponding Gaussian. Formally, the optimization problem, in its unconstrained primal form, is the minimization with respect to $\mathbf{w}$, b of

$$\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{\ell}\sum_{i=1}^{\ell} \int_{\mathbb{R}^n} \max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x} + b)\right) f_{\mathbf{X}_i}(\mathbf{x})\, \mathrm{d}\mathbf{x}, \tag{3.2}$$

where

$$f_{\mathbf{X}_i}(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}_i)^\top \Sigma_i^{-1}(\mathbf{x} - \mathbf{x}_i)\right)$$

is the probability density function (PDF) of the i-th Gaussian distribution. The above objective function $\mathcal{J}\colon \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ can be written as

$$\mathcal{J}(\mathbf{w}, b) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{\ell}\sum_{i=1}^{\ell} \mathcal{L}\big(\mathbf{w}, b; (\mathbf{x}_i, \Sigma_i, y_i)\big), \tag{3.3}$$

where, as stated above, the loss function $\mathcal{L}$ for the i-th example (i.e. the i-th Gaussian) is defined as the expected value of the hinge loss for the Gaussian in question. That is,

$$\mathcal{L}(\mathbf{w}, b) = \int_{\mathbb{R}^n} \max\big(0, 1 - y_i(\mathbf{w}^\top \mathbf{x} + b)\big) f_{\mathbf{X}_i}(\mathbf{x})\, \mathrm{d}\mathbf{x}. \tag{3.4}$$

We proceed to express the objective function (3.3) and its derivatives in closed form. This will allow us to solve the corresponding optimization problem using an efficient SGD approach. More specifically, the loss can be expressed as

$$\mathcal{L}(\mathbf{w}, b) = \int_{\Omega_i} \Big[1 - y_i(\mathbf{w}^\top \mathbf{x} + b)\Big] f_{\mathbf{X}_i}(\mathbf{x})\, \mathrm{d}\mathbf{x}, \tag{3.5}$$

where Ω_i denotes the halfspace of $\mathbb{R}^n$ that is defined by the hyperplane $\mathcal{H}'\colon y_i(\mathbf{w}^\top \mathbf{x} + b) = 1$ as $\Omega_i = \{\mathbf{x} \in \mathbb{R}^n \colon y_i(\mathbf{w}^\top \mathbf{x} + b) \leq 1\}$, and is the halfspace to which misclassified samples lie. This is illustrated in Fig. 3.2b, where a misclassified example $(\mathbf{x}_i, \Sigma_i, y_i)$ introduces a loss indicated by the shaded region. For the calculation of this loss, all points that belong to the halfspace $\Omega_i = \{\mathbf{x} \in \mathbb{R}^n \colon y_i(\mathbf{w}^\top \mathbf{x} + b) \leq 1\}$, i.e., the points $\mathbf{x}' \in \Omega_i$, contribute to it by a quantity of $[1 - y_i(\mathbf{w}^\top \mathbf{x}' + b)] f_{\mathbf{X}_i}(\mathbf{x}')$. For one such $\mathbf{x}'$ denoted by a red circle in Fig. 3.2b, the first part of the above product, $1 - y_i(\mathbf{w}^\top \mathbf{x}' + b)$, corresponds to the typical hinge loss of SVM, shown as a red dashed line in this example. The total loss introduced by the misclassified example $(\mathbf{x}_i, \Sigma_i, y_i)$ is obtained by integrating all these quantities over the halfspace Ω_i.

Using Theorem 1 proved in Appendix A, for the halfspace

$$\Omega_i^+ = \left\{\mathbf{x} \in \mathbb{R}^n \colon y_i \left(\mathbf{w}^\top \mathbf{x} + b\right) \leq 1\right\},$$

the above integral is evaluated in terms of $\mathbf{w}$ and b as follows

$$\mathcal{L}(\mathbf{w}, b) = \frac{d_{\mathbf{x}_i}}{2}\left[\operatorname{erf}\left(\frac{d_{\mathbf{x}_i}}{d_{\Sigma_i}}\right) + 1\right] + \frac{d_{\Sigma_i}}{2\sqrt{\pi}} \exp\left(-\frac{d_{\mathbf{x}_i}^2}{d_{\Sigma_i}^2}\right), \tag{3.6}$$

where $d_{\mathbf{x}_i} = 1 - y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right)$, $d_{\Sigma_i} = \sqrt{2\mathbf{w}^\top \Sigma_i \mathbf{w}}$, and $\operatorname{erf}\colon \mathbb{R} \to (-1, 1)$ is the error function, defined as $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, \mathrm{d}t$. For a training example $(\mathbf{x}, \Sigma, y)$, Fig. 3.2c shows the proposed loss in dashed green lines for constant values of d_Σ (constant amounts of uncertainty). We note that as $d_\Sigma \to 0$, SVM-GSU's loss virtually coincides with the SVM's hinge loss, while it can be easily verified that, regardless of d_Σ, as $d_{\mathbf{x}} \to \infty$ the SVM-GSU's loss will eventually converge to zero (as the hinge loss does).

Let us note that the covariance matrix of each training example describes the uncertainty around the corresponding mean; that is, as the covariance matrix approaches

the zero matrix, the certainty increases. At the extreme[2], as $\Sigma \to \mathbf{0}$, the proposed loss converges to the hinge loss function used in the standard SVM formulation [43]. This implies that the proposed formulation is a generalization of the standard SVM; the two classifiers are equivalent when the covariance matrices tend to the zero matrix.

In Appendix B we show that the objective function (3.3) is convex with respect to $\mathbf{w}$ and b; therefore, based on Pegasos [96] SVM solver, we modify and present a SGD algorithm in Sect. 3.4 for solving the corresponding optimization problem. Since the objective function is convex, we can obtain the global optimal solution. Moreover, it can be shown that the proposed loss function (3.4) enjoys the consistency property [89, 125], i.e., it leads to consistent results with the $0-1$ loss given the presence of infinite data. By differentiating $\mathcal{J}$ with respect to $\mathbf{w}$ and b, we obtain, respectively,

$$\frac{\partial \mathcal{J}}{\partial \mathbf{w}} = \lambda \mathbf{w} + \frac{1}{\ell} \sum_{i=1}^{\ell} \left[\frac{\exp\left(-d_{\mathbf{x}_i}^2 / d_{\Sigma_i}^2\right)}{\sqrt{\pi} d_{\Sigma_i}} \Sigma_i \mathbf{w} - \frac{1}{2} \left(\operatorname{erf}\left(\frac{d_{\mathbf{x}_i}}{d_{\Sigma_i}} \right) + 1 \right) \mathbf{x}_i \right], \tag{3.7}$$

and

$$\frac{\partial \mathcal{J}}{\partial b} = -\frac{1}{\ell} \sum_{i=1}^{\ell} \left[\operatorname{erf}\left(\frac{d_{\mathbf{x}_i}}{d_{\Sigma_i}} \right) + 1 \right]. \tag{3.8}$$

Despite the complex appearance of the loss function and its derivatives, their computation essentially requires the calculation of the inner product $\mathbf{w}^\top \mathbf{x}_i$ (which is the same as in standard SVM), plus that of the quadratic form $\mathbf{w}^\top \Sigma_i \mathbf{w}$, which requires $\frac{n(n+1)}{2}$ multiplications, since Σ_i is symmetric. The latter, in the case of diagonal covariance matrices, is equivalent to the computation of an inner product, i.e., of complexity $\mathcal{O}(n)$. Moreover, each one of $\mathbf{w}^\top \mathbf{x}_i$ and $\mathbf{w}^\top \Sigma_i \mathbf{w}$ needs to be computed just once for calculating the loss function and its derivatives for a given $\mathbf{w}$. It is worth noting that, in practice, as shown in Sect. 3.5, in real-world problems uncertainty usually rises in diagonal form. In such cases, the proposed algorithm is quite efficient and exhibits very similar complexity to the standard linear SVM (less than 10% slower).

Once the optimal values of the parameters $\mathbf{w}$ and b are learned, an unseen testing datum, $\mathbf{x}_t$, can be classified to one of the two classes according to the sign of the (signed) distance between $\mathbf{x}_t$ and the separating hyperplane. That is, the predicted label of $\mathbf{x}_t$ is computed as $y_t = \operatorname{sgn}(d_t)$, where $d_t = (\mathbf{w}^\top \mathbf{x}_t + b)/\|\mathbf{w}\|$. The posterior class probability, i.e, a probabilistic degree of confidence that the testing sample belongs to the class to which it has been classified, can be calculated using the well-known Platt scaling approach [85] for fitting a sigmoid function, $S(t) = 1/(1 + e^{\sigma_A t + \sigma_B})$. This is the same approach that is used in the standard linear SVM formulation (e.g., see [17]) for evaluating a sample's class membership at the testing phase.

[2] A zero covariance matrix exists due to the well known property that the set of symmetric positive definite matrices is a convex cone with vertex at zero.

3.2 Solving the linear SVM-GSU in linear subspaces

The derivations in Sect. 3.1 were made for the general case of full rank covariance matrices that can be different for each of the examples. Clearly, one can introduce constraints on the covariance matrices, such as them being diagonal, block diagonal, or multiples of the identity matrix. In this way one can model different types of uncertainty – examples will be given in the section of experimental results. However, in some cases, especially when the dimensionality of the data is high, most of the mass of the Gaussian distributions will lie in a few directions in the feature space that may be different for each example and may not be aligned with the feature axes. To address this issue we alter the formulation and work directly in the subspaces that preserve most of the variance. More specifically, we propose a methodology for approximating the loss function of SVM-GSU, by projecting the vectors $\mathbf{x}$ in (3.5) into a linear subspace and integrating the hinge loss function in that subspace instead of the original feature space. A separate subspace is used for each of the training examples, that is, for each of the input Gaussians. For a given Gaussian distribution, the projection matrix is found by performing eigenanalysis on the covariance matrix and the dimensionality of each subspace is defined so as to preserve a certain fraction of the total variance.

More specifically, by performing eigenanalysis on the covariance matrix of the random vector $\mathbf{X}_i$, the latter is decomposed as $\Sigma_i = U_i \Lambda_i U_i^\top$, where Λ_i is an $n \times n$ diagonal matrix consisting of the eigenvalues of Σ_i, i.e. $\Lambda_i = \text{diag}(\lambda_i^1, \ldots, \lambda_i^n)$, so that $\lambda_i^1 \geq, \ldots, \geq \lambda_i^n > 0$, while U_i is an $n \times n$ orthonormal matrix, whose j-th column, $\mathbf{u}_i^j$, is the eigenvector corresponding to the j-th eigenvalue, λ_i^j.

Let us keep the first $d_i \leq n$ eigenvectors, so that a certain fraction $p \in (0, 1]$ of the total variance is preserved, i.e., $\frac{\sum_{t=1}^{d_i} \lambda_i^t}{\sum_{t=1}^{n} \lambda_i^t} > p$. Then, we construct the $n \times d_i$ matrix U_i' by keeping the first d_i columns of U_i, i.e., $U_i' = [\mathbf{u}_i^1 \ \mathbf{u}_i^2 \ \ldots \ \mathbf{u}_i^{d_i}]$. Now, by using the projection matrix $P_i = {U_i'}^\top$, we define a new random vector $\mathbf{Z}_i$, such that $\mathbf{Z}_i = P_i \mathbf{X}_i$. Then, $\mathbf{Z}_i \in \mathbb{R}^{d_i}$ follows a multivariate Gaussian distribution (since $\mathbf{X}_i \sim \mathcal{N}(\mathbf{x}_i, \Sigma_i)$), i.e. $\mathbf{Z}_i \sim \mathcal{N}(\mathbf{z}_i, \Sigma_i^z)$, with mean vector $\mathbf{z}_i = \mathbb{E}\big[P_i \mathbf{X}_i\big] = P_i \mathbb{E}\big[\mathbf{X}_i\big] = P_i \mathbf{x}_i$ and (diagonal) covariance matrix $\Sigma_i^z = \Lambda_i^z$. Let $f_{\mathbf{Z}_i}$ denote the PDF of $\mathbf{Z}_i$.

We proceed to approximate the expected value of the hinge loss in the original space (3.5), by considering the integral in the new, lower-dimensional space where most of the variance is preserved. More specifically, $\mathbf{x} \approx P_i^\top \mathbf{z} \implies \mathbf{w}^\top \mathbf{x} \approx \mathbf{w}^\top (P_i^\top \mathbf{z}) = {\mathbf{w_z}}^\top \mathbf{z}$, where $\mathbf{w}_z = P_i \mathbf{w}$. Consequently, the loss function for the i-th example, that is the integral in the RHS of (3.5) can be approximated by the quantity

$$\int_{\Omega_i^z} \Big[1 - y_i \big(\mathbf{w}_z^\top \mathbf{z} + b\big)\Big] \, f_{\mathbf{Z}_i}(\mathbf{z}) \, \mathrm{d}\mathbf{z},$$

where Ω_i^z denotes the projected halfspace on $\mathbb{R}^{d_i}$, that is, $\Omega_i^z = \big\{\mathbf{z} \in \mathbb{R}^{d_i} : y_i \big(\mathbf{w}_z^\top \mathbf{z} + b\big) \leq 1\big\}$. Using Theorem 1 (Appendix A), we can then give this approximation of the loss

function $\mathcal{L}'\colon \mathbb{R}^{d_i} \times \mathbb{R} \to \mathbb{R}$, in closed form as follows:

$$\mathcal{L}'(\mathbf{w}, b) = \frac{d_{\mathbf{z}_i}}{2}\left[\operatorname{erf}\left(\frac{d_{\mathbf{z}_i}}{d_{\Sigma_i^z}}\right) + 1\right] + \frac{d_{\Sigma_i^z}}{2\sqrt{\pi}} \exp\left(-\frac{d_{\mathbf{z}_i}^2}{d_{\Sigma_i^z}^2}\right) \tag{3.9}$$

where $d_{\mathbf{z}_i} = 1 - y_i\left(\mathbf{w}_z^\top \mathbf{z}_i + b\right)$, $d_{\Sigma_i^z} = \sqrt{2\mathbf{w}_z^\top \Sigma_i^z \mathbf{w}_z}$. Therefore, the objective function $\mathcal{J}'\colon \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$, given by (3.3) can be approximated as follows

$$\mathcal{J}'(\mathbf{w}, b) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{\ell}\sum_{i=1}^{\ell} \mathcal{L}'\left(P_i\mathbf{w}, b; (\mathbf{z}_i, \Sigma_i^z, y_i)\right). \tag{3.10}$$

Similarly to $\mathcal{J}$, we can show that $\mathcal{J}'$ is also convex with respect to the unknown parameters $\mathbf{w}$ and b of the separating hyperplane. Moreover, using the chain rule, we can obtain the partial derivatives of $\mathcal{J}'$ with respect to $\mathbf{w}$ and b in closed form, and therefore use a stochastic gradient method to arrive at the global optimum. More specifically,

$$\frac{\partial \mathcal{J}'}{\partial \mathbf{w}} = \lambda\mathbf{w} + \frac{1}{\ell}\sum_{i=1}^{\ell} \frac{\partial}{\partial \mathbf{w}_z}\mathcal{L}'\big(\mathbf{w}_z, b; (\mathbf{z}_i, \Sigma_i^z, y_i)\big)\frac{\partial \mathbf{w}_z}{\partial \mathbf{w}},$$

where $\frac{\partial}{\partial \mathbf{w}}\mathbf{w}_z = \frac{\partial}{\partial \mathbf{w}}P_i\mathbf{w} = P_i$. By differentiating $\mathcal{L}'$ with respect to $\mathbf{w}_z$, and replacing in the above, we arrive at

$$\frac{\partial \mathcal{J}'}{\partial \mathbf{w}} = \lambda\mathbf{w} + \frac{1}{\ell}\sum_{i=1}^{\ell}\left[\frac{\exp\left(-d_{\mathbf{z}_i}^2/d_{\Sigma_i^z}^2\right)}{\sqrt{\pi}d_{\Sigma_i^z}} P_i^\top(\Sigma_i^z\mathbf{w}_z) - \frac{1}{2}\left(\operatorname{erf}\left(\frac{d_{\mathbf{z}_i}}{d_{\Sigma_i^z}}\right) + 1\right)P_i^\top\mathbf{z}_i\right], \tag{3.11}$$

that is a closed form equation that gives the partial derivatives of the cost with respect to $\mathbf{w}$. Similarly, the first partial derivative of $\mathcal{J}'$ with respect to b can be obtained as follows

$$\frac{\partial \mathcal{J}'}{\partial b} = -\frac{1}{\ell}\sum_{i=1}^{\ell}\left[\operatorname{erf}\left(\frac{d_{\mathbf{z}_i}}{d_{\Sigma_i^z}}\right) + 1\right]. \tag{3.12}$$

where $\mathbf{w}_z = P_i\mathbf{w}$, $\Sigma_i^z = P_i\Sigma_i P_i^\top$.

To summarize, in the low-dimensional spaces $\mathbb{R}^{d_i}$, the loss function is computed as shown in (3.9). The objective function is computed as shown in (3.10) and its first derivatives are computed as in (3.11) and (3.12). Finally, let us note that in the above equations, the only matrix operations involve the projection matrix P_i. Since the covariance matrices Σ_i^z are diagonal, all operations that involve them boil down to efficient vector rescaling and vector norm calculations.

3.3 To sample or not to sample?

The data term in our formulation (see (3.4)) is the expected value of the classical SVM cost when data samples are drawn from the multi-dimensional Gaussian distributions. It therefore follows that a standard linear SVM would arrive at the same hyperplane when sufficiently many samples are drawn from them. How many samples are needed

to arrive at the same hyperplane is something that cannot be computed analytically. Nevertheless, our analysis and results indicate that this number can be prohibitively high, especially in the case of high-dimensional spaces.

More specifically, in what follows, we show that the difference between the analytically calculated expected value of the hinge loss (3.4) and its sample mean is bounded by a quantity that is inversely related to the dimensionality of the feature space. Let $\mathcal{L}$ be the expected loss given analytically as in (3.4), and $\tilde{\mathcal{L}}_N$ its approximation when N samples are drawn from the Gaussians. Since the hinge loss is $\|\mathbf{w}\|$-Lipschitz[3] with respect to the Euclidean norm, we can use a result due to Tsirelson et al. [106] that provides a concentration inequality for Lipschitz functions of Gaussian variables. By doing so, for all $r \geq 0$, we arrive at the following concentration inequality

$$P\left(\left|\mathcal{L} - \tilde{\mathcal{L}}_N\right| \geq r\right) \leq 2\exp\left(-\frac{r^2}{2\|\mathbf{w}\|^2}\right). \tag{3.13}$$

That is, the tails of the error probability decay exponentially with r^2. More interestingly, they increase with the squared norm of $\|\mathbf{w}\|$, and therefore with the dimensionality of the input space, n. Consequently, as n increases, one needs to generate more samples from the Gaussians in order to preserve a desired approximation of the loss.

This means that for spaces of high dimensionality the number of samples needed to approximate (3.4) sufficiently well, can be prohibitively high. We experimentally demonstrated this with a toy example in Sect. 3.5.1 (see Fig. 3.4), where we show that in 2 dimensions we need approximately 3 orders of magnitude more samples to arrive at the same hyperplane, while for 3 dimensions we need 4 orders of magnitude more samples. Our experimental results on the large-scale MED dataset (Sect. 3.5.6) also show the limitations of a sampling approach.

3.4 A stochastic gradient descent solver for SVM-GSU

Motivated by the Pegasos algorithm (Primal Estimated sub-GrAdient SOlver for SVM), first proposed by Shalev-Shwartz et al. in [96], we modify and present a stochastic sub-gradient descent algorithm for solving SVM-GSU in order to efficiently address scalability requirements[4].

Pegasos is a well-studied algorithm [96, 55] providing both state of the art classification performance and great scalability. It requires $\tilde{\mathcal{O}}(1/\epsilon)$ number of iterations in order to obtain a solution of accuracy ϵ, in contrast to previous analyses of SGD methods that require $\tilde{\mathcal{O}}(d/(\lambda\epsilon))$ iterations, where d is a bound on the number of non-zero features in each example[5]. Since the run-time does not depend directly on the size

[3] A function $h\colon \mathbb{R}^n \to \mathbb{R}$ is $\mathscr{L}$-Lipschitz with respect to the Euclidean norm if $|h(\mathbf{x}) - h(\mathbf{y})| \leq \mathscr{L}\|\mathbf{x} - \mathbf{y}\|$, $\mathscr{L} > 0$. Indeed, the hinge loss $h(\mathbf{x}) = \max(0, 1 - y(\mathbf{w}^\top \mathbf{x} + b))$ is $\|\mathbf{w}\|$-Lipschitz since $|h(\mathbf{x}) - h(\mathbf{y})| \leq \left|1 - y(\mathbf{w}^\top \mathbf{x} + b) - 1 + y(\mathbf{w}^\top \mathbf{y} + b)\right| \leq \|\mathbf{w}\|\|\mathbf{x} - \mathbf{y}\|$.

[4] A C++ implementation of the proposed method can be found at `https://github.com/chi0tzp/svm-gsu`.

[5] We use the $\tilde{\mathcal{O}}$ notation (soft-O) as a shorthand for the variant of $\mathcal{O}$ (big-O) that ignores logarithmic factors; that is, $f(n) \in \tilde{\mathcal{O}}(g(n)) \iff \exists k \in \mathbb{N}\colon f(n) \in \mathcal{O}(g(n)\log^k(g(n)))$.

of the training set, the resulting algorithm is especially suited for learning from large datasets.

Given a training set $\mathcal{X} = \big\{(\mathbf{x}_i, \Sigma_i, y_i) \colon \mathbf{x}_i \in \mathbb{R}^n, \Sigma_i \in \mathbb{S}^n_{++}, y_i \in \{\pm 1\},\ i = 1, \ldots, \ell\big\}$, the proposed algorithm solves the following optimization problem

$$\min_{\mathbf{w},b} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{\ell}\sum_{i=1}^{\ell} \mathcal{L}\big(\mathbf{w}, b; (\mathbf{x}_i, \Sigma_i, y_i)\big). \tag{3.14}$$

The algorithm receives as input two parameters: (i) the number of iterations, T, and (ii) the number of examples to use for calculating sub-gradients, k. Initially, we set $\mathbf{w}^{(1)}$ to any vector whose norm is at most $1/\sqrt{\lambda}$ and $b^{(1)} = 0$. On the t-th iteration, we randomly choose a subset of $\mathcal{X}$, of cardinality k, i.e., $\mathcal{X}_t \subseteq \mathcal{X}$, where $|\mathcal{X}_t| = k$, and set the learning rate to $\eta_t = \frac{1}{\lambda t}$. Then, we approximate the objective function of the above optimization problem with

$$\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{k}\sum_{(\mathbf{x}_i,\Sigma_i,y_i)\in\mathcal{X}_t} \mathcal{L}\big(\mathbf{w}, b; (\mathbf{x}_i, \Sigma_i, y_i)\big).$$

Then, we perform the update steps

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \frac{\eta_t}{k}\frac{\partial J}{\partial \mathbf{w}}, \quad b^{(t+1)} \leftarrow b^{(t)} - \frac{\eta_t}{k}\frac{\partial J}{\partial b},$$

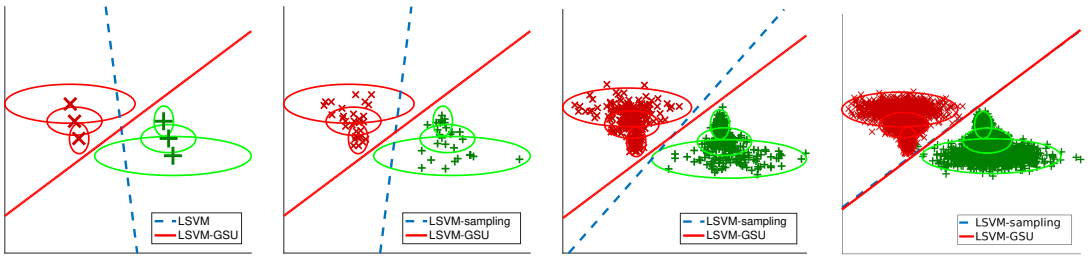
where the first-order derivatives are given in (3.7), (3.8), if the training is conducted in the original space (Sect. 3.1), or in (3.11), (3.12), if the learning is conducted in linear subspaces (Sect. 3.2). Last, we project $\mathbf{w}^{(t+1)}$ onto the ball of radius $1/\sqrt{\lambda}$, i.e., the set $\mathcal{B} = \{\mathbf{w} \colon \|\mathbf{w}\| \leq 1/\sqrt{\lambda}\}$. The output of the algorithm is the pair of $\mathbf{w}^{(T+1)}$, $b^{(T+1)}$. Algorithm 2 describes the proposed method in pseudocode.

Algorithm 1 A stochastic sub-gradient descent algorithm for solving SVM-GSU.

Inputs:
  $\mathcal{X}$, λ, T, k
Initialize:
  $b^{(1)} = 0$, $\mathbf{w}^{(1)}$ such that $\|\mathbf{w}^{(1)}\| \leq \frac{1}{\sqrt{\lambda}}$
for $t = 1, 2, \ldots, T$ **do**
  Choose $\mathcal{X}_t \subseteq \mathcal{X}$, where $|\mathcal{X}_t| = k$
  Set $\eta_t = \frac{1}{\lambda t}$
  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \frac{\eta_t}{k}\frac{\partial J}{\partial \mathbf{w}}$
  $\mathbf{w}^{(t+1)} \leftarrow \min\Big(1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}^{(t+1)}\|}\Big)\mathbf{w}^{(t+1)}$
  $b^{(t+1)} \leftarrow b^{(t)} - \frac{\eta_t}{k}\frac{\partial J}{\partial b}$
end for

3.5 Experiments

In this section we first illustrate the workings of the proposed linear SVM-GSU classifier on a synthetic 2D toy example (Sect. 3.5.1) and then apply the algorithm on five



(a) (b) $N = 10$ (c) $N = 10^2$ (d) $N = 10^3$

Figure 3.3: Toy example illustrating on 2D data, (a) the proposed LSVM-GSU (red solid line) in comparison with the standard LSVM (blue dashed line), and (b)-(d) with the standard SVM that learns by sampling from the input Gaussians (LSVM-sampling), where N is the sampling size.

different classification problems using publicly available and popular datasets. Here, we summarize how the uncertainty is modeled in each case, so as to illustrate how our framework can be applied in practice.

First, we address the problem of image classification of handwritten digits (Sect. 3.5.2) using the MNIST dataset. As we show in Appendix C, by using a first-order Taylor approximation around a certain image with respect to some common image transformations (small translations in our case), we show that the images that would be produced by those translations would follow a Gaussian distribution with mean the image in question and a covariance matrix whose elements are functions of the derivatives of the image intensities/color with respect to those transformations. In the simple case of spatial translations, the covariance elements are functions of the spatial gradients. This is a case where the uncertainty is modeled. We show that our method outperforms the linear SVM and other SVM variants that handle uncertainty isotropically.

Second, we address the binary classification problem using the Wisconsin Diagnostic Breast Cancer (WDBC) dataset (Sect. 3.5.3). This is a case in which each data example summarizes a collection of samples by their second order statistics. More specifically, each data example contains as features the mean and the variance of measurements on several cancer cells – mean and variances over the different cells. With our formulation we obtain state of the art results on this dataset.

Third, we address the problem of emotional analysis using electroencephalogram (EEG) signals (Sect. 3.5.4). In this case, we exploit a very popular method for estimating the power spectrum of time signals; namely the Welch method, which allows for estimating not only the mean values of the features (periodograms), but also their variances, making it suitable for using the proposed SVM-GSU.

Fourth, we address the problem of detection of advertisements in TV news videos (Sect. 3.5.5). This is an interesting case where uncertainty information is given only for a few dimensions of the input space, rendering inapplicable the methods that treat uncertainty isotropically. In contrast, the proposed method can model such uncertainty

types using low-rank covariance matrices.

Finally, we address the challenging problem of complex event detection in video (Sect. 3.5.6). We used the ∼5K outputs of a pre-trained DCNN in order to extract a representation for each frame in a video and calculated the mean and covariances over the frames of a video in order to classify it. This is a second example in which the mean and the covariance matrices are calculated from data. We show that our formulation outperforms the linear SVM and other SVM variants that handle uncertainty isotropically.

3.5.1 Toy example using synthetic data

In this subsection, we present a toy example on 2D data that provides insights to the way the proposed algorithm works. As shown in Fig. 3.3a, negative examples are denoted by red $\times$ marks, while positive ones by green crosses. We assume that the uncertainty of each training example is given via a covariance matrix. For illustration purposes, we draw the iso-density loci of points at which the value of the PDF of the Gaussian is the 0.03% of its maximum value.

First, a baseline linear SVM (LSVM) is trained using solely the centres of the distributions; i.e., ignoring the uncertainty of each example. The resulting separating boundary is the dashed blue line in Fig. 3.3a. The proposed linear SVM-GSU (LSVM-GSU) is trained using both the centres of the above distributions and the covariance matrices. The resulting separating boundary is the solid red line in Fig. 3.3a. It is clear that the separating boundaries can be very different and that the solid red line is a better one given the assumed uncertainty modeling.

Next, we investigate on how many samples are needed in order to obtain LSVM-GSU's separating line by sampling N samples from each Gaussian and using the standard LSVM (LSVM-sampling). The results for various values of N are depicted in Fig. 3.3, where it is clear that ones needs almost 3 orders of magnitude more examples. In order to investigate how this number changes with the dimensionality of the feature space we performed the same experiment in a similar 3D dataset. In Fig. 3.4 we plot the angle between the hyperplanes obtained by the LSVM-GSU and the LSVM-sampling for both the 2D and the 3D datasets. We observe that, in the 3D case, we need at least one order of magnitude more samples from each Gaussian, compared to the 2D case; that is, in the 2D case, we obtain $\theta \approx 1.7^\circ$ using $N = 10^3$ samples from each Gaussian, while in the 3D case, the sampling size for obtaining the same approximation ($\theta \approx 1.7^\circ$) is $N = 5 \times 10^4$. This is indicative of the difficulties of using the sampling approach when dealing with high-dimensional data, where the number of dimensions is in the hundreds or thousands.

3.5.2 Hand-written digit classification

Dataset and experimental setup

The proposed algorithm is also evaluated in the problem of image classification using the MNIST dataset of handwritten digits [14]. The MNIST dataset provides a training

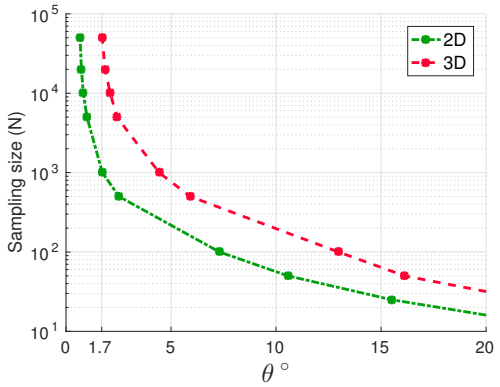


Figure 3.4: Difference between the separating hyperplanes of LSVM-GSU and the standard LSVM with sampling (angle θ), when varying the number of samples used in the standard SVM, for the 2D and 3D toy datasets.

Table 3.1: MNIST "1" versus "7" experimental results in terms of testing accuracy. The proposed LSVM-GSU is compared to the baseline linear SVM (LSVM), Power SVM (PSVM) [126], and a linear SVM extension which handles the uncertainty isotropically (LSVM-iso), as in [12, 87].

Dataset		**D_0**	**D_1**	**D_2**
LSVM		0.9952	0.9362	0.8240
PSVM [126]		0.9963	0.9315	0.8157
LSVM-iso (as in [12, 87])		0.9968	0.9327	0.8133
LSVM-GSU	Learning in original space	0.9971	0.9452	0.8310
	Learning in linear subspaces	**0.9972** (0.99)	**0.9480** (0.97)	**0.8562** (0.89)
Dataset		**D_3**	**D_4**	**D_5**
LSVM		0.6830	0.6558	0.6027
PSVM [126]		0.7017	0.6650	0.6259
LSVM-iso (as in [12, 87])		0.7222	0.6675	0.6328
LSVM-GSU	Learning in original space	0.7216	0.6708	0.6353
	Learning in linear subspaces	**0.7543** (0.85)	**0.6974** (0.95)	**0.6640** (0.25)

set of 60K examples (approx. 6000 examples per digit), and a test set of 10K examples (approx. 1000 examples per digit). Each sample is represented by a 28×28 8-bit image.

In order to make the dataset more challenging, as well as to model a realistic distortion that may happen to this kind of images, the original MNIST dataset was "polluted" with noise. More specifically, each image example was rotated by a random angle uniformly drawn from $[-\theta, +\theta]$, where θ is measured in degrees. Moreover, each image was translated by a random vector $\mathbf{t}$ uniformly drawn from $[-t_p, +t_p]^2$, where t_p is a positive integer expressing distance that is measured in pixels. We created five different noisy datasets by setting $\theta = 15°$ and $t_p \in \{3, 5, 7, 9, 11\}$, resulting in the polluted datasets D_1 to D_5, respectively. D_0 denotes the original MNIST dataset.

We created six different experimental scenarios using the above datasets (D_0-D_5). First, we defined the problem of discriminating the digit one ("1") from the digit seven ("7") similarly to [33]. Each class in the training procedure consists of 25 samples, ran-

domly chosen from the pool of digits one ($6k$ totally) and seven ($6k$ totally), while the evaluation of the trained classifier is carried out on the full testing set ($2k$ examples). In each experimental scenario we report the average of 100 runs and we compare the proposed linear SVM-GSU (LSVM-GSU) to the baseline linear SVM (LSVM), Power SVM [126], and LSVM-iso (a variation of SVM formulation that handles only isotropic uncertainty, similarly to [12, 87]). We report the testing accuracy and the mean testing accuracy across 100 runs. Finally, we repeat the above experiments for various sizes of the training set; i.e., using 25, 50, 100, 500, 1000, 3000, 6000 positive examples per digit, in order to investigate how this affects the results.

Uncertainty modeling

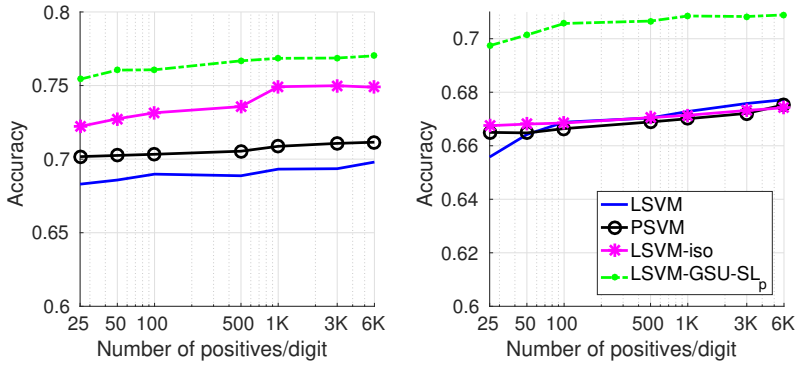
In Appendix C, we propose a methodology that, given an image, models the distribution of the images that result by small random translations of it. We show that under a first-order Taylor approximation of the image intensities/color with respect to those translations, and the assumption that the translations are small and follow a Gaussian distribution, the resulting distribution of the images is also a Gaussian with mean the original image and a covariance matrix whose elements are functions of the image derivatives with respect to the transforms – in this case functions of the image spatial gradients. The derivation could be straightforwardly extended to other transforms (e.g. rotations, scaling). However, in this work we solely adopt translations.

In our experiments in this dataset we set the variances of the horizontal and the vertical components of the translation, denoted by σ_h^2 and σ_v^2 respectively, to $\sigma_h^2 = \sigma_v^2 = \left(\frac{p_t}{3}\right)^2$, so that the translation falls in the square $[-p_t, p_t] \times [-p_t, p_t]$ with probability 99.7%. The p_t is measured in pixels and for the experiments described below, it is set to $p_t = 5$ pixels.

Experimental results

Table 3.1 shows the performance of the proposed classifier (LSVM-GSU) and the compared techniques in terms of testing accuracy for each dataset defined above; i.e., for each of the datasets D_0-D_5, where 25 training examples are used for each class. The optimization of the training parameter for the various SVM variants was performed using a line search on a 3-fold cross-validation procedure. The performance of LSVM-GSU when the training of each classifier is carried out in the original feature space is shown in row 5, and in linear subspaces in row 6. In row 6 we report both the classification performance, and in parentheses the fraction of variance that resulted in the best classification result.

The performance of the baseline linear SVM (LSVM) is shown in the second row, the performance of Power SVM (PSVM) [126] is shown in the third row, and the performance of the linear SVM extension, based on the proposed formulation, handling the noise isotropically, as in [12, 87], (LSVM-iso) is shown in the fourth row. Moreover, Fig. 3.6 shows the results of the above experimental scenarios for datasets D_0-D_5. The horizontal axis of each subfigure describes the fraction of the total variance preserved for each covariance matrix, while the vertical axis shows the respective performance



(a) (b)

Figure 3.5: MNIST "1" versus "7" experimental results using 25, 50, 100, 500, 1000, 3000, 6000 positive examples per digit. The proposed LSVM-GSU using learning linear subspaces (LSVM-GSU-S$_p$) is compared to the baseline linear SVM (LSVM), Power SVM (PSVM) [126], and a linear SVM extension which handles the uncertainty isotropically (LSVM-iso), as in [12, 87]. The fraction of variance preserved for the proposed method is (a) $p = 0.85$ (dataset D_3), (b) $p = 0.95$ (dataset D_4). Very similar results are observed for all other datasets.

of LSVM-GSU with learning in linear subspaces (LSVM-GSU-SL$_p$). Furthermore, in each subfigure, for $p = 1$ we draw the result of LSVM-GSU in the original feature space (denoted with a rhombus), the result of PSVM [126] (denoted with a circle), as well as the result of LSVM-iso [12, 87] (denoted with a star).

We report the mean, and with an error-bar show the variance of the 100 iterations. The performance of the baseline LSVM is shown with a solid line, while two dashed lines show the corresponding variance of the 100 runs. From the obtained results, we observe that the proposed LSVM-GSU with learning in linear subspaces outperforms LSVM, PSVM, and LSVM-iso for all datasets D_0-D_5. Moreover, LSVM-GSU achieves better classification results than PSVM in all datasets, and than LSVM-iso in 5 out of 6 datasets, when learning is carried out in the original feature space. Finally, all the reported results are shown to be statistically significant using the t-test [45]; significance values (p-values) were much lower than the significance level of 1%. Finally, in Fig. 3.5, we show the experimental results using various training set sizes and we observe that this does not qualitatively affect the behavior of the various compared methods.

3.5.3 Wisconsin Diagnostic Breast Cancer dataset

The Wisconsin Diagnostic Breast Cancer (WDBC) dataset [68] consists of features computed from 569 images, each belonging to one of the following two classes: *malignant* (212 instances) and *benign* (357 instances). The digitized images depict breast mass obtained by Fine Needle Aspirate (FNA) and they describe characteristics of the cell nuclei present in the image. Each feature vector is of the form

$$\mathbf{x} = (x_1, \ldots, x_{10}, s_1, \ldots, s_{10}, w_1, \ldots, w_{10})^\top \in \mathbb{R}^{30},$$

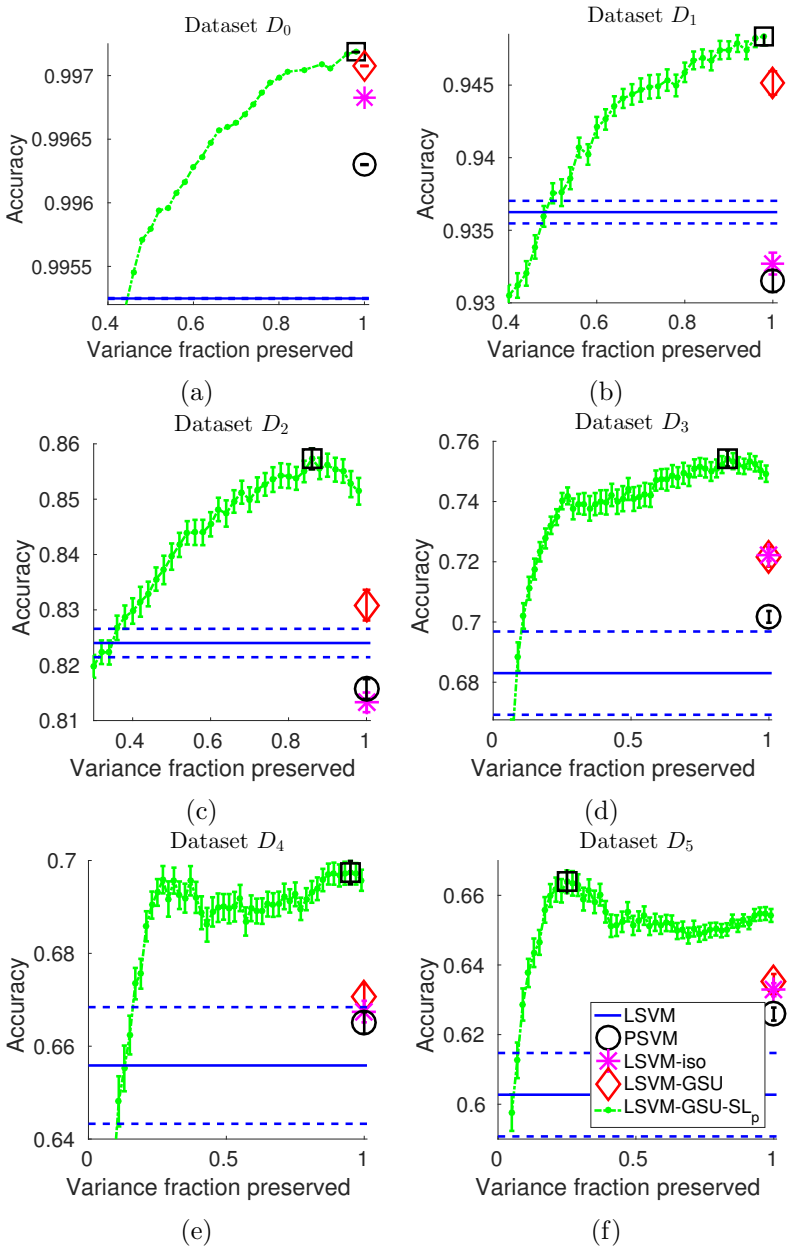


Figure 3.6: Comparisons between the proposed LSVM-GSU, the baseline LSVM, and the LSVM with isotropic noise in (a) the original MNIST dataset (D_0), and (b)-(f) the noisy generated datasets D_1-D_5.

where x_j is the mean value, s_j the standard error, and w_j the largest value of the j-th feature, $j = 1, \ldots, 10$. Ten real-valued features are computed for each cell nucleus.

Since the standard error s_i and variance σ_i^2 are connected via the relation $s_i = \frac{\sigma_i^2}{N}$, where N is the (unknown) size of the sample where standard deviation was computed, we assign to each input example a diagonal covariance matrix given by $\Sigma_i =$

$\text{diag}(\sigma_1^2, \ldots, \sigma_{10}^2, \sigma_0^2, \ldots, \sigma_0^2) \in \mathbb{S}_{++}^{30}$, where σ_0^2 is set to a small positive constant (e.g., 10^{-6}) indicating very low uncertainty for the respective features, and σ_j^2 is computed using the standard error by scaling the standard error values into the range of mean values; that is, the maximum variance is set to 80% of the range of the corresponding mean value.

The proposed algorithm is compared in terms of testing accuracy both to the baseline linear SVM (LSVM), Power SVM [126] (PSVM), and to LSVM-iso, similarly to Sect. 3.5.2. Since the original dataset does not provide a division in training and evaluation subsets, we divided the dataset randomly into a training subset (90%) and an evaluation subset (10%). The optimization of the λ parameter for all classifiers was performed using a line search on a 10-fold cross-validation procedure. We repeated the experiment 10 times and report the average results in Table 3.2. The results are statistically significant and show the superiority of LSVM-GSU. More specifically, we used the t-test [45] and obtained significance values (p-values) lower than 0.05.

Table 3.2: Comparison between the proposed LSVM-GSU, the baseline LSVM, Power SVM, and LSVM-iso.

Classifier	**Testing Accuracy**
LSVM	95.15%
PSVM [126]	96.37%
LSVM-iso (as in [12, 87])	96.53%
LSVM-GSU (proposed)	**97.14**%

3.5.4 Emotion analysis using physiological signals

Dataset and experimental setup

For evaluating the proposed method in the domain of emotional analysis using physiological signals, we used the publicly available DEAP [58] dataset, which provides EEG features of 32 participants who were recorded while watching 40 one-minute long excerpts of music videos. Three different binary classification problems were defined: the classification of low/high arousal, low/high valence and low/high liking videos.

From the EEG signals, power spectral features were extracted using the Welch method [115]. The logarithms of the spectral power from *theta* ($4-8$ Hz), *slow alpha* ($8-10$ Hz), *alpha* ($8-12$ Hz), *beta* (12-30Hz), and *gamma* (30+ Hz) bands were extracted from all 32 electrodes as features, similarly to [58]. In addition to power spectral features, the difference between the spectral power of all the symmetrical pairs of electrodes on the right and left hemisphere was extracted to measure the possible asymmetry in the brain activities due to emotional stimuli. The total number of EEG features of a video for 32 electrodes is 216. For feature selection, we used Fisher's linear discriminant similarly to [58].

Uncertainty modeling

For modeling the uncertainty of each training example, we used a well-known property of the Welch method [115] for estimating the power spectrum of a time signal. First,

Table 3.3: Comparisons between the proposed LSVM-GSU, the baseline NB, LSVM, Power SVM, and the LSVM with isotropic noise.

	Arousal		Valence		Liking	
Classifier	**ACC**	**F1**	**ACC**	**F1**	**ACC**	**F1**
NB [58]	0.620	**0.583**	0.576	0.563	0.554	0.502
LSVM	0.626	0.451	0.616	0.538	0.655	0.470
PSVM [126]	0.625	0.521	0.633	0.561	0.651	0.522
LSVM-iso [12, 87]	0.645	0.531	0.645	0.603	0.658	0.530
LSVM-GSU	**0.659**	0.551	**0.650**	**0.609**	**0.666**	**0.539**

the time signal was divided into (overlapping or non-overlapping) windows, where the periodogram was computed for each window. Then the resulting frequency-domain values were averaged over all windows. Besides these mean values, that are the desired outcomes of the Welch method, we also computed the variances, and, thus, each 216-element vector was assigned with a diagonal covariance matrix.

Experimental results

Table 3.3 shows the performance of the proposed linear SVM-GSU (LSVM-GSU) in terms of accuracy and F1 score for each target class in comparison to LSVM, PSVM [126], and LSVM-iso, similarly to Sect. 3.5.2 and 3.5.3, as well as the Naive Bayesian (NB) classifier used in [58]. For each participant, the F1 measure was used to evaluate the performance of emotion classification in a leave-one-out cross validation scheme. At each step of the cross validation, one video was used as the test-set and the rest were used for training. For optimizing the λ parameter of the various SVM classifiers, we used a line search on a 3-fold cross-validation procedure.

From the obtained results, we observe that the proposed algorithm achieved better classification performance than LSVM, PSVM, LSVM-iso, as well as the NB classifier used in [58] for all three classes, in terms of testing accuracy, and for the two out of three classes in terms of F1 score.

3.5.5 TV News Channel Commercial Detection

Dataset and experimental setup

The proposed algorithm is evaluated for the problem of detection of advertisements in TV news videos using the very large publicly available dataset of [112]. This dataset comprises 120 hours of TV news broadcasts from CNN, CNNIBN, NDTV, and TIMES NOW (approximately $22k$, $33k$, $17k$, and $39k$ videos, respectively). The authors of [112] used various low-level audio and static-, motion-, and text-based visual features, to extract and provide a 4125-dimensional representation for each video, that includes the variance values for 24 of the above features. For a detailed description of the dataset, see [112].

Table 3.4: Comparisons between the proposed LSVM-GSU and the baseline LSVM, similarly to [112].

		Training							
		CNN		CNNIBN		NDTV		TIMES NOW	
		LSVM	LSVM-GSU	LSVM	LSVM-GSU	LSVM	LSVM-GSU	LSVM	LSVM-GSU
Testing	CNN	0.7799	**0.9589**	0.7799	**0.8050**	0.7799	**0.8113**	0.7799	**0.9226**
	CNNIBN	0.7915	**0.8836**	0.7915	**0.9215**	0.7915	**0.8978**	0.7915	**0.8611**
	NDTV	0.8484	**0.9248**	0.8484	**0.8565**	0.8484	**0.9709**	0.8484	**0.8823**
	TIMES NOW	0.7809	**0.9461**	0.7809	**0.7863**	**0.7809**	0.7493	0.7809	**0.9421**

Uncertainty modeling

This dataset represents a real-world case where uncertainty information is given only for a few dimensions of the feature space. In this case we model the covariance matrix of each input example as a low-rank diagonal matrix, whose non-zero variance values correspond to the dimensions for which uncertainty is provided. Each such matrix corresponds to a Gaussian with non-zero variance along the few specific given dimensions. Since the information about the input variance is provided just for the 24 of the 4125 features, there is no natural way of estimating a single variance value, i.e., an isotropic covariance matrix, for each training example.

Experimental results

Table 3.4 shows the performance of the proposed linear SVM-GSU (LSVM-GSU) in terms of F1 score in comparison to LSVM, similarly to [112]. As discussed above, since methods that model the uncertainty isotropically (such as [12, 87, 126]), are not applicable in this dataset, we experimented on this dataset using only the proposed algorithm and the standard linear SVM. Following the protocol of [112], we did cross-dataset training and testing. For optimizing the λ parameter of both LSVM and LSVM-GSU we used a line search on a 3-fold cross-validation procedure. From the obtained results, we observe that the proposed algorithm achieved considerably better classification than LSVM in almost all cases (more than 10% relative boost on average).

3.5.6 Video Event Detection

Dataset and experimental setup

In our experiments on video event detection we used datasets from the challenging TRECVID Multimedia Event Detection (MED) task [83]. For training, we used the MED 2015 training dataset consisting of the "pre-specified" (PS) video subset (2000 videos, 80 hours) and the "event background" (Event-BG) video subset (5000 videos, 200 hours). For testing, we used the large-scale "MED14Test" dataset [83, 52] ($\sim$ 24K videos, 850 hours). Each video in the above datasets belongs to, either one of 20 target event classes, or to the "rest of the world" (background) class. More specifically, in the training set, 100 positive and 5000 negative samples are available for each event class, while the evaluation set includes only a small number of positive (e.g., only 16 positives for event E021, and 28 for E031) and approximately 24K negative videos.

For video representation, approximately 2 keyframes per second were extracted from each video. Each keyframe was represented using the last hidden layer of a pre-trained deep convolutional neural network (DCNN). More specifically, a 22-layer inception style network, trained according to the GoogLeNet architecture [102], was used. This network had been trained on various selections of the ImageNet "Fall 2011" dataset and provides scores for 5055 concepts [90].

Uncertainty modeling

Let us now define a set $\mathcal{X}$ of ℓ annotated random vectors representing the aforementioned video-level feature vectors. Each random vector is assumed to be distributed normally; i.e., for the random vector representing the i-th video, $\mathbf{X}_i$, we have $\mathbf{X}_i \sim \mathcal{N}(\mathbf{x}_i, \Sigma_i)$. That is, $\mathcal{X} = \{(\mathbf{x}_i, \Sigma_i, y_i) \colon \mathbf{x}_i \in \mathbb{R}^n, \Sigma_i \in \mathbb{S}^n_{++}, y_i \in \{\pm 1\},\ i = 1, \ldots, \ell\}$. For each random vector $\mathbf{X}_i$, a number, N_i, of observations, $\{\mathbf{x}_i^t \in \mathbb{R}^n \colon t = 1, \ldots, N_i\}$ are available (these are the keyframe-level vectors that have been computed). Then, the sample mean vector and the sample covariance matrix of $\mathbf{X}_i$ are computed. However, the number of observations per each video that are available for our dataset is in most cases much lower than the dimensionality of the input space. Consequently, the covariance matrices that arise are typically low-rank; i.e. $\text{rank}(\Sigma_i) \leq N_i \leq n$. To overcome this issue, we assumed that the desired covariance matrices are diagonal. That is, we require that the covariance matrix of the i-th training example is given by $\widehat{\Sigma_i} = \text{diag}\left(\hat{\sigma}_i^1, \ldots, \hat{\sigma}_i^n\right)$, such that the squared Frobenious norm of the difference $\Sigma_i - \widehat{\Sigma_i}$ is minimum. That is, the estimator covariance matrix $\widehat{\Sigma_i}$ must be equal to the diagonal part of the sample covariance matrix Σ_i, i.e. $\widehat{\Sigma_i} = \text{diag}\left(\sigma_i^1, \ldots, \sigma_i^n\right)$. We note that, using this approximation approach, the covariance matrices are diagonal but anisotropic and different for each training input example. This is in contrast with other methods (e.g. [126, 12, 87]) that assume more restrictive modeling for the uncertainty; e.g., isotropic noise for each training sample.

Experimental results

We experimented using two different feature configurations. First, we used the mean vectors and covariance matrices as computed using the method discussed above (Sect. 3.5.6). Furthermore, in order to investigate the role of variances in learning with baseline LSVM, we constructed mean vectors and covariance matrices as shown in Table 3.6, where σ_0 is typically set to a small positive constant (e.g., 10^{-6}) indicating very low uncertainty for the respective features.

For both feature configurations, Table 3.5 shows the performance of the proposed linear SVM-GSU (LSVM-GSU) in terms of average precision (AP) [83, 108] for each target event in comparison with LSVM, PSVM [126], and LSVM-iso approaches. Moreover, for each dataset, the mean average precision (MAP) across all target events is reported. The optimization of the λ parameter for the various SVMs was performed using a line search on a 10-fold cross-validation procedure. The bold-faced numbers indicate the best result achieved for each event class. We also report the results of the McNemar [80, 36], statistical significance tests. A $*$ denotes statistically significant differences between the proposed LSVM-GSU and baseline LSVM, a $\diamond$ denotes

Table 3.5: Event detection performance (AP and MAP) of the linear SVM-GSU compared to the baseline linear SVM, Power SVM [126], and a LSVM extension for handling isotropic uncertainty (as in [12, 87]) using the MED15 (for training) and MED14Test (for testing) datasets.

Event Class	**Feature Configuration 1** (5055-**D**)				
	LSVM	**PSVM [126]**	**LSVM-iso [12, 87]**	**LSVM-GSU (proposed)**	**McNemar Tests**
E021	0.0483	0.0510	0.0500	**0.0515**	$*, \diamond, \sim$
E022	0.0227	0.0310	**0.0350**	0.0277	$*, \diamond, \sim$
E023	0.4159	0.4515	**0.6059**	0.6057	$*, \diamond$
E024	0.0071	0.0081	0.0097	**0.0105**	$\diamond$
E025	0.0052	0.0052	**0.0074**	0.0068	
E026	0.0457	0.0459	0.0606	**0.0608**	$\diamond$
E027	**0.1319**	0.1424	0.1174	0.1219	$*, \diamond, \sim$
E028	0.4242	0.4125	0.3819	**0.4335**	$*, \diamond, \sim$
E029	0.0812	0.0914	**0.1793**	0.1791	$\diamond$
E030	0.0516	0.0551	0.0877	**0.0884**	
E031	0.4416	0.4425	0.4480	**0.4796**	$*, \diamond, \sim$
E032	0.0280	0.0400	0.0870	**0.1196**	$*, \diamond, \sim$
E033	0.3483	0.3614	0.3901	**0.4187**	$*, \sim$
E034	0.0583	0.0588	0.0599	**0.0614**	$\diamond$
E035	0.3330	0.3419	**0.3500**	0.3369	$*, \diamond, \sim$
E036	**0.0894**	0.0748	0.0695	0.0704	$\diamond$
E037	0.0884	0.0880	**0.1981**	0.1968	$*, \diamond, \sim$
E038	0.0261	0.0241	0.0212	**0.0291**	$\diamond$
E039	0.2677	0.2698	**0.2959**	0.2757	$*, \diamond, \sim$
E040	0.0421	0.0315	0.0375	**0.0377**	$*, \diamond$
MAP	0.1478	0.1513	0.1746	**0.1806**	–

Event Class	**Feature Configuration 2** (10110-**D**)				
	LSVM	**PSVM [126]**	**LSVM-iso [12, 87]**	**LSVM-GSU (proposed)**	**McNemar Tests**
E021	0.0829	0.0834	**0.1074**	0.0778	$\diamond, \sim$
E022	0.0674	0.0773	0.1023	**0.1429**	$*, \diamond, \sim$
E023	0.7050	0.7236	0.7802	**0.7943**	$*, \diamond, \sim$
E024	0.0187	0.0223	**0.0394**	0.0367	$*$
E025	**0.0219**	0.0245	0.0161	0.0135	$\diamond$
E026	0.0731	0.0745	0.0976	**0.1109**	$*, \diamond, \sim$
E027	0.1152	0.0133	0.1254	**0.1812**	$*, \diamond, \sim$
E028	0.1863	0.2214	**0.2700**	0.2278	$*, \diamond, \sim$
E029	0.2046	0.1987	**0.2149**	0.1999	$*, \diamond, \sim$
E030	0.1001	0.1276	0.1596	**0.1774**	$*, \diamond, \sim$
E031	0.7595	0.7599	0.7422	**0.7697**	$*, \diamond, \sim$
E032	0.0989	0.1011	0.1290	**0.1292**	$*, \diamond$
E033	0.4571	0.4789	0.5091	**0.5164**	$*$
E034	0.3207	0.3214	0.3200	**0.3380**	$*, \diamond, \sim$
E035	**0.3516**	0.3419	0.3252	0.3059	$*, \diamond$
E036	0.1156	0.1186	0.1064	**0.1288**	$*, \diamond, \sim$
E037	0.1169	0.1257	0.1598	**0.1629**	$*, \diamond, \sim$
E038	**0.0558**	0.0498	0.0557	0.0539	$\diamond$
E039	0.4188	0.4219	**0.4349**	0.4271	$*, \diamond, \sim$
E040	0.0837	0.0889	0.0856	**0.0902**	$*, \diamond$
MAP	0.2177	0.2187	0.2390	**0.2442**	–

statistically significant differences between LSVM-GSU and PSVM, and a $\sim$ denotes statistically significant differences between LSVM-GSU and LSVM-iso.

Table 3.6: Mean vector and covariance matrix of the i-th example for feature configurations 1 and 2 of the video event detection experiments.

Configuration 1	$\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n})^\top \in \mathbb{R}^n$ $\Sigma_i, = \text{diag}\left(\sigma_i^1, \ldots, \sigma_i^n\right) \in \mathbb{S}_{++}^n$
Configuration 2	$\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n}, \sigma_i^1, \ldots, \sigma_i^n)^\top \in \mathbb{R}^{2n}$ $\Sigma_i, = \text{diag}\left(\sigma_i^1, \ldots, \sigma_i^n, \sigma_0, \ldots, \sigma_0\right) \in \mathbb{S}_{++}^{2n}$

From the obtained results, we observe that the proposed algorithm achieved better detection performance than LSVM, PSVM, and LSVM-iso, in both feature configurations. For feature configuration 1, the proposed LSVM-GSU achieved a relative boost of 22.2% compared to the baseline standard LSVM and 19.4% compared to Power SVM, while for feature configuration 2 respective relative boosts of 12.7% and 11.7%, respectively, in terms of MAP. It is worth noting that configuration 2 exhibits much better performance compared to configuration 1 (for all the compared methods), since it uses the extra knowledge of the variance as additional dimensions in its feature representation scheme. Finally, we also experimented using directly the samples from which the covariance matrix of each example was estimated and obtained inferior results; that is, a MAP of 10.15%, compared to LSVM's 14.78% and 18.06% of the proposed SVM-GSU.

3.6 Conclusion

In this chapter we proposed a novel linear classifier that efficiently exploits uncertainty in its input under the SVM paradigm. The proposed SVM-GSU was evaluated on synthetic data and on five publicly available datasets; namely, the MNIST dataset of handwritten digits, the WDBC, the DEAP for emotion analysis, the TV News Commercial Detection dataset and TRECVID MED for the problem of video event detection. For each of the above datasets and problems, either uncertainty information (e.g., variance for each example and for all or some of the input space dimensions) was part of the original dataset, or a method for modeling and estimating the uncertainty of each training example was proposed. As shown in the experiments, SVM-GSU efficiently takes input uncertainty into consideration and achieves better detection or classification performance than standard SVM, previous SVM extensions that model uncertainty isotropically, and other state of the art methods. Finally, we plan to investigate the kernalization of the proposed algorithm and the extensions of it for the problem of regression under Gaussian input uncertainty.

Chapter 4

Kernel Maximum Margin Classifier for Learning from Uncertain Data

Contents

In this chapter we develop a non-liner version of the linear SVM-GSU presented in the previous chapter using the well-known RBF kernel function. We proceed to this development under the assumption of isotropic input uncertainty. For doing so, we recast the optimization problem of linear SVM-GSU as a variational calculus minimization problem; that is, the original optimization problem is rewritten as a problem of minimizing an equivalent (objective) functional, and, thus, instead of looking for a separating hyperplane (i.e., its parameters) in the original input feature space, we look for a minimizer function that lives in a richer, higher-dimensional (in our case infinite-dimensional) space. We prove that the above functional is such that its minimizer can be represented as a finite linear combination of kernel products (in our case using the RBF kernel function). Additionally, due to the convexity of our objective functional, we can efficiently solve the problem using an appropriate SGD algorithm (similarly to the linear case), i.e., the Pegasos algorithm [96].

Then, we combine the proposed kernel classifier with the previously proposed Relevance Degree SVM (RD-SVM) [108, 107]. In RD-SVM each training example is associated with a confidence value (called relevance degree) indicating the degree of relevance of the respective training example with the class that it is related. This is essentially a method that handles uncertainty in the truth labels, and combined with the proposed KSVM-iGSU provide a methodology for handling uncertainty both in label and feature representation.

Finally, we apply the proposed methods to two challenging problems of video understanding and indexing: video event detection and video aesthetic quality assessment. For video event detection, we experimented on the challenging TRECVID MED 2014 dataset, using a limited number of positive and related samples for training. Another challenging video dataset, the CERTH-ITI-VAQ700, was used for experimenting on video aesthetic quality assessment. The experimental results of our methods show considerable performance improvement in comparison to the state-of-the-art learning methods that are used in the literature for the video event detection and aesthetic quality assessment problems.

4.1 Kernel SVM with Isotropic Gaussian Sample Uncertainty (KSVM-iGSU)

Let us first revisit linear SVM-GSU's [110] optimization problem as proposed in the previous chapter. LSVM-GSU is a maximum-margin classifier that takes as input training data that are described not solely by a set of feature representations, i.e. a set of vectors $\mathbf{x}_i$ in some n-dimensional space, but rather by a set of multi-variate Gaussian distributions which model the uncertainty of each training example. In this chapter we will consider only isotropic Gaussian uncertainty. That is, every training datum is characterized by a mean vector $\mathbf{x}_i \in \mathbb{R}^n$ and an isotropic covariance matrix, i.e. a scalar multiple of the identity matrix, $\Sigma_i = \sigma_i^2 I_n \in \mathbb{S}_{++}^n$[1]. We will denote this classifier as linear SVM with isotropic Gaussian Sample Uncertainty (LSVM-iGSU). LSVM-iGSU is obtained by minimizing, with respect to $\mathbf{w}$, b, the objective function $\mathcal{J}\colon \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ given by

$$\mathcal{J}(\mathbf{w}, b) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{\ell}\sum_{i=1}^{\ell} \mathcal{L}\big(\mathbf{w}, b; (\mathbf{x}_i, \sigma_i^2 I_n, y_i)\big), \tag{4.1}$$

where ℓ is the number of training data, $\mathbf{w}^\top\mathbf{x}+b = 0$ denotes the separating hyperplane, and the loss $\mathcal{L}\colon \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is given by

$$\mathcal{L}(\mathbf{w}, b) = \frac{d_{\mathbf{x}_i}}{2}\left[\operatorname{erf}\left(\frac{d_{\mathbf{x}_i}}{d_{\Sigma_i}}\right) + 1\right] + \frac{d_{\Sigma_i}}{2\sqrt{\pi}}\exp\left(-\frac{d_{\mathbf{x}_i}^2}{d_{\Sigma_i}^2}\right), \tag{4.2}$$

where $d_{\mathbf{x}_i} = 1 - y_i\left(\mathbf{w}^\top\mathbf{x}_i + b\right)$, $d_{\Sigma_i} = \sqrt{2\sigma_i^2\|\mathbf{w}\|^2}$, $\mathbf{x}_i$ and $\sigma_i^2 I_n$ denote the mean vector and the covariance matrix of the i-th input entity (Gaussian distribution), respectively, y_i denotes its ground-truth label, and $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,\mathrm{d}t$ denotes the error function.

As discussed in the previous chapter, (4.1) is convex and thus a (global) optimal solution $(\mathbf{w}, b)$ can be obtained using a gradient descent algorithm. The resulting (linear) decision function $f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x}+b$ is used in the testing phase for classifying an unseen sample similarly to the standard linear SVM algorithm [17]; that is, according to the distance between the testing sample and the separating hyperplane, without

[1] $\mathbb{S}_{++}^n$ denotes the convex cone of all symmetric positive definite $n \times n$ matrices with entries in $\mathbb{R}$. I_n denotes the identity matrix of order n.

taking into account any uncertainty estimates that could be made for the testing sample representation.

The optimization problem discussed in the previous section can be recast as a variational calculus problem of finding the function f that minimizes the functional $\Phi[f]$:

$$\min_{f\in\mathcal{H}} \Phi[f], \tag{4.3}$$

where the functional $\Phi[f]$ is given by

$$\Phi[f] = \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 + \frac{1}{\ell}\sum_{i=1}^{\ell}\left[\frac{d_{\mathbf{x}_i}}{2}\left[\operatorname{erf}\left(\frac{d_{\mathbf{x}_i}}{d_{\Sigma_i}}\right)+1\right] + \frac{d_{\Sigma_i}}{2\sqrt{\pi}}\exp\left(-\frac{d_{\mathbf{x}_i}^2}{d_{\Sigma_i}^2}\right)\right], \tag{4.4}$$

where $d_{\mathbf{x}_i} = 1 - y_i\left(f(\mathbf{x}_i)+b\right)$, $d_{\Sigma_i} = \sqrt{2\sigma_i^2\|f\|_{\mathcal{H}}^2}$, λ is a regularization parameter and f belongs to a Reproducing Kernel Hilbert Space (RKHS), $\mathcal{H}$, with associated kernel k. Using a generalized semi-parametric version [93] of the representer theorem [57], it can be shown that the minimizer of the above functional admits a solution of the form

$$f(\mathbf{x}) = \sum_{i=1}^{\ell}\alpha_i k(\mathbf{x},\mathbf{x}_i) - b, \tag{4.5}$$

where $b\in\mathbb{R}$, $\alpha_i\in\mathbb{R}$, $i=1,\ldots,\ell$.

Using the reproducing property, we have

$$\|f\|_{\mathcal{H}}^2 = \langle f,f\rangle_{\mathcal{H}} = \left\langle \sum_{i=1}^{l}\alpha_i k(\cdot,\mathbf{x}_i), \sum_{j=1}^{l}\alpha_j k(\cdot,\mathbf{x}_j)\right\rangle_{\mathcal{H}} = \boldsymbol{\alpha}^\top K\boldsymbol{\alpha}, \tag{4.6}$$

where K is the kernel matrix, i.e. the symmetric positive definite $\ell\times\ell$ matrix defined as $K = (k(\mathbf{x}_i,\mathbf{x}_j))_{i,j=1}^{\ell}$, and $\boldsymbol{\alpha} = (\alpha_1,\cdots,\alpha_l)^\top$. Moreover, we observe that $f(\mathbf{x}_i) = \sum_{j=1}^{\ell}\alpha_j k(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{K}_i^\top\boldsymbol{\alpha}$, where $\mathbf{K}_i$ denotes the i-th column of the kernel matrix K. Then, the objective function $\mathcal{J}_{\mathcal{H}}\colon\mathbb{R}^\ell\times\mathbb{R}\to\mathbb{R}$ is given by

$$\mathcal{J}_{\mathcal{H}}(\boldsymbol{\alpha},b) = \frac{\lambda}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} + \frac{1}{\ell}\sum_{i=1}^{\ell}\left[\frac{d_{\mathbf{x}_i}}{2}\left[\operatorname{erf}\left(\frac{d_{\mathbf{x}_i}}{d_{\Sigma_i}}\right)+1\right] + \frac{d_{\Sigma_i}}{2\sqrt{\pi}}\exp\left(-\frac{d_{\mathbf{x}_i}^2}{d_{\Sigma_i}^2}\right),\right], \tag{4.7}$$

where $d_{\mathbf{x}_i} = 1 - y_i\left(\mathbf{K}_i^\top\boldsymbol{\alpha}+b\right)$, $d_{\Sigma_i} = \sqrt{2\sigma_i^2\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}}$. We (jointly) minimize the above convex[2] objective function with respect to $\boldsymbol{\alpha}$ and b using a stochastic gradient descent method, which we present in section 4.2.1. The first order derivatives of the objective function with respect to the optimization variables $\boldsymbol{\alpha}$ and b are given[3], respectively, as follows

$$\frac{\partial\mathcal{J}_{\mathcal{H}}}{\partial\boldsymbol{\alpha}} = \lambda K\boldsymbol{\alpha} + \frac{1}{\ell}\sum_{i=1}^{\ell}\left[\frac{\sigma_i^2\exp\left(-\frac{d_{\mathbf{x}_i}^2}{d_{\Sigma_i}^2}\right)}{\sqrt{\pi}d_{\Sigma_i}}K\boldsymbol{\alpha} - \frac{\mathbf{K}_i}{2}\left[\operatorname{erf}\left(\frac{d_{\mathbf{x}_i}}{d_{\Sigma_i}}\right)+1\right]\right], \tag{4.8}$$

[2]Convexity can be shown using Theorem 2 proved in B.

[3]Their derivation is omitted, as it is technical but straightforward.

and

$$\frac{\partial \mathcal{J}_{\mathcal{H}}}{\partial b} = -\frac{1}{2\ell}\sum_{i=1}^{\ell}\left[\operatorname{erf}\left(\frac{d_{\mathbf{x}_i}}{d_{\Sigma_i}}\right)+1\right]. \tag{4.9}$$

Since $J_{\mathcal{H}}$ is a convex function on $\mathbb{R}^{\ell}\times\mathbb{R}$, the proposed SGD solver leads to a global optimal solution; that is, at a pair $(\boldsymbol{\alpha}, b)$ such that the decision function given in the form of (4.5) minimizes the functional (4.4). We call this classifier kernel SVM-iGSU (KSVM-iGSU).

4.2 Relevance Degree KSVM-iGSU

Motivated by [108], we reformulate the optimization problem in (4.3)-(4.4) such that a different penalty parameter $c_i \in (0, 1]$ (hereafter called as relevance degree) is introduced to each input datum. That is, the functional $\Phi[f]$ of (4.4) is now given by

$$\Phi[f] = \frac{\lambda}{2}\|f\|^2_{\mathcal{H}} + \frac{1}{\ell}\sum_{i=1}^{\ell} c_i\left[\frac{d_{\mathbf{x}_i}}{2}\left[\operatorname{erf}\left(\frac{d_{\mathbf{x}_i}}{d_{\Sigma_i}}\right)+1\right] + \frac{d_{\Sigma_i}}{2\sqrt{\pi}}\exp\left(-\frac{d^2_{\mathbf{x}_i}}{d^2_{\Sigma_i}}\right)\right]. \tag{4.10}$$

It is worth noting that the relevance degree, i.e., the penalty parameter c_i, is a user-defined parameter and, as such, it is treated similarly to the other classifier's parameters (e.g., the regularization parameter λ). This means that it can be selected using some cross-validation procedure, or it can merely be given based on prior knowledge inspired by the problem in hand (e.g., see Sect. 4.3.1). To solve $\min_{f\in\mathcal{H}}\Phi[f]$, we follow a similar path as in Sect. 4.1, and we arrive at the following convex objective function

$$\mathcal{J}_{\mathcal{H}}(\boldsymbol{\alpha}, b) = \frac{\lambda}{2}\boldsymbol{\alpha}^{\top}K\boldsymbol{\alpha} + \frac{1}{\ell}\sum_{i=1}^{\ell} c_i\left[\frac{d_{\mathbf{x}_i}}{2}\left[\operatorname{erf}\left(\frac{d_{\mathbf{x}_i}}{d_{\Sigma_i}}\right)+1\right] + \frac{d_{\Sigma_i}}{2\sqrt{\pi}}\exp\left(-\frac{d^2_{\mathbf{x}_i}}{d^2_{\Sigma_i}}\right)\right], \tag{4.11}$$

which we again minimize using the proposed SGD algorithm. The (global) optimal solution $(\boldsymbol{\alpha}, b)$ determines the decision function given in the form of (4.5). The new extension of KSVM-iGSU obtained in this way is hereafter referred to as a Relevance Degree KSVM-iGSU (RD-KSVM-iGSU).

4.2.1 A stochastic gradient descent solver

Similarly to the linear case (see Sect. 3.4), motivated by the Pegasos algorithm [95], we modify and present a stochastic sub-gradient descent algorithm for solving KSVM-iGSU in order to efficiently mount scalability requirements. As discussed in the previous chapter, Pegasos is a well-studied algorithm [96, 55] providing both state-of-the-art classification performance and great scalability. Pegasos requires $\tilde{O}(1/\epsilon)$ number of iterations in order to obtain a solution of accuracy ϵ, in contrast to previous analyses of stochastic gradient descent methods that require $\tilde{O}(d/(\lambda\epsilon))$ iterations, where d is a bound on the number of non-zero features in each example. Since the run-time does not depend directly on the size of the training set, the resulting algorithm is especially

Algorithm 2 A stochastic sub-gradient descent algorithm for solving KSVM-iGSU.

1: **Inputs:**
   $\mathcal{X}$, λ, T, k
2: **Initialize:**
   $b^{(1)} = 0$, $\boldsymbol{\alpha}^{(1)}$ such that $\|\boldsymbol{\alpha}^{(1)}\| \leq \frac{1}{\sqrt{\lambda}}$
3: **for** $t = 1, 2, \ldots, T$ **do**
4: Choose $\mathcal{X}_t \subseteq \mathcal{X}$, where $|\mathcal{X}_t| = k$
5: Set $\eta_t = \frac{1}{\lambda t}$
6: $\boldsymbol{\alpha}^{(t+1)} \leftarrow \boldsymbol{\alpha}^{(t)} - \frac{\eta_t}{k} \frac{\partial J}{\partial \boldsymbol{\alpha}}$
7: $\boldsymbol{\alpha}^{(t+1)} \leftarrow \min\left(1, \frac{1/\sqrt{\lambda}}{\|\boldsymbol{\alpha}^{(t+1)}\|}\right)\boldsymbol{\alpha}^{(t+1)}$
8: $b^{(t+1)} \leftarrow b^{(t)} - \frac{\eta_t}{k} \frac{\partial J}{\partial b}$
9: **end for**

suited for learning from large datasets. Algorithm 2 describes the proposed method in pseudocode.

We note that we can distinguish three variants of the algorithm: (a) a pure SGD approach, where only one training example is used in each iteration for computing the objective function, (b) a mini-batch variant, where $1 < k < \ell$ examples are used, and (c) a standard gradient descent (GD) one, where the whole set ($k = \ell$) of the training examples contribute to the learning process. For the sake of training speed, in our experiments we used a mini-batch SGD variant using a subset (of cardinality approximately equal to the 1/10 of the original training set) for computing the loss and its derivatives in each iteration.

4.3 Experiments

4.3.1 Application of KSVM-iGSU and RD-KSVM-iGSU to Video Event Detection

Problem statement, video representation, and uncertainty

As discussed in the previous chapter, high-level video event detection is about deciding whether a certain video depicts a given event or not. A typical high-level (or complex) event is an interaction between humans, or between humans and physical objects [53]. Similarly to the previous chapter (see Sect. 3.5.6), for our experiments two keyframes per second are extracted at regular time intervals from each video. Each keyframe is represented using the last hidden layer of a pre-trained Deep Convolutional Neural Network (DCNN). More specifically, a 16-layer pre-trained deep ConvNet network provided in [98] is used. This network had been trained on the ImageNet data [25], providing scores for 1000 ImageNet concepts; thus, each keyframe has a 1000-element vector representation. Then, the typical procedure followed in state-of-the-art event detection systems includes the computation of a video-level representation for each video by taking the average of the corresponding keyframe-level representations [123, 39, 18, 13].

Moreover, we estimated the uncertainty of each training video similarly to Sect. 3.5.6. That is, let $\mathcal{X}$ be a set of l annotated random vectors representing the aforementioned video-level model vectors. We assume that each random vector is distributed normally; i.e., for the random vector representing the i-th video, $\mathbf{X}_i$, we have $\mathbf{X}_i \sim \mathcal{N}(\mathbf{x}_i, \Sigma_i)$. Also, for each random vector $\mathbf{X}_i$, a number, N_i, of observations, $\{\mathbf{x}_i^t \in \mathbb{R}^n \colon t = 1, \ldots, N_i\}$ is available; these are the keyframe-level model vectors that have been computed. Then, the mean vector and the covariance matrix of $\mathbf{X}_i$ are computed respectively as follows

$$\mathbf{x}_i = \frac{1}{N_i} \sum_{t=1}^{N_i} \mathbf{x}_i^t, \quad \Sigma_i = \frac{\sum_{t=1}^{N_i} (\mathbf{x}_i^t - \mathbf{x}_i)(\mathbf{x}_i^t - \mathbf{x}_i)^\top}{N_i - 1}. \tag{4.12}$$

Now, due to the assumption for isotropic covariance matrices, we approximate the above covariance matrices as multiples of the identity matrix, i.e. $\widehat{\Sigma_i} = \sigma_i^2 I_n$ by minimizing the squared Frobenious norm of the difference $\Sigma_i - \widehat{\Sigma_i}$ with respect to σ_i^2. It can be shown (by using simple matrix algebra [37]) that for this it suffices to set σ_i^2 equal to the mean value of the elements of the main diagonal of Σ_i.

Dataset and evaluation measures

Similarly to Sect. 3.5.6, the proposed algorithms are tested on a subset of the large video dataset of the TRECVID Multimedia Event Detection (MED) 2014 benchmarking activity [84]. Similarly to our previous works [109, 110], we use only the training portion of the TRECVID MED 2014 task dataset, which provides ground-truth information for 30 complex event classes, since for the corresponding evaluation set of the original TRECVID task there is no ground-truth data available. Hereafter, we refer to the aforementioned ground-truth-annotated dataset as MED14 and we divide it into a training subset, consisting of 50 positive and 25 related (near-miss) samples per event class, together with 2496 background samples (i.e. videos that are negative examples for all the event classes), and an evaluation subset consisting of approximately 50 positive and 25 related samples per event class, along with another 2496 background samples.

For assessing the detection performance of each trained event detector, the average precision (AP) [88] measure is utilized, while for measuring the detection performance of a classifier across all the event classes we use the mean average precision (MAP), as is typically the case in the video event detection literature, e.g. [34, 84, 108].

Experimental results and comparisons

The proposed KSVM-iGSU and RD-KSVM-iGSU are compared to standard kernel SVM (KSVM), LSVM-iGSU [110] and RD-KSVM [108]. We note here that for the problem of video event detection (and especially when only a few positive training samples are available), kernel SVM is the state-of-the-art approach [18, 13], while, when also a few related samples are available, RD-KSVM leads to state-of-the-art detection performance [108]. We experimented on the problem of learning from 10 positive examples per each event class, together with 5 related samples, that are drawn from the

set of 25 related samples provided for each event class following the method presented in [108]; i.e., the 5 nearest to the median of all 25 related samples were kept for training both RD-KSVM and RD-SVM-iGSU. Also, we randomly chose 70 negative samples for each event class, while we repeated each experiment 10 times. That is, for each different experimental scenario, the obtained performance of each classifier (KSVM, RD-KSVM, LSVM-iGSU, KSVM-iGSU, and RD-SVM-iGSU) is averaged over 10 iterations, for each of which 10 positive samples have been randomly selected from the pool of 50 positive samples that are available in our training dataset for each target event class.

For all the above experimental scenarios where a kernel classifier is used, the radial basis function (RBF) kernel has been used. Training parameters (C for LSVM-iGSU; C, γ for KSVM, KSVM-iGSU; and C, γ, and c for RD-KSVM, RD-KSVM-iGSU) are obtained via cross-validation. For C, γ, a 10-fold cross-validation procedure (grid search) is performed with C, γ being searched in the range $\{2^{-16}, 2^{-15}, \ldots, 2^2, 2^3\}$. For c, an approach similar to that presented in [108] is followed. That is, related samples are initially treated as true positive and true negative ones (in two separate cross-validation processes) and C, γ are optimized as described above; then, by examining the minimum cross-validation errors of the two above processes, we automatically choose whether to treat the related samples as weighted positive or weighted negative ones, and also fix the value of C to the corresponding optimal value. Using this C, we proceed with a new cross-validation process (again grid search) for finding the optimal γ, c pair (where c is searched in the range $[0.01, 1.00]$ with a step of 0.05).

Table 5.1 shows the performance of the proposed KSVM-iGSU and RD-KSVM-iGSU, compared to LSVM-iGSU [110], the standard KSVM, and the RD-KSVM [108], respectively, in terms of average precision (AP), for each target event, and mean AP (MAP), across all target events. Bold-faced values indicate the best performance for each event class. We can see that LSVM-iGSU, whose improved performance over the standard linear SVM was studied extensively in [110], cannot outperform the kernel methods that are typically used for the video event detection problem, achieving a MAP of 0.1761. Without using any related samples, KSVM-iGSU that takes into account the input uncertainty, outperformed the standard kernel SVM for 25 out of 30 target event classes, achieving a MAP of 0.2527 in comparison to KSVM's 0.2128 (relative boost of 18.75%). Moreover, when related samples were used for training, the proposed RD-KSVM-iGSU outperformed the baseline RD-KSVM for 27 out of 30 target event classes, achieving a MAP of 0.2730, in comparison to RD-KSVM's 0.2218 (i.e. a relative boost of 23.08%). This RD-KSVM-iGSU result also represents a 8% relative improvement (MAP of 0.2730 versus 0.2527) in comparison to KSVM-iGSU, which does not take advantage of related video samples during training. The above results suggest that using uncertainty for training video event detectors leads to improved results, while the additional exploitation of related samples can further improve event detection performance.

Finally, in Fig. 4.1 we present indicative results of the proposed RD-KSVM-iGSU in comparison with the baseline RD-KSVM [108] for four event classes, showing the top-5 videos each classifier retrieved. Green borders around frames indicate correct detection results, while red ones indicate false detection. These indicative results illustrate the

Event Class	LSVM-iGSU (see Chap. 3)	KSVM (e.g. [39, 18])	KSVM-iGSU (proposed)	RD-KSVM [108]	RD-KSVM-iGSU (proposed)
E021	0.1741	0.1763	0.1923	0.1823	**0.2167**
E022	0.1847	0.1903	0.2495	0.2009	**0.2604**
E023	0.4832	0.5665	0.6361	0.5435	**0.6432**
E024	0.0536	0.0482	**0.0667**	0.0489	0.0549
E025	0.0117	0.0210	0.0257	0.0200	**0.0287**
E026	0.1002	0.1388	0.1530	0.1385	**0.1701**
E027	0.1600	0.2882	**0.4162**	0.2899	0.4002
E028	0.2030	0.2234	0.2338	0.2250	**0.2495**
E029	0.2394	0.2321	0.2948	0.2521	**0.3106**
E030	0.1612	**0.2464**	0.2220	0.2398	0.2451
E031	0.4911	0.4595	0.6122	0.4762	**0.6497**
E032	0.0706	0.1278	0.1490	0.1301	**0.1729**
E033	0.2217	0.3170	0.3731	0.3265	**0.3971**
E034	0.1658	0.2129	0.3302	0.2231	**0.6541**
E035	0.2331	0.2650	0.3580	0.2874	**0.3771**
E036	0.1753	0.1897	0.2139	0.1923	**0.2230**
E037	0.2454	0.2928	0.3325	0.3133	**0.3569**
E038	0.0745	0.1127	0.1231	0.1187	**0.1259**
E039	0.2161	0.2531	**0.3990**	0.3294	0.3986
E040	0.5809	**0.3205**	0.3157	0.3095	0.3021
E041	0.0489	0.1589	0.2166	0.1782	**0.2254**
E042	0.1021	0.1358	0.1787	0.1532	**0.1799**
E043	0.0967	0.1568	0.2037	0.1890	**0.2101**
E044	0.0732	**0.2697**	0.2087	0.2543	0.1968
E045	0.1307	0.2315	0.2517	0.2385	**0.2786**
E046	0.1952	0.2457	0.2668	0.2412	**0.2721**
E047	0.0531	0.0837	0.1796	0.1187	**0.1865**
E048	0.0672	0.0642	0.0672	0.0654	**0.0674**
E049	0.0641	0.1250	0.1245	0.1189	**0.1329**
E050	0.2076	0.2321	0.1867	**0.2489**	0.2039
MAP	0.1761	0.2128	0.2527	0.2218	**0.2730**

Table 4.1: Evaluation of event detection approaches on the MED14 dataset.

practical importance of the AP and MAP differences between these two methods that are observed in Table 5.1.

4.3.2 Application of KSVM-iGSU and RD-KSVM-iGSU to Video Aesthetic Quality Assessment

Problem statement, video representation, and uncertainty

Video aesthetic quality assessment is about the automatic assessment of a given video's aesthetic value and the corresponding ranking of the videos within a dataset, such that videos of higher aesthetic value can be ranked higher.

For video aesthetic quality assessment, exploiting both static and motion information is important. Most of the existing works in video aesthetic quality assessment borrow rules from photography and cinematography in order to represent the videos. Typically, each video is divided into its shots or it is sampled such that a set of key-

Figure 4.1: Indicative results (top-5 returned shots) for comparing RD-KSVM-iGSU with RD-KSVM, for four event classes.

frames are extracted. Then a set of photo- and motion-based descriptors are applied to

each keyframe and the resulting keyframe-level representations are averaged in order to obtain a final video-level representation.

For our experiments, similarly to several works in literature, initially each video is divided into its shots using the shot detection method of [3]. Then, for each video, we estimate the mean duration of its shots, and, considering that the shot transitions can be either abrupt or gradual, we estimate for each of these transition types their duration as a percentage of the whole video's duration. This results in a 3-element video-level vector.

Subsequently, one keyframe per second is extracted from the original raw video sequence (irrespective of shot boundaries), and photo- and motion- based features are extracted for each one of them. Photo-based features include the simplicity, colorfulness, sharpness, pattern and overall aesthetic quality values, which are extracted based on the still-image aesthetic quality assessment method proposed in [78]. Motion-based features, adopted from [122], include: a) a measure of similarity between successive frames (cross-correlation between these frames), b) a measure of the diversity of motion directions (motion direction entropy), c) a measure of the stability of the camera during the capturing process (hand-shaking), and d) a measure which can distinguish the difference between three categories of shots: focused shots, panorama shots and static shots (shooting type). The above result in a 44-element keyframe-level feature vector. Concatenating with it the video-level feature vector, we end up with a 47-element vector as the final representation for each keyframe.

In contrast to other state-of-the-art approaches, which merely take the average of the aforementioned keyframe-level video representations in order to obtain video-level ones, we treat the keyframe-level vectors as observations of the input Gaussian distributions that describe the training videos. That is, let $\mathcal{X}$ be a set of l annotated random vectors representing the video-level feature vectors. We assume that each random vector is distributed normally; i.e., for the random vector representing the i-th video, $\mathbf{X}_i$, we have $\mathbf{X}_i \sim \mathcal{N}(\mathbf{x}_i, \Sigma_i)$. Also, for each random vector $\mathbf{X}_i$, a number, N_i, of observations, $\{\mathbf{x}_i^t \in \mathbb{R}^n : t = 1, \ldots, N_i\}$ is available; these are the keyframe-level feature vectors that have been computed. Then, the mean vector and the covariance matrix of $\mathbf{X}_i$ are computed as in the case of video event detection (see Sect. 4.3.1) using Eq. (4.12). Then, again similarly to Sect. 4.3.1, due to the assumption for isotropic covariance matrices, we approximate the above covariance matrices as multiples of the identity matrix, i.e. $\widehat{\Sigma_i} = \sigma_i^2 I_n$, setting σ_i^2 equal to the mean value of the elements of the main diagonal of Σ_i.

Dataset and evaluation measures

Existing datasets for VAQ assessment, e.g., [122, 38], contain only very short video segments (e.g., < 60 seconds) or segments that are extracted from professionally-produced videos (e.g., Hollywood movies), thus not being sufficiently representative of the user-generated content found in social platforms such as YouTube. We aim to perform video aesthetic quality assessment under conditions that are as close to real-life scenarios as possible, and for this reason we introduce a new video dataset that consists

of user-created videos, capturing moments of everyday life, such as excursions, school concerts, and training processes. We downloaded from YouTube 700 videos covering a variety of categories, such as outdoor activities, do it yourself videos, make up tutorials, lectures, and home-made video, licensed under Creative Commons Attribution [1]. The duration of each of these videos ranges from 1 to 6 minutes.

Subsequently, we conducted an annotation process that involved 12 annotators watching and evaluating the aesthetic value of these videos by assigning binary aesthetic quality ratings; 1 being assigned to videos of high aesthetic quality and 0 to videos of low aesthetic quality. Each video was assessed by 5 different annotators. Before the annotation process, the annotators watched some indicative examples of videos of high and low aesthetic quality, and were instructed to remain as uninfluenced as possible by the video's semantics. The final aesthetic score of each annotated video was calculated as the median of the annotators' individual scores, while the average of those annotators' individual scores was also preserved such that the videos can be distinguished into two main categories, those that were annotated as of high or low aesthetic quality by all annotators, and those for which not all annotators were in agreement. As a result of the annotation process, 350 videos are rated as being of high aesthetic quality and another 350 as being of low aesthetic quality. Indicative frames of such videos are shown in Fig. 4.2. We call this dataset CERTH-ITI-VAQ700 and make it publicly available for research purposes[4].

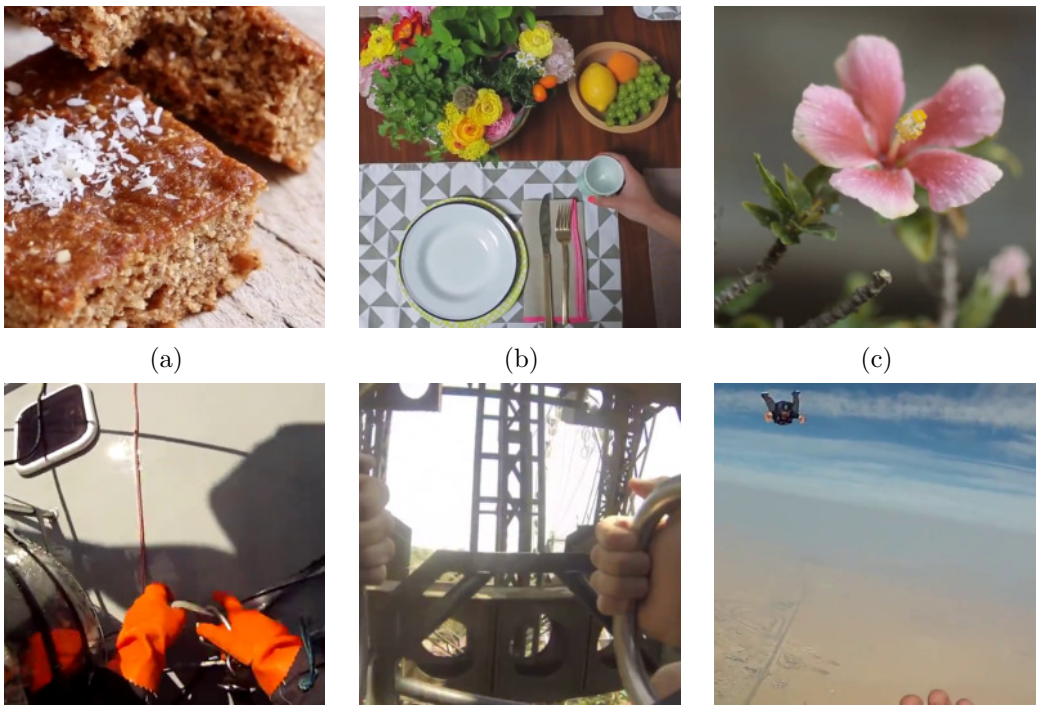
In our experiments, the above dataset is randomly split into a training subset (50%) and an evaluation subset (50%), each maintaining a positive-negative ratio of 1:1. That is, each of the training and evaluation subsets includes 175 positive (high aesthetic) and 175 negative (low aesthetic) video examples. As discussed in section 4.3.2, for video representation, 1 keyframe per second was extracted at regular time intervals from each video, and each keyframe was represented using the proposed photo- and motion-based features.

Since the problem of video aesthetic quality assessment can be naturally seen as a retrieval application, where a user queries for videos of high aesthetic quality within a dataset, for assessing the performance of our method we use (similarly to Sect. 4.3.1 for the event detection problem) the average precision (AP) [88], as well as the precision and the recall, which are measures that are typically used in the VAQ assessment literature [78, 122, 82, 121, 76].

Experimental results and comparisons

The proposed KSVM-iGSU and Relevance Degree KSVM-iGSU (RD-KSVM-iGSU) are tested and compared to the standard linear SVM (LSVM), the standard kernel SVM (KSVM), the linear LSVM-iGSU, and the Relevance Degree Kernel SVM (RD-KSVM). Linear and kernel SVMs are the state-of-the-art classifier for the problem of VAQ assessment [78, 122, 82, 121, 81]. For all kernel-based classifiers, the radial basis function (RBF) kernel was used. Training parameters λ, γ were obtained via

[4]Our video aesthetic quality assessment dataset and the corresponding ground-truth annotation are publicly available at `http://mklab.iti.gr/project/certh-iti-vaq700-dataset`.



(a) (b) (c)

(d) (e) (f)

Figure 4.2: Indicative examples of videos of high (a,b,c) and low (d,e,f) aesthetic value, available in our dataset.

Table 4.2: Performance of the proposed methods compared to the standard LSVM, KSVM, and RD-KSVM in terms of average precision using CERTH-ITI-VAQ700 dataset.

LSVM (e.g. [10])	**LSVM-iGSU (see Chap. 3)**	**KSVM (e.g. [122, 82, 121])**	**KSVM-iGSU (proposed)**	**RD-KSVM [108]**	**RD-KSVM-iGSU (proposed)**
0.6335	0.6889	0.6325	0.7084	0.6415	**0.7117**

a 3-fold cross-validation procedure (grid search) with λ being searched in the range $\{2^{-4}, 2^{-3}, \ldots, 2^{6}, 2^{7}\}$ and γ in the range $\{2^{-7}, 2^{-6}, \ldots, 2^{3}, 2^{4}\}$. Especially for the cases of RD-KSVM and RD-KSVM-iGSU, the videos of the training set that were not annotated with the same label by all the annotators, were assigned a relevance degree c that was optimized by a cross-validation procedure (line search) with c being searched in the range $\{0.05, 0.10, \ldots, 0.95, 1.00\}$. In the latter cross-validation process, the values of λ and γ were set to the optimal values found in the cases of KSVM and KSVM-iGSU, respectively, i.e., where all samples were used with relevance degree equal to the unity. Each of the above experiments was repeated 50 times using different random training/evaluation subsets, similarly to [78].

Table 4.2 shows the average performance of KSVM-iGSU and RD-KSVM-iGSU compared to the standard LSVM, KSVM, LSVM-iGSU, and RD-KSVM in terms of

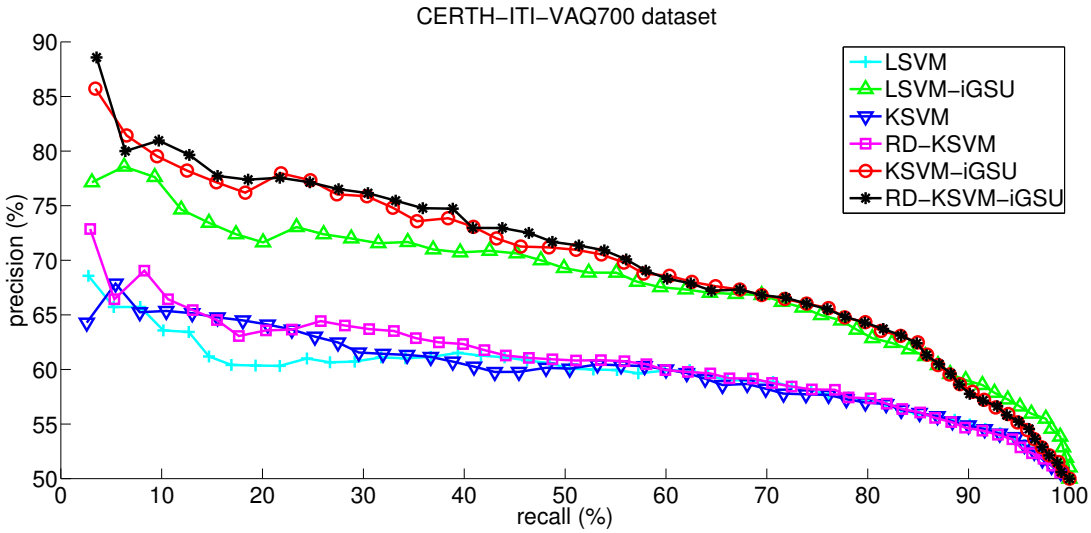


Figure 4.3: Precision-Recall curves for the proposed KSVM-iGSU and RD-KSVM-iGSU, compared to the state-of-the-art LSVM, KSVM, RD-KSVM, and LSVM-iGSU methods, on the CERTH-ITI-VAQ700 dataset.

average precision. We see that the proposed learning methods lead to better VAQ assessment results compared to the state-of-the-art learning approaches. Specifically, introducing uncertainty leads to AP being increased from 0.6325 (KSVM) to 0.7084 (KSVM-iGSU) – a relative boost of 12% – while also exploiting information about the confidence of the training data ground-truth labels leads to AP being increased from 0.6415 (RD-KSVM) to 0.7117 – a relative boost of 11%. The kernelization of our previous learning method LSVM-iGSU, which had not been used for VAQ assessment in [110], also leads to a moderate AP increase from 0.6889 (LSVM-iGSU) to 0.7084 (KSVM-iGSU) or 0.7117 (RD-KSVM-iGSU) – a relative boost of up to 3.3%.

Finally, Fig. 4.3 shows the recall-precision curves of the proposed methods (KSVM-iGSU and RD-KSVM-iGSU) and standard LSVM, KSVM, LSVM-iGSU [110], and RD-KSVM [108]. Again, we observe the superiority of RD-KSVM-iGSU and KSVM-iGSU over their linear counterpart, LSVM-iGSU, and even greater performance differences between them and LSVM, KSVM, and RD-KSVM.

4.4 Conclusions

Two new learning methods were proposed in this paper, building on the linear SVM-GSU (see Chapter 3), which is a linear classifier that takes input uncertainty into consideration. The first proposed method (KSVM-iGSU) results in non-linear decision boundaries, while the second one (RD-KSVM-iGSU) additionally takes into account the confidence in the ground-truth labels of the training data. We applied the proposed methods to two challenging problems of video understanding and indexing: video event detection and video aesthetic quality assessment. For video event detection, we experimented on the challenging TRECVID MED 2014 dataset, using

a limited number of positive and related samples for training. Another challenging video dataset, the CERTH-ITI-VAQ700, was used for experimenting on video aesthetic quality assessment. The experimental results of our methods show considerable performance improvement in comparison to the state-of-the-art learning methods that are used in the literature for the video event detection and aesthetic quality assessment problems.

Chapter 5

Exploiting Uncertainty in Deep Convolutional Neural Networks

Contents

In this chapter, we will investigate the potential use of the ideas presented in the previous chapters under the Deep Convolutional Neural Networks (DCNNs) framework. DCNNs, a class of deep feed-forward artificial neural networks, use a variation of multi-layer perceptrons, and have been designed to require minimal preprocessing, compared to other image classification algorithms. Deep Convolutional Neural Networks (DCNNs) have achieved state-of-the-art performance on various Computer Vision tasks, such as image classification [98, 44, 124]. Typical architectures comprise of a set of layers implementing filtering and pooling operations, and a final classification layer implementing a softmax operation. In the last years, research has focused mainly on the architectural choices, as well, the elementary processing units (e.g., residual networks, non-linear filters, inception architectures [44, 124, 69, 102, 118, 128]).

While there has been significant research in the design of the architecture and of the filters in the convolution layers, there is less work around the pooling layers. Pooling layers, typically implemented as max and mean filters, are aimed at reducing the dimensionality of the input and producing feature maps that are robust to small spatial tranformations. This has proven to be extremely effective and as a result, pooling layers, are included in all state-of-the-art architectures such the WRN [124], Inception [103] and VGG [98]. Typically, they are introduced towards the last stages of the networks, and almost invariably before the final classification layer. Despite their success, their use has been pointedly criticised by Hinton [91], among others, since information about the exact spatial location of the activation maps is lost during pooling.

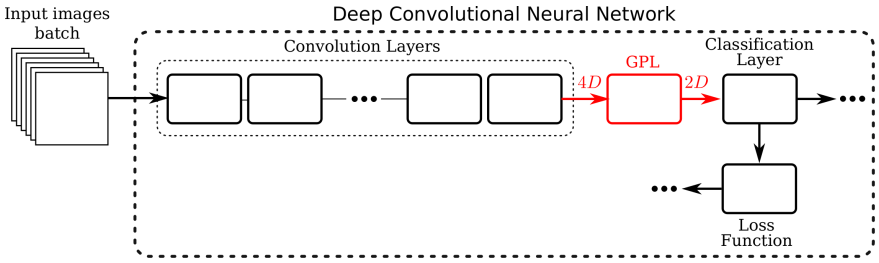


Figure 5.1: A typical deep convolutional neural network architecture for image classification: input batch of images are passed through a set of convolution layers, and before the classification stage (softmax + cross-entropy), the output of the last convolution layer is passed through an averaging pooling layers called Global Pooling Layer (GPL).

Some works model explicitly geometric transformations, (e.g., Jaderberg et al. [49]), however this is typically done at the early layers and comes at a computational cost.

In addition, while the choice of the loss function has been an active problem before the Deep Learning (DL) era, typically classification DCNN use the cross-entropy loss after a softmax layer [16]. Recently, the DL community has investigated on incorporating into the DL framework losses that were typically used in other learning paradigms. Those losses are usually motivated by the specific problem domain in hand, such the face recognition problem [113, 72]. Among them, some adopt a large-margin loss function. For example, Liu et al. [73] proposed a generalized large-margin softmax (L-Softmax) loss which explicitly encourages intra-class compactness and inter-class separability between learned features. However, these works typically use prior knowledge about the problem domain (e.g., face recognition), for example by constraining learned features to be discriminative on a hypersphere manifold where faces are assumed to lie.

In this chapter we try to address the above issues by proposing a novel "uncertainty-aware" maximum-margin loss function that is used to train the last two layers of the network that implement an average pooling and a classification operation respectively. Typical architectures that apply that averaging pooling operation before the classification layer include most of the most competitive ones in recent years, such as the AlexNet [62], VGG [98], Inception [103], ResNet [44], and Wide Residual Network [124]. We propose, during training to not only compute the first-order statistics, that is the mean that is calculated by the pooling layer, but also the second-order statistics, that is the variance – clearly, those statistics are calculated in the area where the mean pooling filter is applied. The means and the variances are used to model each input example as a Gaussian distribution. That is, each input example is modeled as a distinct Gaussian distribution. Then, at the classification layer, for each training example, we compute the probability that it is misclassified and use this probability in order to weight the squared hinge loss accordingly. The proposed method does not add any further computational cost, and also does not affect the convergence, compared to the standard approach.

5.1 A typical CNN architecture for image classification

First, let us briefly review a typical (D)CNN architecture for the problem of image classification. In Fig. 5.1, we illustrate such an architecture, which typically includes a set of convolution layers (either using residual blocks or not), an average pooling layer that is usually referred to as the Global Pooling Layer (GPL), and a classification layer, which is typically implemented as a fully connected (FC) layer followed by the softmax function and the cross-entropy loss function.

Since we are interested solely in the classification stage of a CNN, we will omit any details concerning the exact architecture in hand; the latter could be substituted by virtually any other architecture, so long as that the input to the classification layer is the result of a pooling operation (see Fig. 5.1). This is the case for some of the most successful networks used in recent years, such as the AlexNet [62], VGG [98], Inception [103], ResNet [44], and Wide Residual Network [124].

Let us focus on the last layer of the network where the classification process is realized. The input to this layer is the result of an averaging pooling layer (Global Pooling Layer – GPL) that transforms its input (a 4-dimensional tensor – the output of the last convolution layer) to a 2-dimensional matrix, which along with the ground truth labels, forms the training set of the network's classifier.

In Fig. 5.2 we illustrate the Global Pooling Layer (GPL) that is used in many successful architectures (such as the WRN [124]). This layer receives as input a 4-dimensional tensor of size $n \times B \times \rho \times \rho$, that is, the output of the last convolution layer, and outputs a 2-dimensional matrix of size $n \times B$, where B denotes the batch size, n the number of channels (i.e., the number of filters used in the last convolution layer), and $\rho \times \rho$ the size of the activation maps after the last convolution layer. That is, the output of this layer, along with the corresponding ground truth labels, form the training set of the classifier of the CNN. Each input datum (image), after the spatial pooling layer, is represented by an n-dimensional feature vector, whose elements have been computed by averaging the corresponding $\rho \times \rho$ blocks (see Fig. 5.2). The value of ρ varies across different CNN architectures; for instance, in the case of the WRN [124] it holds that $\rho = 8$, and thus each element of the feature vector of an input datum is the mean of 64 values, while in the case of the AlexNet [62] architecture, $\rho = 2$ and thus the mean of 4 values form each element of each feature vector.

As discussed above, the output of the average pooling layer feeds the classification layer of the network. The latter is typically performed by an FC layer, which in the case of a K-class classification problem, is described by a set of K linear classifiers. That is, the classifier for the j-th class, is usually given as a hyperplane $\mathcal{H}_j \colon \mathbf{w}_j^\top \mathbf{x} + b_j = 0$, where $\mathbf{x} \in \mathbb{R}^n$ denotes the feature representation of an input datum (i.e., an image of the input batch), and $\mathbf{w}_j \in \mathbb{R}^n$, $b_j \in \mathbb{R}$ are the parameters of the hyperplane. The parameters of the FC layer, thus, can be encapsulated in a weights matrix $W = (\mathbf{w}_1 \cdots \mathbf{w}_j \cdots \mathbf{w}_K) \in \mathbb{R}^{n \times K}$ and a bias terms vector $\mathbf{b} = (b_1, \ldots, b_j, \ldots, b_K)^\top \in \mathbb{R}^K$.

The standard approach followed in literature (e.g., in [62, 98, 103, 44, 124]) merely uses the mean vectors for computing a set of scores corresponding to each class and

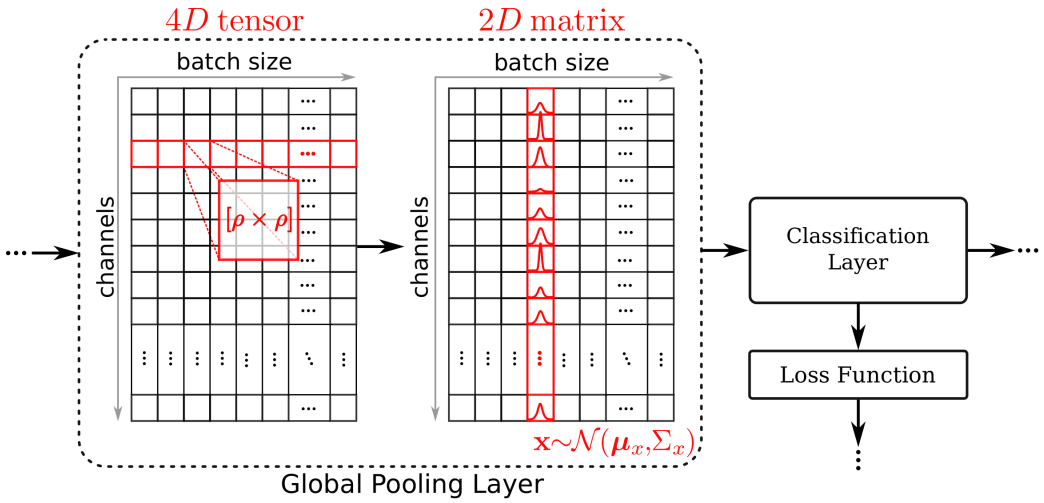


Figure 5.2: A typical average pooling layer (Global Pooling Layer – GPL) preceding the classification layer of a CNN, and its modification so as we compute not only the first-order statistics (means), but also the second-order statistics (variances). These statistics are calculated in the area where the mean pooling filter is applied. The means and the variances are used to model each input example as a Gaussian distribution.

then applies the softmax function [16] for transforming these scores in $[0, 1]$ so as to be interpreted as a-posteriori class membership probabilities. More specifically, the feature representation of an input datum, $\mathbf{x} \in \mathbb{R}^n$, is assigned a score $s_j(\mathbf{x})$ with respect to the j-th class, that is given by

$$s_j(\mathbf{x}) = \mathbf{w}_j^\top \mathbf{x} + b_j, \;\; j = 1, \ldots, K. \tag{5.1}$$

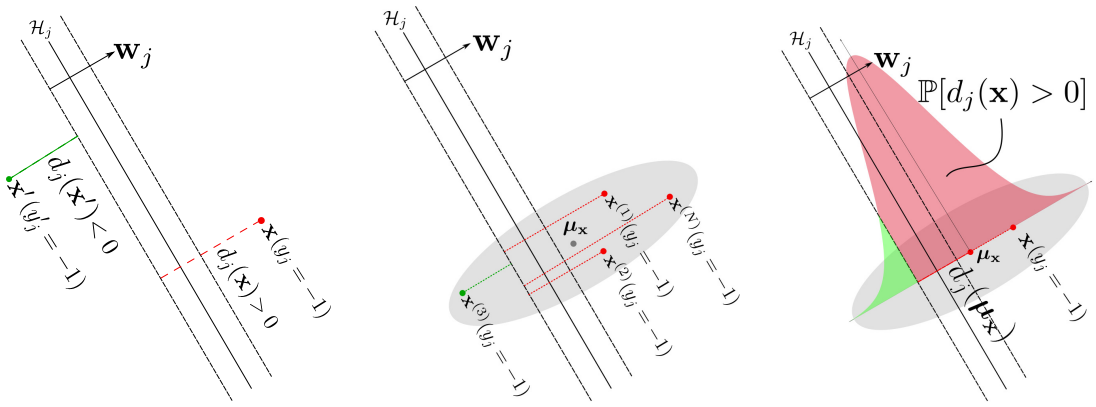
Then, a prediction probability $p_j \in [0, 1]$ is computed with respect to the j-th class using the softmax function as follows

$$p_j(\mathbf{x}) = \frac{\exp(s_j)}{\sum_{k=1}^{K} \exp(s_k)} \;\; j = 1, \ldots, K. \tag{5.2}$$

Finally, the (categorical) cross-entropy function is used for computing the classification loss. That is, an arbitrary input datum $\mathbf{x}$ which has been assigned with a prediction p_j and is labeled with $y_j \in \{\pm 1\}$ with respect to the j-th class, introduces a loss of the form

$$\mathcal{L}_j(\mathbf{x}) = -\sum_{j=1}^{K} y_j \log(p_j) \;\; , j = 1, \ldots, K. \tag{5.3}$$

The total loss introduced by a training example $\mathbf{x}$ is computed by aggregating all the individual losses, i.e., $\mathcal{L}(\mathbf{x}) = \sum_{j=1}^{K} \mathcal{L}_j(\mathbf{x})$.



(a) (b) (c)

Figure 5.3: (a) Illustration of the squared hinge loss: a training datum $\mathbf{x}$ is misclassified when $d_j(\mathbf{x}) = 1 - y_j(\mathbf{w}_j^\top \mathbf{x} + b_j) > 0$ introducing a loss equal to $\Big(1 - y_j(\mathbf{w}_j^\top \mathbf{x} + b_j)\Big)^2$. (b) When $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \Sigma_x)$, there is uncertainty in the position (value) of the training example $\mathbf{x}$. (c) Proposed approach: we weight the square of the expected value of $d_j(\mathbf{x})$ by the probability that the training example $\mathbf{x}$ is misclassified, i.e., $\mathbb{P}\left(d_j(\mathbf{x}) > 0\right)$.

5.2 A maximum-margin classifier for CNNs

Following the discussion and utilizing the formulation that presented above, we will now proceed to the substitution of the standard classification layer of a typical CNN, i.e., an FC layer followed by softmax and cross-entropy, with a maximum-margin classifier that uses the hinge loss function. This new classification layer is again implemented as an FC layer, parametrized by a weights matrix $W \in \mathbb{R}^{n \times K}$ and bias terms vector $\mathbf{b} \in \mathbb{R}^K$, similarly to the standard FC layer discusses above. However, instead of using the softmax function that subsequently would feed the cross-entropy loss function, we directly use the outputs of the FC layer and compute the squared hinge loss. That is, if an input datum $\mathbf{x}$ (i.e., the n-dimensional feature representation that corresponds to an input image), is annotated with a truth label $y_j \in \{\pm 1\}$ with respect to the j-th class, then the loss that it introduces (with respect to the j-th class) is given by

$$\mathcal{L}_j(\mathbf{x}) = \max\Big(0, 1 - y_j(\mathbf{w}_j^\top \mathbf{x} + b_j)\Big)^2. \tag{5.4}$$

This is illustrated in Fig. 5.3a, where we show two input examples, $\mathbf{x}$ and $\mathbf{x}'$ – the former being misclassified and the latter being correctly classified with respect to the j-th class.

The former example, $\mathbf{x}$, introduces a loss equal to $\Big(1 - y_j(\mathbf{w}_j^\top \mathbf{x} + b_j)\Big)^2$, since $1 - y_j(\mathbf{w}_j^\top \mathbf{x} + b_j) > 0$, while the latter, $\mathbf{x}'$, introduces a zero loss, since $1 - y_j(\mathbf{w}_j^\top \mathbf{x}' + b_j) < 0$. Similarly to the hinge loss function, the total loss introduced by a training example $\mathbf{x}$ is computed by aggregating all the individual losses, i.e., $\mathcal{L}(\mathbf{x}) = \sum_{j=1}^{K} \mathcal{L}_j(\mathbf{x})$.

5.3 Exploiting uncertainty in DCNN

Let us revisit the pooling operation (see Figs. 5.1 and 5.2), since this is the stage where the training set, which subsequently feeds the classifier, is formed. As shown in Fig. 5.2 and discussed in Sect. 5.1, each feature that forms a feature vector that represents an input datum, is the result of an averaging operation over ρ^2 values (in the case of a WRN [124], $\rho = 8$ and thus each feature is the mean value of 64 values). That is, if $\mathbf{x}$ is the n-dimensional feature vector of an arbitrary input datum, then it is of the following form:

$$\mathbf{x} = (x_1, \ldots, x_j, \ldots, x_n)^\top,$$

where $x_j = \frac{1}{\rho^2} \sum_{t=1}^{\rho^2} x_j^t,\ \ j = 1, \ldots, n,\ x_j^t$ is the t-th element of the j-th $\rho \times \rho$ block. Clearly, an amount of information is lost during such a process. This is illustrated in Fig. 5.3b. In order to exploit this additional source of information, besides the above mean values, we also compute the corresponding variances. As a result, each n-dimensional feature vector that corresponds to an input datum is described by a pair of a mean vector $\boldsymbol{\mu}_x$ and a (diagonal) covariance matrix Σ_x.

We consider the input datum $\mathbf{x}$ as a random vector that follows a multi-variate Gaussian distribution with given mean vector and covariance matrix, i.e., $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \Sigma_x)$. A sample $\mathbf{x}$ drawn from that distribution that is annotated with a ground truth label $y_j \in \{\pm 1\}$ with respect to the j-th class introduces a loss that can be expressed as

$$\mathcal{L}_j(\mathbf{x}) = \max\left(0, 1 - y_j(\mathbf{w}_j^\top \mathbf{x} + b_j)\right)^2,\ \ j = 1, \ldots, K \tag{5.5}$$

and is misclassified when $1 - y_j(\mathbf{w}_j^\top \mathbf{x} + b_j) > 0$. Let us for notation brevity define:

$$d_j(\mathbf{x}) = 1 - y_j(\mathbf{w}_j^\top \mathbf{x} + b_j),\ \ j = 1, \ldots, K. \tag{5.6}$$

Since $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \Sigma_x)$, then d_j, which is a linear transformation of $\mathbf{x}$, is also random and, more specifically, it is a uni-variate Gaussian variable, with mean value and variance that are given with respect to $\mathbf{x}$'s respective moments. More specifically, the mean value of $d_j(\mathbf{x})$, μ_{d_j}, and its variance, $\sigma_{d_j}^2$, are given respectively as

$$\mu_{d_j} = \mathbb{E}[d_j(\mathbf{x})] = 1 - y_j(\mathbf{w}_j^\top \boldsymbol{\mu}_x + b_j), \tag{5.7}$$

and

$$\sigma_{d_j}^2 = \mathbb{E}\left[\left(d_j(\mathbf{x}) - \mu_{d_j}\right)^2\right] = \mathbf{w}_j^\top \Sigma_x \mathbf{w}_j. \tag{5.8}$$

Then, the probability that an input example $\mathbf{x}$ with label y_j is misclassified, that is the probability that $d_j(\mathbf{x})$ is positive, can be calculated as follows

$$\mathbb{P}\left(d_j(\mathbf{x}) > 0\right) = 1 - \mathbb{P}\left(\frac{d_j(\mathbf{x}) - \mu_{d_j}}{\sigma_{d_j}} < -\frac{\mu_{d_j}}{\sigma_{d_j}}\right) = 1 - \mathbb{P}\left(z < -\frac{\mu_{d_j}}{\sigma_{d_j}}\right), \tag{5.9}$$

where $z \sim \mathcal{N}(0, 1)$ is the standard uni-variate Gaussian variable. If Φ denotes the CDF of z, it follows that

$$\mathbb{P}\left(d_j(\mathbf{x}) > 0\right) = 1 - \Phi\left(-\frac{\mu_{d_j}}{\sigma_{d_j}}\right) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{\mu_{d_j}}{\sqrt{2\sigma_{d_j}^2}}\right)\right]. \tag{5.10}$$

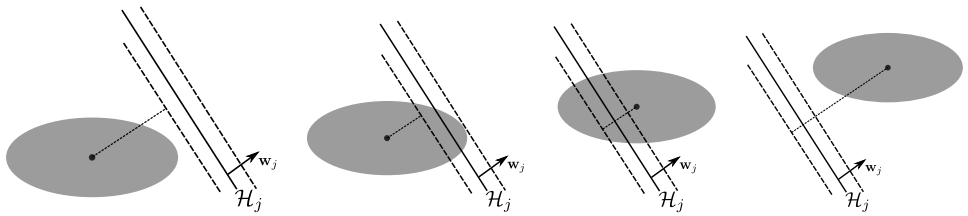


Figure 5.4: Illustration of the proposed loss function. In cases (a) and (d), the proposed loss is equal to the standard squared hinge loss. In (b) the proposed loss has a positive value, in contrast to the standard squared hinge loss that is zero, while in (c) the proposed loss is less than the standard squared hinge loss, since there is a probability that the example lies on the opposite halfspace than the one its mean lies.

The above probability can serve as a degree of confidence that the input training example at hand is misclassified (see Fig. 5.3c). Thus, instead of using the standard squared hinge loss, we propose using the following loss function

$$\mathcal{L}_j(\mathbf{x}) = \mathbb{P}\left(d_j(\mathbf{x}) > 0\right) \mu_{d_j}^2 = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{\mu_{d_j}}{\sqrt{2\sigma_{d_j}^2)}}\right)\right] \mu_{d_j}^2. \tag{5.11}$$

It is worth noting that, the above loss degenerates to the standard squared hinge loss when the input uncertainty is zero. Furthermore, in Fig. 5.4 we illustrate how the proposed loss behaves under Gaussian input uncertainty and its relation to the standard squared hinge loss.

5.4 Experimental results

We evaluate the proposed method on the well-known CIFAR-10 and CIFAR-100 datasets [61] for the problem of image classification using an experimental setup that is similar to [124]. CIFAR-10 and CIFAR-100 datasets consist of 32×32 color images drawn from 10 and 100 classes that are split into 50000 train and 10000 test images. For data augmentation we do horizontal flips and take random crops from images padded by 4 pixels on either side, filling missing pixels with reflections of the original image. Our experiments are based on the ResNet architecture proposed in [124] that uses pre-activation residual blocks – we use this as a baseline. We preprocessed the input images by simple mean/std normalization so that we can directly compare with [124].

We compared the proposed method in terms of test error both to the baseline WRN [124] that uses the cross-entropy loss, and to a variant of WRN that uses the squared hinge loss (WRN+SVM). We experimented using three different variants of the WRN architecture in terms of depth and width, similarly to [124]. That is, for each of the above loss functions, we trained a 52-1, a 16-4, and a 28-10 network[1]. We show

[1]Following the nomenclature of [124], in a n-k WRN, n defines the depth parameter, and k defines

Table 5.1: Comparison between our proposed loss function, the squared hinge loss (WRN+SVM), and the standard cross-entropy (WRN) in terms of test error, using the Wide Residual Network architecture [124] for various depth and width parameters.

CIFAR-10				
depth	width	WRN	WRN+SVM	Ours
52	1	6.28	6.15	**6.07**
16	4	4.81	4.60	**4.23**
28	10	**3.89**	4.05	3.92
CIFAR-100				
depth	width	WRN	WRN+SVM	Ours
52	1	29.65	29.45	**29.32**
16	4	23.39	22.80	**22.43**
28	10	18.87	18.73	**18.62**

the results in Table 5.1. Each reported result is the median of five runs, as in [124]. In CIFAR-10, our method outperforms both the baseline WRN (cross-entropy loss) and the WRN+SVM (squared hinge loss) significantly[2] in the case of the 16-4 network architecture, while in the cases of 52-1 and 28-10 architectures the differences are not statistically significant. For CIFAR-100, our method achieved better results for all network architectures – in the case of 16-4 the differences are statistically significant.

The 52-1 network has approximately 760K parameters and is the "thinnest" network architecture we experimented with. As a consequence, the dimensionality of the input space of the classification layers is relatively low (64). This may explain why the SVM-based classification layers do not exhibit large differences in comparison to the softmax classification layers – it is generally acknowledged that linear SVM works better at high dimensional input spaces. At the other end of the spectrum, that is the 28-10 network, despite the fact that the dimensionality of the input space of the classification layer is higher (640), it also contains more than 36.5M parameters, and thus modifying solely the last layer (about 6500 parameters) does not seem to have a considerable impact on the overall network's classification performance. However, in the case of the 16-4 network that contains about 2.7M parameters and the dimensionality of the input space of the classification layer is 256, that is, it is a moderately deep and wide one, it seems that exploiting uncertainty can indeed improve the classification performance significantly. Such architectures are typically used by the research community [19, 47], since deeper and wider architectures (e.g., the WRN 28-10) are not feasible in large-scale datasets, such as the ImageNet.

Finally, in Table 5.2 we provide experimental results for the WRN 16-4 network, similarly to [117]. We show the results of our implementation (for the baseline WRN using the cross-entropy loss, and the WRN with the squared hinge loss and the WRN

the width parameter.

[2]We used the two sample t-test [22].

Table 5.2: Comparison between our proposed loss function (Ours), the squared hinge loss (WRN+SVM), the standard cross-entropy (WRN), and GNPP [117] in terms of test error, using the Wide Residual Network 16-4 architecture [124].

Our implementation			As implemented in [117]	
WRN	WRN+SVM	Ours	WRN without GNPP	WRN with GNPP
CIFAR-10				
4.81	4.60	**4.23**	5.54	5.31
CIFAR-100				
23.39	22.80	**22.43**	25.52	25.01

with the proposed one), as well as of the implementation used in [117] for the same network with or without the GNPP layer.

5.5 Conclusion

In this chapter we investigated the introduction and exploitation of uncertainty into the Deep Convolutional Neural Network learning paradigm. More specifically, we proposed a method for improving a CNN's classification performance by a) substituting the classification layer of the network (i.e., softmax followed by the cross-entropy loss function) with a maximum-margin classifier that uses the squared hinge loss, and b) by additionally exploiting the uncertainty information derived from the pooling layer of the network using an appropriate maximum-margin loss function. The proposed method can be applied to any architecture that includes a pooling operation before the classification stage, such as [62, 98, 103, 44, 124]. In this chapter, we modified the WRN architecture by substituting the loss function at the classification layer (last layer) of the network, and derived two variants using a) the standard squared hinge loss, and b) a variant of the squared hinge loss that takes the uncertainty introduced by the pooling layer into consideration. The latter, i.e. the uncertainty, is calculated as the (diagonal) variance within the mean filter's extend at the pooling layer, that is, at training time, we keep track not only of the first-order statistics as calculated by the pooling layer (i.e., the mean of the values within the mean-filter's receptive field), but also the second-order statistics (i.e., the variances of the values within the mean-filter's receptive field). We modified the loss function so as the variance of each input datum is taken into account during training and experimented on two popular datasets for image classification, i.e., CIFAR-10 and CIFAR-100, and three standard WRN architectures. Our experimental results show that exploiting uncertainty can improve testing accuracy of moderately deep and wide networks.

Chapter 6

Conclusions

In this thesis we studied the problem of supervised learning under input uncertainty using the maximum-margin SVM paradigm. We started by introducing the linear variant of our classifier and continued with its kernelization using the RBF kernel function. Finally, we investigated the introduction and exploitation of input uncertainty in the DCNN learning framework.

We started by defining the problem of supervised learning under input uncertainty, which we studied in this thesis, by defining what an uncertain training datum is. That is, in this thesis an uncertain training datum is an annotated multi-variate Gaussian distribution with given moments; i.e., with given mean vector and covariance matrix. Each training datum, or training example, can be a distinct Gaussian distribution; that is; uncertainty characterizes each training example separately, not a whole class, for instance. For modeling input uncertainty we chose a well-studied and ubiquitously-used distribution, i.e., the multi-variate Gaussian distribution, for a plethora of reasons such as the facts that a Gaussian distribution a) is expected to serve as a good model, since it is the limit of the sum of a large number of unknown (but reasonably bounded) uncertainty sources (Central Limit Theorem), b) IT is completely described by its first- and second-order statistics, which is absent in other random distributions, c) it provides greater convenience (in terms of mathematical manipulation) compared to other, even conceptually simpler, distributions (e.g., multi-variate uniform distribution), and finally d) compared to other distributions that are inherently isotropic over the input dimensions, or rigid over input dimensions, a Gaussian distribution provides flexibility in modeling anisotropic uncertainty for arbitrary number of input dimensions. That is, one can model the anisotropic uncertainty solely on a set of input dimensions of interest. That is, one can introduce constraints on the covariance matrices, such as them being diagonal, block diagonal, or multiples of the identity matrix. In this way one can model different types of uncertainty.

Then, we used a popular maximum-margin classifier, i.e., the standard linear SVM along with the hinge loss function, as our baseline learning algorithm, and modified it so as the loss introduced by a training example takes its covariance information into consideration. More specifically, we optimized for the soft margin using the *expected*

value of the hinge loss. This essentially means that the loss that is potentially introduced by a training example is measured not solely using a single feature vector (e.g., the mean feature vector of the input Gaussian), but rather using the knowledge of the discrepancy of the respective distribution, i.e., its (co)variance. For this purpose, we provided an analytical evaluation (in Appendix A) of the proposed loss in closed form and we proved (in Appendix B) that it is convex with respect to the optimization parameters. As result, we were allowed to obtain the global optimal solution using an appropriate iterative gradient descent algorithm that is linear with respect to the number training data. For this purpose, we modified and used a popular SGD algorithm, namely the Pegasos algorithm. We also proposed a linear subspace learning approach in order to address the situation where most of the mass of the training Gaussians lie in a low dimensional manifold that can be different for each Gaussian and subsequently solve the problem in lower-dimensional spaces.

It is worth noting that, using the proposed classifier, one would arrive at the same decision border with the classical SVM trained on a dataset containing samples drawn from the Gaussians in question, as the number of samples tend to infinity. However, we showed that as the dimensionality of the input space increases, one needs to generate more samples from the Gaussians in order to preserve a desired approximation of the loss and, thus, of the optimal decision function. We also showed that for spaces of high dimensionality the number of samples needed can be prohibitively high. In addition, we noted that our method degenerates to a classical SVM in the case that all of the Gaussians are isotropic with a variance that tends to zero.

Next, we proceeded to the experimental evaluation of the proposed linear classifier on five publicly available datasets; namely, the MNIST dataset of handwritten digits, the WDBC, the DEAP for emotion analysis, the TV News Commercial Detection dataset and TRECVID MED for the problem of video event detection. For each of the above datasets and problems, either uncertainty information (e.g., variance for each example and for all or some of the input space dimensions) was part of the original dataset, or a method for modeling and estimating the uncertainty of each training example was proposed. As shown in the experiments, SVM-GSU efficiently takes input uncertainty into consideration and achieves better detection or classification performance than standard SVM, previous SVM extensions that model uncertainty isotropically, and other state of the art methods. For the important task of the modeling or the estimation of input uncertainty, we showed that it is a domain- and/or dataset-specific problem, and in this respect, similarly to all of the other methods in the literature that model/use uncertainties, we did not offer a definitive answer on how this can or should be done on any existing dataset. We addressed this important issue by providing a number of methodologies that mainly depend on the specific problems, but can also be applied in similar problems/datasets.

More specifically, in contrast to previous works that model uncertainty in the SVM framework either by considering isotropic noise or by using expensive sampling schemes to approximate their loss functions, our formulation allows for full covariance matrices that can be different for each example. This allows dealing, among others, with cases where the uncertainty of only a few examples, and/or the uncertainty along only a few

of their dimensions, is known or modeled. In the experimental results section we show several real-world problems in which such modeling is beneficial. More specifically, we show cases, in which the variances along (some) of the dimensions are part of the dataset – this includes medical data where both the means and the variances of several measurements are reported, and large scale video datasets, where the means and the variances of some of the features that are extracted at several time instances in the video in question are reported. We then show a case in which means and variances are a by-product of the feature extraction method, namely the Welch method for extracting periodograms from temporal EEG data. And finally, we show a case in which, for an image dataset (MNIST) we model the distribution of images under small geometric transforms as Gaussians, using a first-order Taylor approximation to arrive in an analytic form. In particular, the Taylor expansion method (Appendix B) that is behind the modeling used in Sect. 3.5.2, has been used to model the propagation of uncertainties due to a feature extraction process in other domains; for instance, in [26] (Sect. II.B) this is used to model as Gaussian the uncertainty in the estimation of illumination invariant image derivatives.

In the second main chapter of the thesis, we moved on to the extension of the above linear classifier so as to result in non-linear boundaries. For this, we recast the optimization problem of the linear SVM-GSU into a variational calculus problem; that is, the original optimization problem was rewritten as a problem of minimizing an equivalent (objective) functional, and, thus, instead of looking for a separating hyperplane (i.e., its parameters) in the original input feature space, we looked for a minimizer function that lives in a richer, higher-dimensional (in our case infinite-dimensional) space. We proved that the above functional is such that its minimizer can be represented as a finite linear combination of kernel products (in our case using the RBF kernel function). Additionally, due to the convexity of our objective functional, we could efficiently solve the problem using an appropriate SGD algorithm (i.e., the Pegasos algorithm, similarly to the linear case).

Next, we combined our kernel classifier (i.e., KSVM-iGSU) with the previously proposed Relevance Degree SVM (RD-SVM). In RD-SVM each training example is associated with a confidence value (called relevance degree) indicating the degree of relevance of the respective training example with the class that it is related. This is essentially a method that handles uncertainty in the truth labels, and combined with the proposed KSVM-iGSU provide a methodology for handling uncertainty both in label and feature representation.

We applied the above kernel classifiers in two challenging multimedia understanding problems, namely the video event detection and aesthetic quality video assessment. Especially in the visual understanding domain, the majority of the learning methods employed in video understanding and indexing applications do not address the uncertainty in the training data explicitly. That is, firstly, each training example is assumed to be described by a fixed position in some vector space (feature representation). However, such an approach does not account for the fact that the underlying process of extracting the feature representation may be imperfect or noisy, hence introducing some degree of uncertainty to the generated features. Secondly, each training example

is typically annotated with a binary ground-truth label. This is essentially the result of a quantization process, where different pieces of data that may be perfect or not-so-perfect examples of a class have to be assigned a binary label, and this inevitably introduces some form of quantization error. For instance, in some cases the annotation process could naturally lead to three types of labels, i.e., positive, negative, and "near-miss" or "related", the latter expressing the fact that the example is closely related with the positive class but does not meet the exact requirements for being characterized as a positive instance; yet, for training a binary classifier, these annotations need to be subsequently quantized to just two classes: positive and negative. Similarly, in many cases the ground-truth annotation of training data is carried out by a number of experts who decide on the label of each given example, and a final binary assignment of each sample to the positive or negative class is made by averaging and binarizing, or aggregating in another similar way, the responses of the different annotators. In both the above examples, the ground truth annotation process endows every binary annotation with some level of confidence on it, but this is typically ignored in the subsequent training of a binary classifier. In thesis, we address the above using the proposed KSVM-iGSU and RD-KSVM-iGSU.

Finally, in the last main chapter of the thesis, we investigated the introduction and exploitation of uncertainty into the Deep Convolutional Neural Network learning paradigm. For this purpose we chose a state-of-the-art architecture, i.e., the Wide Residual Network (WRN) that extends the previously proposed Residual Network (ResNet) in an attempt to address the problem of high number of ResNet's layers (depth) that results in high training times. We modified the WRN architecture by substituting the loss function at the classification layer (last layer) of the network, where we used a) the standard squared hinge loss, in an attempt to resemble the standard linear SVM classifier at the top of the network, and b) a variant of the squared hinge loss that takes input uncertainty into consideration. For extracting the information about the uncertainty at the input space of the classification layer, we modified the averaging pooling operation that precedes the classifier and, besides a mean vector per each input datum, we additionally computed a (diagonal) covariance matrix. Thus, each input datum can be represented as a multi-variate distribution with given mean and covariance. Under the assumption of Gaussian uncertainty, similarly to the previous chapters, we modified the loss function so as the variance of each input datum is taken into account during training. We experimented on two popular datasets for image classification, i.e., CIFAR-10 and CIFAR-100, and three standard WRN architectures, and we showed that exploiting uncertainty can improve testing accuracy under moderate network depth and width conditions.

Appendix **A**

On Gaussian-like integrals over halfspaces

Theorem 1. *Let $\mathbf{X} \in \mathbb{R}^n$ be a random vector that follows the multivariate Gaussian distribution with mean vector $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\Sigma \in \mathbb{S}^n_{++}$, where $\mathbb{S}^n_{++}$ denotes the space of $n \times n$ symmetric positive definite matrices with real entries. The probability density function of this distribution is given by $f_{\mathbf{X}} \colon \mathbb{R}^n \to \mathbb{R}$,*

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

Moreover, let $\mathcal{H}$ be the hyperplane given by $\mathbf{a}^\top \mathbf{x} + b = 0$. $\mathcal{H}$ divides the Euclidean n-dimensional space into two halfspaces, i.e., $\Omega_\pm = \{\mathbf{x} \in \mathbb{R}^n \colon \mathbf{a}^\top \mathbf{x} + b \gtrless 0\}$, so that $\Omega_+ \cup \Omega_- = \mathbb{R}^n$ and $\Omega_+ \cap \Omega_- = \varnothing$. Then, the integrals $I_\pm \colon \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$, defined as

$$I_\pm(\mathbf{a}, b) \triangleq \int_{\Omega_\pm} (\mathbf{a}^\top \mathbf{x} + b) f_{\mathbf{X}}(\mathbf{x}) \, \mathrm{d}\mathbf{x},$$

are given by

$$I_\pm(\mathbf{a}, b) = \frac{d_\mu}{2}\left[1 \pm \operatorname{erf}\left(\frac{d_\mu}{d_\Sigma}\right)\right] \pm \frac{d_\Sigma}{2\sqrt{\pi}} \exp\left(-\frac{d_\mu^2}{d_\Sigma^2}\right), \tag{A.1}$$

where $d_\mu = \mathbf{a}^\top \boldsymbol{\mu} + b$ and $d_\Sigma = \sqrt{2\mathbf{a}^\top \Sigma \mathbf{a}}$.

Proof. We begin with the integral I_+. In our approach we will need several coordinate transforms. First, we start with a translation in order to get rid of the mean, $\mathbf{x} = \mathbf{y} + \boldsymbol{\mu}$. Then

$$I_+(\mathbf{a}, b) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \int_{\Omega_1^+} (\mathbf{a}^\top \mathbf{y} + \mathbf{a}^\top \boldsymbol{\mu} + b) \exp\left(-\frac{1}{2}\mathbf{y}^\top \Sigma^{-1} \mathbf{y})\right) \mathrm{d}\mathbf{y},$$

where $\Omega_1^+ = \{\mathbf{y} \in \mathbb{R}^n \colon \mathbf{a}^\top \mathbf{y} + \mathbf{a}^\top \boldsymbol{\mu} + b \geq 0\}$. Next, since $\Sigma \in \mathbb{S}^n_{++}$, there exist an orthonormal matrix U and a diagonal matrix D with positive elements, i.e. the eigenvalues of Σ, such that $\Sigma = U^\top D U$. Thus, it holds that $\Sigma^{-1} = (U^\top D U)^{-1} =$

$U^{-1}D^{-1}(U^\top)^{-1} = U^\top D^{-1}U$. Then, by letting $\mathbf{z} = U\mathbf{y}$ and $\mathbf{a}_1 = U\mathbf{a}$, we have $\mathbf{a}^\top\mathbf{y} = \mathbf{a}^\top(U^{-1}U)\mathbf{y} = \mathbf{a}^\top U^\top U\mathbf{z} = \mathbf{a}_1^\top\mathbf{z}$, and $\mathbf{y}^\top\Sigma^{-1}\mathbf{y} = \mathbf{y}^\top(U^\top DU)^{-1}\mathbf{y} = (\mathbf{y}^\top U^\top)D^{-1}(U\mathbf{y}) = (U\mathbf{y})^\top D^{-1}(U\mathbf{y}) = \mathbf{z}^\top D^{-1}\mathbf{z}$. Then

$$I_+(\mathbf{a}, b) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}\int_{\Omega_2^+}(\mathbf{a}_1^\top\mathbf{z} + \mathbf{a}^\top\boldsymbol{\mu} + b)\exp\left(-\frac{1}{2}\mathbf{z}^\top D^{-1}\mathbf{z}\right)\,\mathrm{d}\mathbf{z},$$

where $\Omega_2^+ = \{\mathbf{z} \in \mathbb{R}^n \colon \mathbf{a}_1^\top\mathbf{z} + \mathbf{a}^\top\boldsymbol{\mu} + b \geq 0\}$, since for the Jacobian $J = |U|$, it holds that $|J| = 1$. Now, in order to do rescaling, we set $\mathbf{z} = D^{\frac{1}{2}}\mathbf{v}$ and $\mathbf{a}_2 = D^{\frac{1}{2}}\mathbf{a}_1$. Thus

$$\mathbf{z}^\top D^{-1}\mathbf{z} = (D^{\frac{1}{2}}\mathbf{v})^\top D^{-1}(D^{\frac{1}{2}}\mathbf{v}) = \mathbf{v}^\top(D^{\frac{1}{2}}D^{-1}D^{\frac{1}{2}})\mathbf{v} = \mathbf{v}^\top\mathbf{v}.$$

Moreover, $\mathbf{a}_1^\top\mathbf{z} = \mathbf{a}_1^\top(D^{\frac{1}{2}}\mathbf{v}) = (D^{\frac{1}{2}}\mathbf{a}_1)^\top\mathbf{v} = \mathbf{a}_2^\top\mathbf{v}$. Also, it holds that $|D|^{\frac{1}{2}} = |\Sigma|^{\frac{1}{2}}$ and $\mathrm{d}\mathbf{z} = |D^{\frac{1}{2}}|\mathrm{d}\mathbf{v} = |\Sigma|^{\frac{1}{2}}\mathrm{d}\mathbf{v}$. Consequently,

$$I_+(\mathbf{a}, b) = \frac{1}{(2\pi)^{\frac{n}{2}}}\int_{\Omega_3^+}(\mathbf{a}_2^\top\mathbf{v} + \mathbf{a}^\top\boldsymbol{\mu} + b)\exp\left(-\frac{1}{2}\mathbf{v}^\top\mathbf{v}\right)\,\mathrm{d}\mathbf{v},$$

where $\Omega_3^+ = \{\mathbf{v} \in \mathbb{R}^n \colon \mathbf{a}_2^\top\mathbf{v} + \mathbf{a}^\top\boldsymbol{\mu} + b \geq 0\}$. Let B be an orthogonal matrix such that $B\mathbf{a}_2 = \|\mathbf{a}_2\|\mathbf{e}_n$, which also means that $\mathbf{a}_2 = B^\top\|\mathbf{a}_2\|\mathbf{e}_n$. Moreover, let $\mathbf{m} = B\mathbf{v}$. Then, $\mathbf{a}_2^\top\mathbf{v} = (B^\top\|\mathbf{a}_2\|\mathbf{e}_n)^\top\mathbf{v} = \|\mathbf{a}_2\|\mathbf{e}_n^\top(B\mathbf{v}) = \|\mathbf{a}_2\|\mathbf{e}_n^\top\mathbf{m}$. Moreover, $\mathbf{v}^\top\mathbf{v} = \mathbf{v}^\top(B^{-1}B)\mathbf{v} = \mathbf{m}^\top\mathbf{m}$. Then

$$I_+(\mathbf{a}, b) = \frac{1}{\sqrt{2\pi}}\int_c^{+\infty}\big(\|\mathbf{a}_2\|t + \mathbf{a}^\top\boldsymbol{\mu} + b\big)\exp\left(-\frac{1}{2}t^2\right)\,\mathrm{d}t,$$

where $c = -\frac{\mathbf{a}^\top\boldsymbol{\mu}+b}{\|\mathbf{a}_2\|}$. Since $\|\mathbf{a}_2\|^2 = \mathbf{a}^\top\Sigma\mathbf{a}$,

$$I_+(\mathbf{a}, b) = \frac{1}{\sqrt{2\pi}}\int_c^{+\infty}\big(\sqrt{\mathbf{a}^\top\Sigma\mathbf{a}}t + \mathbf{a}^\top\boldsymbol{\mu} + b\big)\exp\left(-\frac{1}{2}t^2\right)\,\mathrm{d}t,$$

which is easily evaluated as (A.1). Following similar arguments as above, we arrive at I_-. □

Appendix **B**

On the convexity of the SVM-GSU loss function

Let $\mathcal{J}$ be the objective function of (3.3). We will show that $\mathcal{J}$ is convex with respect to the optimization variables, $\mathbf{w}$ and b, over $\mathbb{R}^n \times \mathbb{R}$. First, as every norm is convex, and every non-negative weighted sum preserves the convexity, it suffices to show that $\mathcal{L}$, as shown in (3.4), is convex with respect to $\mathbf{w}$, b for all $i = 1, \ldots, l$. We will prove an associated theorem first, which we will use to prove the convexity of $\mathcal{L}$, $\forall i$.

Theorem 2. *Let $f\colon \mathbb{R}^n \to \mathbb{R}_+$ be a non-negative, real-valued function. Then, $\phi\colon \mathbb{R}^d \to \mathbb{R}$, given by*

$$\phi(\boldsymbol{\theta}) = \int_{\mathbb{R}^n} \max\Big(0, h(\boldsymbol{\theta}, \mathbf{x})\Big) f(\mathbf{x})\, \mathrm{d}\mathbf{x}, \tag{B.1}$$

is convex with respect to $\boldsymbol{\theta}$ over $\mathbb{R}^d$, if the function h is convex with respect to $\boldsymbol{\theta}$ over $\mathbb{R}^d$.

Proof. Let $\lambda \in [0, 1]$ and $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$. Then,

$$\begin{aligned}
&\phi(\lambda\boldsymbol{\theta}_1 + (1-\lambda)\boldsymbol{\theta}_2) = \\
&\int_{\mathbb{R}^n} \max\Big(0, h(\lambda\boldsymbol{\theta}_1 + (1-\lambda)\boldsymbol{\theta}_2, \mathbf{x}) f(\mathbf{x})\, \mathrm{d}\mathbf{x} \\
&\qquad \leq \int_{\mathbb{R}^n} \max\Big(0, \lambda h(\boldsymbol{\theta}_1, \mathbf{x})\Big) f(\mathbf{x})\, \mathrm{d}\mathbf{x} \\
&\quad + \int_{\mathbb{R}^n} \max\Big(0, (1-\lambda) h(\boldsymbol{\theta}_2, \mathbf{x})\Big) f(\mathbf{x})\, \mathrm{d}\mathbf{x},
\end{aligned}$$

since h is convex and for $p, q, r \in \mathbb{R}$ it holds that $p \leq q + r \Rightarrow \max(0, p) \leq \max(0, q) +$

$\max(0, r)$. Moreover, $\max(0, \lambda p) = \lambda \max(0, p)$, for $\lambda \geq 0$, $p \in \mathbb{R}$, and thus,

$$\begin{aligned} &\phi(\lambda\boldsymbol{\theta}_1 + (1-\lambda)\boldsymbol{\theta}_2) \leq \\ &\lambda \int_{\mathbb{R}^n} \max\Big(0, h(\boldsymbol{\theta}_1, \mathbf{x})\Big) f(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ +(1-\lambda) &\int_{\mathbb{R}^n} \max\Big(0, h(\boldsymbol{\theta}_2, \mathbf{x})\Big) f(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= \lambda\phi(\boldsymbol{\theta}_1) + (1-\lambda)\phi(\boldsymbol{\theta}_2). \end{aligned}$$

Consequently, ϕ is convex with respect to $\boldsymbol{\theta}$ over $\mathbb{R}^d$. □

Using the results of the above theorem, by setting $f(\mathbf{x}) = f_{\mathbf{X}_i}(\mathbf{x})$, which is a real-valued, non-negative function (as a probability density function), and $h(\boldsymbol{\theta}, \mathbf{x}) = 1 - y_i(\mathbf{w}^\top \mathbf{x} + b)$, which is convex with respect to $\boldsymbol{\theta} = (\mathbf{w}^\top, b)^\top$ over $\mathbb{R}^d \equiv \mathbb{R}^n \times \mathbb{R}$, $\mathcal{L}$ is proven to be convex for all i. Consequently, the objective function $\mathcal{J}$ is convex. That means that every local minimum of $\mathcal{J}$ is also a global one.

Appendix C

Modeling the uncertainty of an image

Theorem 3 (Multivariate Taylor's Theorem)**.** *Let* $\mathbf{t} = (t_1, \ldots, t_n)^\top \in \mathbb{R}^n$ *and consider a function* $f\colon \mathbb{R}^n \to \mathbb{R}$*. Let* $\mathbf{a} = (a_1, \ldots, a_n)^\top \in \mathbb{R}^n$ *and suppose that* f *is differentiable (all first partial derivatives with respect to* $t_1, \ldots, t_n$ *exist) in an open ball* $\mathcal{B}$ *around* $\mathbf{a}$*. Then, the first-order case of Taylor's theorem states that:*

If f *is differentiable on an open ball* $\mathcal{B}$ *around* $\mathbf{a}$ *and* $\mathbf{t} \in \mathcal{B}$*, then*

$$f(\mathbf{t}) = f(\mathbf{a}) + \sum_{k=1}^{n} \frac{\partial f}{\partial t_k}(\mathbf{b})(t_k - a_k) = f(\mathbf{a}) + \nabla^\top f(\mathbf{b})(\mathbf{t} - \mathbf{a}), \qquad \text{(C.1)}$$

for some $\mathbf{b}$ *on the line segment joining* $\mathbf{a}$ *and* $\mathbf{b}$*.*

Let $\mathbf{f}(\mathbf{0}) = (f_1(\mathbf{0}), \ldots, f_j(\mathbf{0}), \ldots, f_n(\mathbf{0}))^\top \in \mathbb{R}^n$ be an image with n pixels in row-wise form, and let $\mathbf{f}(\mathbf{t}) = (f_1(\mathbf{t}), \ldots, f_j(\mathbf{t}), \ldots, f_n(\mathbf{t}))^\top \in \mathbb{R}^n$ be a translated version of it by $\mathbf{t} = (h, v)^\top$ pixels. Clearly, $f_j\colon \mathbb{R}^2 \to \mathbb{R}$ denotes the intensity function of the j-th pixel, after a translation by $\mathbf{t}$. Fig. C.1 illustrates this case of study.

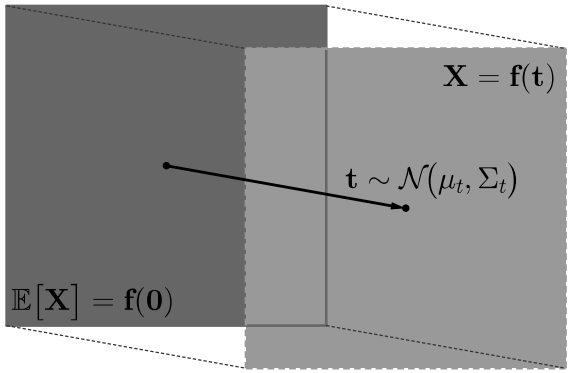


Figure C.1: Image translation by a random vector $\mathbf{t}$.

We will use the multivariate Taylor's theorem in order to approximate the intensity function of the j-th pixel of the given image; i.e., function f_j. That is, the intensity is approximated as follows

$$f_j(\mathbf{t}) = f_j(\mathbf{0}) + \nabla^\top f_j(\mathbf{0})\mathbf{t}.$$

Then,

$$\mathbf{f}(\mathbf{t}) = \mathbf{f}(\mathbf{0}) + \begin{pmatrix} \nabla^\top f_1(\mathbf{0}) \\ \vdots \\ \nabla^\top f_n(\mathbf{0}) \end{pmatrix} \mathbf{t}. \tag{C.2}$$

Let us now assume that $\mathbf{t}$ is a random vector distributed normally with mean $\boldsymbol{\mu}_t$ and covariance matrix Σ_t, i.e. $\mathbf{t} \sim \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)$. Then, $\mathbf{X} = \mathbf{f}(\mathbf{t})$ is also distributed normally with mean vector and covariance matrix that are given, respectively, by

$$\boldsymbol{\mu} = \mathbb{E}\big[\mathbf{X}\big] = \mathbf{f}(\mathbf{0}) + \begin{pmatrix} \nabla^\top f_1(\mathbf{0}) \\ \vdots \\ \nabla^\top f_n(\mathbf{0}) \end{pmatrix} \mathbb{E}\big[\mathbf{t}\big], \tag{C.3}$$

and

$$\Sigma = \mathbb{E}\Big[\big(\mathbf{X} - \boldsymbol{\mu}\big)\big(\mathbf{X} - \boldsymbol{\mu}\big)^\top\Big] = \begin{pmatrix} \nabla^\top f_1(\mathbf{0}) \\ \vdots \\ \nabla^\top f_n(\mathbf{0}) \end{pmatrix} \Sigma_t \begin{pmatrix} \nabla^\top f_1(\mathbf{0}) \\ \vdots \\ \nabla^\top f_n(\mathbf{0}) \end{pmatrix}^\top. \tag{C.4}$$

Thus, by setting $\mathbf{t} \sim \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)$, it holds that $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where the mean vector $\boldsymbol{\mu}$ and the covariance matrix Σ are given by (C.3) and (C.4), respectively.

Bibliography

[1] Creative commons attribution license. https://creative-commons.org/, December 2015. 49

[2] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical programming*, 95(1):3–51, 2003. 13

[3] E. Apostolidis and V. Mezaris. Fast shot segmentation combining global and local visual descriptors. In *Proc. of Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6583–6587. IEEE, 2014. 48

[4] A. Atanov, A. Ashukha, D. Molchanov, K. Neklyudov, and D. Vetrov. Uncertainty estimation via stochastic batch normalization. *arXiv preprint arXiv:1802.04893*, 2018. 17

[5] A. Barbu, L. Lu, H. Roth, A. Seff, and R. M. Summers. An analysis of robust cost functions for cnn in computer-aided diagnosis. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, pages 1–6, 2016. 15

[6] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998. 13

[7] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 35(8):1798–1828, 2013. 3, 11

[8] L. Berrada, A. Zisserman, and M. P. Kumar. Smooth loss functions for deep top-k classification. *arXiv preprint arXiv:1802.07595*, 2018. 15

[9] D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011. 13

[10] S. Bhattacharya, B. Nojavanasghari, T. Chen, D. Liu, S.-F. Chang, and M. Shah. Towards a comprehensive computational model foraesthetic assessment of videos. In *Proc. of the 21st Int. Conf. on Multimedia*, pages 361–364. ACM, 2013. 15, 50

[11] C. Bhattacharyya, P. K. Shivaswamy, and A. J. Smola. A second order cone programming formulation for classifying missing data. In *Advances in Neural Information Processing Systems*, 2004. 13

[12] J. Bi and T. Zhang. Support vector classification with input data uncertainty. In *Advances in Neural Information Processing Systems*, 2004. 8, 1, 12, 16, 29, 30, 31, 33, 34, 35, 36, 37

[13] R. Bolles, B. Burns, J. Herson, et al. The 2014 SESAME multimedia event detection and recounting system. In *Proc. TRECVID Workshop*, 2014. 14, 43, 44

[14] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard, et al. Comparison of classifier methods: a case study in handwritten digit recognition. In *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, volume 2, pages 77–82. IEEE, 1994. 28

[15] J. Bradshaw, A. G. d. G. Matthews, and Z. Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017. 17

[16] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990. 54, 56

[17] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 22, 40

[18] H. Cheng, J. Liu, I. Chakraborty, G. Chen, Q. Liu, M. Elhoseiny, G. Gan, A. Divakaran, H. Sawhney, J. Allan, J. Foley, M. Shah, A. Dehghan, M. Witbrock, and J. Curtis. SRI-Sarnoff AURORA system at TRECVID 2014 multimedia event detection and recounting. In *Proc. TRECVID Workshop*, 2014. 14, 43, 44, 46

[19] M. Cho and J. Lee. Riemannian approach to batch normalization. In *Advances in Neural Information Processing Systems*, pages 5231–5241, 2017. 60

[20] N. R. Council et al. *Risk analysis and uncertainty in flood damage reduction studies*. National Academies Press, 2000. 3

[21] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. In *Advances in neural information processing systems*, pages 414–422, 2009. 12

[22] N. Cressie and H. Whitford. How to use the two sample t-test. *Biometrical Journal*, 28(2):131–148, 1986. 60

[23] F. De La Torre and M. J. Black. A framework for robust subspace learning. *Int. Journal of Computer Vision*, 54(1-3):117–142, 2003. 17

[24] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 37(2):408–423, 2015. 3, 11

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 43

[26] A. Diplaros, T. Gevers, and I. Patras. Combining color and shape information for illumination-viewpoint invariant object recognition. *IEEE Transactions on Image Processing*, 15(1):1–11, 2006. 8, 64

[27] G. Dorta, S. Vicente, L. Agapito, N. D. Campbell, and I. Simpson. Structured uncertainty prediction networks. *arXiv preprint arXiv:1802.07079*, 2018. 17

[28] M. Douze, D. Oneata, M. Paulin, C. Leray, N. Chesneau, D. Potapov, J. Verbeek, K. Alahari, Z. Harchaoui, L. Lamel, J.-L. Gauvain, C. A. Schmidt, and C. Schmid. The INRIA-LIM-VocR and AXES submissions to TRECVID 2014 multimedia event detection. 2014. 14

[29] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning*, pages 264–271. ACM, 2008. 12

[30] J.-X. Du, C.-M. Zhai, Y.-L. Guo, Y.-Y. Tang, and P. C. C. Lung. Recognizing complex events in real movies by combining audio and video features. *Neurocomputing*, 137:89–95, 2014. 14

[31] L. S. Dutt and M. Kurian. Handling of uncertainty–a survey. *International Journal of Scientific and Research Publications*, 3(1):1–4, 2013. 3

[32] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016. 17

[33] A. Ghio, D. Anguita, L. Oneto, S. Ridella, and C. Schatten. Nested sequential minimal optimization for support vector machines. In *Artificial Neural Networks and Machine Learning–ICANN 2012*, pages 156–163. Springer, 2012. 29

[34] F. Gkalelis, Nikolaos andMarkatopoulou, A. Moumtzidou, D. Galanopoulos, K. Avgerinakis, N. Pittaras, S. Vrochidis, V. Mezaris, I. Kompatsiaris, and I. Patras. Iti-certh participation to trecvid 2014. In *Proceedings TRECVID Workshop*, 2014. 44

[35] N. Gkalelis and V. Mezaris. Video event detection using generalized subclass discriminant analysis and linear support vector machines. In *Proceedings of international conference on multimedia retrieval*, page 25. ACM, 2014. 14

[36] N. Gkalelis, V. Mezaris, I. Kompatsiaris, and T. Stathaki. Mixture subclass discriminant analysis link to restricted gaussian model and other generalizations. *Neural Networks and Learning Systems, IEEE Trans. on*, 24(1):8–21, 2013. 36

[37] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012. 44

[38] Grand Challenge at ACM Multimedia Conf. (MM'13). NHK Where is beauty? http://acmmm13.org/subm-issions/call-for-multimedia-grand-challenge-solutions/ task-where-is-beauty/, Barcelona, Spain, October 2013. 48

[39] Y. Guangnan, L. Dong, C. Shih-Fu, S. Ruslan, M. Vlad, D. Larry, G. Abhinav, H. Ismail, G. Sadiye, and M. Ashutosh. BBN VISER TRECVID 2014 multimedia event detection and multimedia event recounting systems. In *Proc. TRECVID Workshop*, 2014. 14, 43, 46

[40] P. Gurevich and H. Stuke. Learning uncertainty in regression tasks by deep neural networks. *arXiv preprint arXiv:1707.07287*, 2017. 17

[41] A. Habibian, T. Mensink, and C. G. Snoek. Videostory: A new multimedia embedding for few-example recognition and translation of events. In *Proceedings of the ACM International Conference on Multimedia*, pages 17–26. ACM, 2014. 14

[42] A. Habibian, K. E. van de Sande, and C. G. Snoek. Recommendations for video event recognition using concept vocabularies. In *Proc. of the 3rd ACM Conf. on Int. Conf. on multimedia retrieval*, pages 89–96. ACM, 2013. 14

[43] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *The Journal of Machine Learning Research*, 5:1391–1415, 2004. 20, 22

[44] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 15, 53, 54, 55, 61

[45] W. W. Hines, D. C. Montgomery, and D. M. G. C. M. Borror. *Probability and statistics in engineering*. John Wiley & Sons, 2008. 31, 33

[46] S. C. H. Hoi, J. Wang, and P. Zhao. Exact soft confidence-weighted learning. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012. 12

[47] L. Huang, X. Liu, B. Lang, and B. Li. Projection based weight normalization for deep neural networks. *arXiv preprint arXiv:1710.02338*, 2017. 60

[48] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems*, pages 487–493, 1999. 11

[49] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 54

[50] E. T. Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003. 3

[51] L. Jiang, D. Meng, T. Mitamura, and A. G. Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the ACM International Conference on Multimedia*, pages 547–556. ACM, 2014. 14

[52] L. Jiang, S.-I. Yu, D. Meng, T. Mitamura, and A. G. Hauptmann. Bridging the ultimate semantic gap: A semantic search engine for internet videos. In *ACM Int. Conf. on Multimedia Retrieval*, 2015. 14, 35

[53] Y.-G. Jiang, S. Bhattacharya, S.-F. Chang, and M. Shah. High-level event recognition in unconstrained videos. *International Journal of Multimedia Information Retrieval*, 2(2):73–101, 2013. 43

[54] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *Computer Vision and Pattern Recognition, Conf. on*, pages 2372–2379. IEEE, 2009. 3, 11

[55] S. M. Kakade and A. Tewari. On the generalization ability of online strongly convex programming algorithms. In *Advances in Neural Information Processing Systems*, pages 801–808, 2009. 25, 42

[56] R. Khemchandani, S. Chandra, et al. Twin support vector machines for pattern classification. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 29(5):905–910, 2007. 13

[57] G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971. 7, 41

[58] S. Koelstra, C. Mühl, M. Soleymani, J.-S. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt, and I. Patras. DEAP: A database for emotion analysis; using physiological signals. *Affective Computing, IEEE Trans. on*, 3(1):18–31, 2012. 33, 34

[59] M. Koestinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2288–2295. IEEE, 2012. 12

[60] A. N. Kolmogorov. Foundations of the theory of probability. 1950. 3

[61] A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014. 59

[62] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 54, 55, 61

[63] G. R. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *The Journal of Machine Learning Research*, 3:555–582, 2003. 13

[64] C. Leibig, V. Allken, M. S. Ayhan, P. Berens, and S. Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):17816, 2017. 17

[65] M. Li and I. K. Sethi. Confidence-based active learning. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 28(8):1251–1261, 2006. 12, 17

[66] Y. Li, J. Chen, and L. Feng. Dealing with uncertainty: a survey of theories and practices. *Knowledge and Data Engineering, IEEE Trans. on*, 25(11):2463–2482, 2013. 3, 11

[67] Z. Liang, N. Inoue, and K. Shinoda. Event detection by velocity pyramid. In *MultiMedia Modeling*, pages 353–364. Springer, 2014. 14

[68] M. Lichman. UCI machine learning repository, 2013. 31

[69] M. Lin, Q. Chen, and S. Yan. Network in network. In *International Conference on Learning Representations*, 2014. 53

[70] C. Liu, C. Hu, Q. Liu, and J. Aggarwal. Video event description in scene context. *Neurocomputing*, 119:82–93, 2013. 14

[71] T. Liu and D. Tao. Classification with noisy labels by importance reweighting. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 38(3):447–461, 2016. 12

[72] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2017. 16, 17, 54

[73] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, pages 507–516, 2016. 15, 54

[74] S. Liwicki, S. Zafeiriou, G. Tzimiropoulos, and M. Pantic. Efficient online subspace learning with an indefinite kernel for visual tracking and recognition. *Neural Networks and Learning Systems, IEEE Trans. on*, 23(10):1624–1636, 2012. 17

[75] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. Uncorrelated multilinear principal component analysis for unsupervised multilinear subspace learning. *Neural Networks, IEEE Trans. on*, 20(11):1820–1836, 2009. 17

[76] Y. Luo and X. Tang. Photo and video quality evaluation: Focusing on the subject. In *Proc. of the 10th European Conference on Computer Vision (ECCV), Marseille, France*, pages 386–399. Springer, 2008. 15, 49

[77] O. L. Mangasarian and E. W. Wild. Multisurface proximal support vector machine classification via generalized eigenvalues. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 28(1):69–74, 2006. 13

[78] E. Mavridaki and V. Mezaris. A comprehensive aesthetic quality assessment method for natural images using basic rules of photography. In *Proc. of Int. Conf. on Image Processing (ICIP)*, pages 887–891. IEEE, 2015. 48, 49, 50

[79] M. Mazloom, A. Habibian, D. Liu, C. G. Snoek, and S.-F. Chang. Encoding concept prototypes for video event detection and summarization. 2015. 14

[80] Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. 36

[81] A. K. Moorthy, P. Obrador, and N. Oliver. Towards computational models of the visual aesthetic appeal of consumer videos. In *Proc. of the 11th European Conference on Computer Vision (ECCV), Heraklion, Crete, Greece, 2010*, pages 1–14. Springer, 2010. 14, 49

[82] Y. Niu and F. Liu. What makes a professional video? a computational aesthetics approach. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(7):1037–1049, 2012. 15, 49, 50

[83] P. Over, G. Awad, J. Fiscus, M. Michel, D. Joy, A. F. Smeaton, W. Kraaij, G. Qu´enot, and R. Ordelman. TRECVID 2015 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proc. of TRECVID 2015*. NIST, USA, 2015. 35, 36

[84] P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, W. Kraaij, A. F. Smeaton, and G. Quéenot. TRECVID 2014 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proc. of TRECVID 2014*. NIST, USA, 2014. 14, 44

[85] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999. 22

[86] C. Qi and F. Su. Contrastive-center loss for deep neural networks. *arXiv preprint arXiv:1707.07391*, 2017. 15

[87] Z. Qi, Y. Tian, and Y. Shi. Robust twin support vector machine for pattern classification. *Pattern Recognition*, 46(1):305–316, 2013. 8, 1, 13, 16, 17, 29, 30, 31, 33, 34, 35, 36, 37

[88] S. Robertson. A new interpretation of average precision. In *Proc. of the 31st annual Int. ACM SIGIR Conf. on Research and development in information retrieval*, pages 689–690. ACM, 2008. 44, 49

[89] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, 2004. 22

[90] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 36

[91] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869, 2017. 53

[92] I. Sarafis, C. Diou, and A. Delopoulos. Building effective svm concept detectors from clickthrough data for large-scale image retrieval. *Int. Journal of Multimedia Information Retrieval*, 4(2):129–142, 2015. 12

[93] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Computational learning theory*, pages 416–426. Springer, 2001. 7, 41

[94] C. Sentelle, G. C. Anagnostopoulos, and M. Georgiopoulos. A simple method for solving the SVM regularization path for semidefinite kernels. *Neural Networks and Learning Systems, IEEE Trans. on*, 27(4):709–722, 2016. 4

[95] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 807–814, 2007. 42

[96] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical programming*, 127(1):3–30, 2011. 6, 7, 9, 17, 22, 25, 39, 42

[97] P. K. Shivaswamy, C. Bhattacharyya, and A. J. Smola. Second order cone programming approaches for handling missing and uncertain data. *The Journal of Machine Learning Research*, 7:1283–1314, 2006. 13, 16

[98] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 43, 53, 54, 55, 61

[99] F. Solera, S. Calderara, and R. Cucchiara. Socially constrained structural learning for groups detection in crowd. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 38(5):995–1008, 2016. 4

[100] X. Song, L. Sun, J. Lei, D. Tao, G. Yuan, and M. Song. Event–based large scale surveillance video summarization. *Neurocomputing, 2015*, 2015. 14

[101] D. Sundgren and A. Karlsson. Uncertainty levels of second-order probability. *Polibits*, (48):5–11, 2013. 3

[102] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9, 2015. 36, 53

[103] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 53, 54, 55, 61

[104] Y. Tang. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*, 2013. 15, 17

[105] S. Theodoridis, K. Koutroumbas, et al. Pattern recognition., 1999. 6

[106] B. S. Tsirelson, I. A. Ibragimov, and V. N. Sudakov. *Norms of Gaussian sample functions*, pages 20–41. Springer Berlin Heidelberg, Berlin, Heidelberg, 1976. 25

[107] C. Tzelepis, D. Galanopoulos, V. Mezaris, and I. Patras. Learning to detect video events from zero or very few video examples. *Image and vision Computing*, 53:35–44, 2016. 9, 12, 39

[108] C. Tzelepis, N. Gkalelis, V. Mezaris, and I. Kompatsiaris. Improving event detection using related videos and relevance degree support vector machines. In *Proc. of the 21st ACM Int. Conf. on Multimedia*, pages 673–676. ACM, 2013. 9, 12, 14, 36, 39, 42, 44, 45, 46, 50, 51

[109] C. Tzelepis, V. Mezaris, and I. Patras. Video event detection using kernel support vector machine with isotropic gaussian sample uncertainty (KSVM-iGSU). In *Proc. of the 22nd Int. Conf. on MultiMedia Modeling (MMM), Miami, FL, USA*, pages 3–15. Springer, 2016. 44

[110] C. Tzelepis, V. Mezaris, and I. Patras. Linear maximum margin classifier for learning from uncertain data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 17, 40, 44, 45, 51

[111] V. Vapnik. *The nature of statistical learning theory*. Springer Heidelberg, 1995. 4

[112] A. Vyas, R. Kannao, V. Bhargava, and P. Guha. Commercial block detection in broadcast news videos. In *Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing*, page 63. ACM, 2014. 1, 34, 35

[113] H. Wang, Y. Wang, Z. Zhou, X. Ji, Z. Li, D. Gong, J. Zhou, and W. Liu. Cosface: Large margin cosine loss for deep face recognition. *arXiv preprint arXiv:1801.09414*, 2018. 15, 16, 17, 54

[114] Y. Wang, Q. Dai, R. Feng, and Y.-G. Jiang. Beauty is here: Evaluating aesthetics in videos using multimodal features and free training data. In *Proc. of the 21st ACM Int. Conf. on Multimedia*, pages 369–372. ACM, 2013. 15

[115] P. D. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Trans. on Audio and Electroacoustics*, 15(2):70–73, 1967. 33

[116] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016. 16

[117] L. Xie, Q. Tian, J. Flynn, J. Wang, and A. Yuille. Geometric neural phrase pooling: Modeling the spatial co-occurrence of neurons. In *European Conference on Computer Vision*, pages 645–661. Springer, 2016. 1, 16, 60, 61

[118] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016. 53

[119] H. Xu, C. Caramanis, and S. Mannor. Robustness and regularization of support vector machines. *The Journal of Machine Learning Research*, 10:1485–1510, 2009. 13

[120] H. Xu and S. Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012. 13

[121] C.-Y. Yang, H.-H. Yeh, and C.-S. Chen. Video aesthetic quality assessment by combining semantically independent and dependent features. In *Proc. of Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1165–1168. IEEE, 2011. 15, 49, 50

[122] H.-H. Yeh, C.-Y. Yang, M.-S. Lee, and C.-S. Chen. Video aesthetic quality assessment by temporal integration of photo-and motion-based features. *IEEE Transactions on Multimedia*, 15(8):1944–1957, 2013. 15, 48, 49, 50

[123] S.-I. Yu, L. Jiang, Z. Mao, X. Chang, X. Du, C. Gan, Z. Lan, Z. Xu, X. Li, Y. Cai, et al. Informedia at TRECVID 2014 MED and MER. In *NIST TRECVID Video Retrieval Evaluation Workshop*, 2014. 43

[124] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1, 9, 15, 53, 54, 55, 58, 59, 60, 61

[125] T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, pages 56–85, 2004. 22

[126] W. Zhang, X. Y. Stella, and S.-H. Teng. Power SVM: Generalization with exemplar classification uncertainty. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conf. on*, pages 2144–2151. IEEE, 2012. 8, 1, 12, 13, 16, 17, 29, 30, 31, 33, 34, 35, 36, 37

[127] H.-J. Zimmermann. Uncertainty modelling and fuzzy sets. *Mathematical Research*, 99:84–100, 1997. 3

[128] G. Zoumpourlis, A. Doumanoglou, N. Vretos, and P. Daras. Non-linear convolution filters for cnn-based learning. *arXiv preprint arXiv:1708.07038*, 2017. 15, 53

[129] M. D. Zúñiga, F. Bremond, and M. Thonnat. Hierarchical and incremental event learning approach based on concept formation models. *Neurocomputing*, 100:3–18, 2013. 14