Hash Kernels and Structured Learning

Qinfeng (Javen) Shi

January 2011

A thesis submitted for the degree of Doctor of Philosophy
of the Australian National University

For my parents, my brother and my girlfriend.

Declaration

The work in this thesis is my own except where otherwise stated.

Qinfeng Shi

Acknowledgements

I am profoundly grateful to my current primal supervisor Tiberio Caetano for his invariable support and encouragement. It is fair to say that I won't be who I am without him.

I would like to thank to my original primal supervisor Alex Smola, from who I have learnt not only techiniques, but also determination. Unfortunately he left ANU and NICTA for Yahoo! Research in the 2nd year of my PhD.

Thank to Vishy who taught me Conditional Random Fields in the 1st year of my PhD. His home dinners gave us not only the food but also a warm welcome. Thanks to Li Cheng and Richard Hartley. Li taught me how to do a computer vision paper and Richard taught me graph cut.

A special thank you to Mark Reid whom I see as a dear mentor and collaborator, though he has no formal arrangement with the supervision. Mark's rigorous math background, effective communication skill and kindness help me improve my theoretical skills considerably.

Thank to Leon Bottou from who I learnt a serious research attitude. Thank to Vladimir Vapnik for his philosophy point of view of research methodology.

I would like to express my appreciation and gratitude to all my collaborators for their excellent ideas and dedication: Junbin Gao, James Petterson, Hanxi Li, Chunhua Shen, Gideon Dror, John Langford, Li Wang and Yasemin Altun.

I would also like to thank to my lab mates for their hard work and friendships: Choonhui Teo, Dmitry Kamenetsky, Jin Yu, Novi Quadrianto, Julian McAuley, Xinhua Zhang, Chris Webers, Marconi Barbosa, Justin Bedo, Scott Sanner, Edwin Bonilla, Hao Shen, Le Song, Owen Thomas, Tao Wang, Matthew Robards, Shengbo Guo, Fangfang Lu, Tim Sears and so on. I see many traditional merits in Choonhui and Jin. Thank to Dmitry, Novi, Owen and Marconi who have brought a lot of fun to our lab. I am impressed with James who is so well organised, which may explain why he can join so many clubs and still be very productive. Thanks a lot to Julian for his help with my English writing.

Many thanks to my other friends who have nothing to do with academia. I will not list all of you here, but my gratitude to you is immense.

Abstract

Vast amounts of data being generated, how to process massive data remains a challenge for machine learning algorithms. We propose hash kernels to facilitate efficient kernels which can deal with massive multi-class problems. We show a principled way to compute the kernel matrix for data streams and sparse feature spaces. We further generalise it via sampling to graphs. Later we exploit the connection between hash kernels with compressed sensing, and apply hashing to face recognition which significantly speeds up over the state-of-the-art with competitive accuracy. And we give a recovery rate on the sparse representation and a bounded recognition rate.

As hash kernels can deal with data with structures in the input such as graphs and face images, the second part of the thesis moves on to an even more challenging task — dealing with data with structures in the output.

Recent advances in machine learning exploit the dependency among data output, hence dealing with complex, structured data becomes possible. We study the most popular structured learning algorithms and categorise them into two categories — probabilistic approaches and Max Margin approaches. We show the connections of different algorithms, reformulate them in the empirical risk minimisation framework, and compare their advantages and disadvantages, which help choose suitable algorithms according to the characteristics of the application.

We have made practical and theoretical contributions in this thesis.

We show some real-world applications using structured learning as follows: a) We propose a novel approach for automatic paragraph segmentation, namely training Semi-Markov models discriminatively using a Max-Margin method. This method allows us to model the sequential nature of the problem and to incorporate features of a whole paragraph, such as paragraph coherence which cannot be used in previous models. b) We jointly segment and recognise actions in video sequences with a discriminative semi-Markov model framework, which incorporates features that capture the characteristics on boundary frames, action

segments and neighbouring action segments. A Viterbi-like algorithm is devised to help efficiently solve the induced optimisation problem. c) We propose a novel hybrid loss of Conditional Random Fields (CRFs) and Support Vector Machines (SVMs). We apply the hybrid loss to various applications such as Text chunking, Named Entity Recognition and Joint Image Categorisation.

We have made the following theoretical contributions: a) We study the recent advance in PAC-Bayes bounds, and apply it to structured learning. b) We propose a more refined notion of Fisher consistency, namely *Conditional Fisher Consistency for Classification* (CFCC), that conditions on the knowledge of the true distribution of class labels. c) We show that the hybrid loss has the advantages of both CRFs and SVMs — it is consistent and has a tight PAC-Bayes bound which shrinks as the margin increases. d) We also introduce Probabilistic margins which take the label distribution into account. And we show that many existing algorithms can be viewed as special cases of the new margin concept which may help understand existing algorithms as well as design new algorithms.

At last, we discuss some future directions such as tightening PAC-Bayes bounds, adaptive hybrid losses and graphical model inference via Compressed Sensing.

Publications arising from this thesis

1. **Qinfeng Shi**, Hanxi Li and Chunhua Shen, **Rapid Face Recognition Using Hashing**, *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 10)*, San Francisco, USA, June 13-18, 2010.

2. **Qinfeng Shi**, Mark Reid, Tiberio Caetano, **Hybrid model of Conditional Random Field and Support Vector Machine**, *Workshop at the 23rd Annual Conference on Neural Information Processing Systems*, Canada, Dec. 2009.

3. **Qinfeng Shi**, James Petterson, Gideon Dror, John Langford, Alex Smola and Vishy Vishwanathan, **Hash Kernels for Structured Data**, *Journal of Machine Learning Research*, Nov. 2009.

4. **Qinfeng Shi**, Li Wang, Li Cheng and Alex Smola, **Discriminative Human Action Segmentation and Recognition using Semi-Markov Model**, (long version cutting plane v.s. Bundle Method), *International Journal of Computer Vision*, Accepted under minor revision, First submitted in Dec. 2008. Revised in 2010.

5. **Qinfeng Shi**, Li Cheng, Luping Zhou and Dale Schuurmans, **Discriminative Maximum Margin Image Object Categorization with Exact Inference**, The 5th International Conference on Image and Graphics, Xi'an, Sep. 20-23, 2009.

6. **Qinfeng Shi**, James Petterson,Gideon Dror, John Langford, Alex Smola, Alex Strehl, and Vishy Vishwanathan, **Hash Kernel**, *Twelfth International Conference on Artificial Intelligence and Statistics*, Florida, April 14-19, 2009.

7. **Qinfeng Shi**, Li Wang, Li Cheng and Alex Smola, **Discriminative Human Action Segmentation and Recognition using Semi-Markov Model**, *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 08)*, Anchorage, Alaska, June 23-28, 2008.

8. **Qinfeng Shi**, Yasemin Altun , Alex Smola and S.V.N. Vishwanathan, **Automatic Paragraph Segmentation via Max-Margin Semi-Markov Models**, *EMNLP-CoNLL*, Jun, 2007, pp. 640-648.

Contents

List of Tables

List of Figures

Notation and Terminology

Notation

$\mathbb{N}, \mathbb{R}$	natural, real numbers
$\mathbb{E}$	expectation
$\mathcal{X}$	input space
$\mathcal{Y}$	output space
$\mathbf{x}$	structured or vectorial input
y	scalar output
$\mathbf{y}$	structured or vectorial output
$\mathbf{y}^{(i)}$	the component of the output $\mathbf{y}$ for the i-th node
ℓ	loss function
$\langle \mathbf{x}, \mathbf{z} \rangle$	inner product of $\mathbf{x}$ and $\mathbf{z}$
$\otimes$	tensor product
$\Phi(\mathbf{x}, \mathbf{y})$	mapping to feature space
$\overline{\Phi}(\mathbf{x}, \mathbf{y})$	hashed mapping to feature space
Φ	a random matrix via hashing used in compressed sensing
$\mathcal{F}$	class of real-valued functions
$f(\mathbf{x})$	real-valued function
$\mathbf{w}$	weight vector

$(x)_+$ or $[x]_+$	equals x, if $x \geq 0$ else 0
h	hash function or classification hypothesis
$\text{sgn}(x)$	equals 1, if $x \geq 0$ else -1
δ	confidence or Kronecker delta
γ	margin
ξ	slack variables
$\mathbf{1}$	identity matrix or indicator function
$\mathbf{K}$	kernel matrix
$k(\mathbf{x}, \mathbf{x}')$	kernel value of $\mathbf{x}$ and $\mathbf{x}'$
$\overline{\mathbf{K}}$	hash kernel matrix
$\overline{k}^h(\mathbf{x}, \mathbf{x}')$	hash kernel value of $\mathbf{x}$ and $\mathbf{x}'$ with hash function h
$\#A$ or $\|A\|$	cardinality of a set A
ln	natural logarithm
e	base of the natural log
η	learning rate
$\mathbf{A}$	a base matrix
$\mathbf{H}$	a random hashing matrix used in compressed sensing
$\mathbf{R}$	a random matrix (such as gaussian random matrix) used in compressed sensing
R_{emp}	empirical risk
R	true risk
P, Q	prior and posterior distributions over the hyperostosis h or parameter $\mathbf{w}$
$p(\cdot\,;\mathbf{w})$ or $\mathbf{P}_\mathbf{w}$	modeled probability/density parameterised by $\mathbf{w}$
$\mathbf{P}$ or Pr	probability or probability density

Part I

Introduction

Chapter 1

Introduction

Machine learning seeks a function that gives a good output from $y \in \mathcal{Y}$ when given an input $\mathbf{x} \in \mathcal{X}$. And the input-output pairs $(\mathbf{x}, y)$ are assumed I.I.D.(independent and identically distributed) drawn from a unknown but fixed underlying data distribution $\Pr(\mathbf{x}, y)$.

1.1 Massive Multi-class

The output can be a scalar y representing a class I.D. for a given input $\mathbf{x}$. The problem comes when there is a massive number of classes. For example, there are 71,000 classes for a website topic categorisation problem shown in Chapter 2. Traditional methods fail for the huge demand of computation and memory usage — for example, multi-class SVMs for the above problem need 96.95G memory just to store the model parameters.

1.2 Structured Label

In many cases, $(\mathbf{x}, y)$ are no longer I.I.D. So one often models those correlated ys as a structured output $\mathbf{y}$ with the assumption that $(\mathbf{x}, \mathbf{y})$ are I.I.D. drawn from $\mathbf{P}(\mathbf{x}, \mathbf{y})$. Here the output $\mathbf{y}$ can be any object associated with $\mathbf{x}$. For example, for an automated paragraph breaking problem, the input $\mathbf{x}$ is a document, and the output $\mathbf{y}$ is a sequence whose entries denote the beginning positions of the paragraphs. For image segmentation, the input $\mathbf{x}$ is an n by m image, and the output $\mathbf{y}$ is a 2-D lattice $\{\mathbf{y}^{i,j}\}_{1 \le i \le n; 1 \le j \le m}$, where $\mathbf{y}^{i,j}$ denotes class id of the pixel $\mathbf{x}^{i,j}$. The learning is called "structured learning" when some interdependency

structure between different parts of the output is exploited. In this case, the output $\mathbf{y}$ is no longer a scalar.

1.2.1 Importance of structured learning

Structured Learning has gained great success in many fields: bioinformatics, document analysis, computer vision, machine learning, sensor network such as Activities Daily living (ADLs) (Park and Kautz, 2008) monitoring and so on. The reason that structured learning outperforms the I.I.D learning, is that the former is capable of modelling complex dependencies of the entries of the output and the dependency does exist in real world applications pervasively. For example in image segmentation, we know that neighbouring pixels are most likely to belong to the same class. Also in human action recognition, we want to predict the current action (walking, running, jumping, ...) at every second. Apparently the current action heavily depends on the predicted action at the previous second, because the actions at two consecutive seconds are likely to be the same; also there are some physical constraints on human movement meaning that it is impossible for a human to immediately switch from a certain action to others. Thus predicting the actions jointly for a given time period is better than predicting the action at each second separately.

1.2.2 Difficulties and Challenges

While structured learning brings superior performance than that of I.I.D. learning, it brings more difficulties and challenges as well.

1. **Inference is slow:** the inference for structured learning is much more expensive than that for I.I.D. learning. For example, for an action sequence with 1000 seconds/frames and 5 actions, there are 5^{1000} possible label values. It makes it impossible to enumerate all possible outputs $\mathbf{y}$, whereas in I.I.D. learning, regardless of how many seconds the action lasts, there are only 5 possible values of y. For a subset of structured learning problems, when the output structure is a chain or a tree, dynamic programming or belief propagation (BP) can exactly infer the best possible label efficiently (linearly in the number of nodes in the structure). For more general structures like nets, there isn't a known exact polynomial algorithm for inference. For example, the complexity of the junction tree algorithm is exponential in the tree-width. Alternatively, many approximate inference algorithms

have been developed, such as loopy belief propagation (LBP) and variational methods. For some very large graphical models, even approximation methods are too slow. See Lauritzen (1996); Jordan (2008); Wainwright and Jordan (2003); Bishop (2006) for details.

2. **Training is slow and often approximation is needed:** during training, some gradient calculation often requires inference: in max margin approaches, finding the most violated example is an inference problem (see Section 4.4); In maximum likelihood approaches (see Section 4.3) such as conditional random fields, computing the pointwise and pairwise distribution is also an inference problem. Approximate inference in loopy graphs results in an approximate gradient, which harms the training (see Lauritzen (1996); Jordan (2008); Wainwright and Jordan (2003); Bishop (2006)).

3. **Existing generalisation bounds are loose:** despite good experimental results of structured learning reported in the literature, there is little known about the generalisation bound. This is because the output space is so huge that traditional techniques (such as VC dimension, Rademacher bound, margin bound, PAC-Bayes bound) yield useless bounds (see Section 8.2): for example, the bounds are easily bigger than or nearly 1.

4. **Fisher Consistency might be too coarse to explain real performance of the algorithms:** many algorithms that are Fisher consistent in the I.I.D. binary case are no longer Fisher consistent in the structured output case. Empirical studies on those algorithms are controversial — they work well in some applications and fail in others. Yet there is little known about why they behave this way. We argue that this is related to their Fisher (in)consistency (see Section 8.1). The existing Fisher consistency for classification (FCC) definition is too restrictive — an algorithm is only FCC when it is consistent for all data distributions, which is not able to provide useful insights.

1.3 Contribution

We try to tackle some of the challenges (but not all) as follows:

- In Chapter 2, we propose hash kernels (Shi et al., 2009a,b) to facilitate efficient kernels which can deal with massive multi-class problems with even

more than 7,000 classes, due to its memory footprint independence in the number of classes. We show a principled way to compute the kernel matrix for data streams and sparse feature spaces. We further generalise it via sampling to graphs.

- In Chapter 3, we exploit the connection between hash kernels and compressed sensing, and apply hashing to face recognition (Shi et al., 2010a) which significantly speeds up the state-of-the-art with competitive accuracy. And we give a recovery rate on the sparse representation and a bounded recognition rate.

- In Chapter 4, we categorise the most popular structured learning algorithms into two categories — probabilistic approaches and Max Margin approaches. And in fact many structured learning algorithms from both categories can be viewed in a unified framework, Empirical Risk Minimisation.

- In Chapter 5, we propose a novel approach for automatic paragraph segmentation (Shi et al., 2007) , namely training Semi-Markov models discriminatively using a Max-Margin method. This method allows us to model the sequential nature of the problem and to incorporate features of a whole paragraph, such as paragraph coherence which cannot be used in previous models.

- In Chapter 6, we jointly segment and recognise actions in video sequences with a discriminative semi-Markov model framework (Shi et al., 2008, 2009d), which incorporates features that capture the characteristics on boundary frames, action segments and neighbouring action segments. A Viterbi-like algorithm is devised to help efficiently solve the induced optimisation problem.

- In Chapter 7, we propose a novel hybrid loss (Shi et al., 2009c) of Conditional Random Fields (CRFs) and Support Vector Machines (SVMs). The hybrid loss has advantages of both CRFs and SVMs — it is consistent and has a tight PAC-Bayes bound which shrinks as the margin increases. We apply the hybrid loss to various applications (Shi et al., 2010b) such as Text chunking, Named Entity Recognition and Joint Image Categorisation.

- In Chapter 8, we study the recent advances in PAC-Bayes bounds, and apply them to structured learning. Moreover, we propose a more refined

notion of Fisher consistency, namely *Conditional Fisher Consistency for Classification* (CFCC) (Shi et al., 2010b), that conditions on the knowledge of true distribution of class labels. We also introduce Probabilistic margins which take the label distribution into account. We show that many existing algorithms can be viewed as special cases of the new margin concept which may help understand existing algorithms as well as design new algorithms.

Part II

Hash Kernels

Chapter 2

Efficient Hash Kernels

We propose hashing (Shi et al., 2009a) to facilitate efficient kernels which can deal with massive multi-class problems. We show a principled way to compute the kernel matrix for data streams and sparse feature spaces. We further generalise it via sampling to graphs (Shi et al., 2009b).

2.1 Introduction

In recent years, a number of methods have been proposed to deal with the fact that kernel methods have slow runtime performance if the number of kernel functions used in the expansion is large. We denote by $\mathcal{X}$ the domain of observations and we assume that $\mathcal{H}$ is a Reproducing Kernel Hilbert Space with kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

2.1.1 Keeping the Kernel Expansion Small

One line of research (Burges and Schölkopf, 1997) aims to reduce the number of basis functions needed in the overall function expansion. This led to a number of reduced set Support Vector algorithms which work as follows: a) solve the full estimation problem resulting in a kernel expansion, b) use a subset of basis functions to approximate the exact solution, c) use the latter for estimation. While the approximation of the full function expansion is typically not very accurate, very good generalisation performance is reported. The big problem in this approach is that the optimisation of the reduced set of vectors is rather nontrivial.

Work on estimation "on a budget" (Dekel et al., 2006) tries to ensure that this problem does not arise in the first place by ensuring that the number of

kernel functions used in the expansion never exceeds a given budget or by using an ℓ_1 penalty (Mangasarian, 1998). For some algorithms, for example, binary classification, guarantees are available in the online setting.

2.1.2 Keeping the Kernel Simple

A second line of research uses variants of sampling to achieve a similar goal. That is, one uses the feature map representation

$$k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle .$$

Here Φ maps $\mathcal{X}$ into some feature space $\mathcal{F}$. This expansion is approximated by a mapping $\overline{\Phi} : \mathcal{X} \to \overline{\mathcal{F}}$

$$\overline{k}(\mathbf{x}, \mathbf{x}') = \left\langle \overline{\Phi}(\mathbf{x}), \overline{\Phi}(\mathbf{x}') \right\rangle \text{ often } \overline{\Phi}(\mathbf{x}) = C\Phi(\mathbf{x}),$$

where $C \in \mathbb{R}$. Here $\overline{\Phi}$ has more desirable computational properties than Φ. For instance, $\overline{\Phi}$ is finite dimensional (Fine and Scheinberg, 2001; Kontorovich, 2007; Rahimi and Recht, 2008), or $\overline{\Phi}$ is particularly sparse (Li et al., 2007).

2.1.3 Our Contribution

Firstly, we show that the sampling schemes of Kontorovich (2007) and Rahimi and Recht (2008) can be applied to a considerably larger class of kernels than originally suggested—the authors only consider languages and radial basis functions respectively. Secondly, we propose a biased approximation $\overline{\Phi}$ of Φ which allows efficient computations even on data streams. Our work is inspired by the count-min sketch of Cormode and Muthukrishnan (2004), which uses hash functions as a computationally efficient means of randomisation. This affords storage efficiency (we need not store random vectors) and at the same time they give performance guarantees comparable to those obtained by means of random projections.

As an application, we demonstrate computational benefits over suffix array string kernels in the case of document analysis and we discuss a kernel between graphs which only becomes computationally feasible by means of a compressed representation.

2.1.4 Outline

We begin with a description of previous work in Section 2.2 and propose hash kernels in Section 2.3 which are suitable for data with simple structure such as

strings. Analysis follows in Section 2.4. We propose a graphlet kernel which generalises hash kernels to data with general structure—graphs—and discuss a MCMC sampler in Section 2.5. Finally we conclude with experiments in Section 2.6.

2.2 Previous Work and Applications

Recently much attention has been given to efficient algorithms with randomisation or hashing in the machine learning community.

2.2.1 Generic Randomisation

Kontorovich (2007) and Rahimi and Recht (2008) independently propose the following: denote by $c \in \mathcal{C}$ a random variable with measure $\mathbf{P}$. Moreover, let $\Phi_c : \mathcal{X} \to \mathbb{R}$ be functions indexed by $c \in \mathcal{C}$. For kernels of type

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{c \sim \mathbf{P}(c)} \left[\Phi_c(\mathbf{x}) \Phi_c(\mathbf{x}') \right] \tag{2.1}$$

an approximation can be obtained by sampling $C = \{c_1, \ldots, c_n\} \sim \mathbf{P}$ and expanding

$$\overline{k}(\mathbf{x}, \mathbf{x}') = \frac{1}{n} \sum_{i=1}^{n} \Phi_{c_i}(\mathbf{x}) \Phi_{c_i}(\mathbf{x}').$$

In other words, we approximate the feature map $\Phi(\mathbf{x})$ by

$$\overline{\Phi}(\mathbf{x}) = n^{-\frac{1}{2}} \left(\Phi_{c_1}(\mathbf{x}), \ldots, \Phi_{c_n}(\mathbf{x}) \right)$$

in the sense that their resulting kernel is similar. Assuming that $\Phi_c(\mathbf{x})\Phi_c(\mathbf{x}')$ has bounded range, that is, $\Phi_c(\mathbf{x})\Phi_c(\mathbf{x}') \in [a, a + R]$ for all c, $\mathbf{x}$ and $\mathbf{x}'$ one may use Chernoff bounds to give guarantees for large deviations between $k(\mathbf{x}, \mathbf{x}')$ and $\overline{k}(\mathbf{x}, \mathbf{x}')$. For matrices of size $m \times m$ one obtains bounds of type $O(R^2 \epsilon^{-2} \log m)$ by combining Hoeffding's theorem with a union bound argument over the $O(m^2)$ different elements of the kernel matrix. The strategy has widespread applications beyond those of Kontorovich (2007) and Rahimi and Recht (2008):

- Kontorovich (2007) uses it to design kernels on regular languages by sampling from the class of languages.

- The marginalised kernels of Tsuda et al. (2002) use a setting identical to (2.1) as the basis for comparisons between strings and graphs by defining a random walk as the feature extractor. Instead of exact computation we could do sampling.

- The Binet-Cauchy kernels of Vishwanathan et al. (2007b) use this approach to compare trajectories of dynamical systems. Here c is the (discrete or continuous) time and $\mathbf{P}(c)$ discounts over future events.

- The empirical kernel map of Schölkopf (1997) uses $\mathcal{C} = \mathcal{X}$ and employs some kernel function κ to define $\Phi_c(\mathbf{x}) = \kappa(c, \mathbf{x})$. Moreover, $\mathbf{P}(c) = \mathbf{P}(\mathbf{x})$, that is, placing our sampling points c_i on training data.

- For RBF kernels, Rahimi and Recht (2008) use the fact that k may be expressed in the system of eigenfunctions which commute with the translation operator, that is the Fourier basis

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\mathbf{w} \sim \mathbf{P}(\mathbf{w})}[e^{-i\langle \mathbf{w}, \mathbf{x} \rangle} e^{i\langle \mathbf{w}, \mathbf{x}' \rangle}]. \tag{2.2}$$

  Here $\mathbf{P}(w)$ is a nonnegative measure which exists for any RBF kernel by virtue of Bochner's theorem, hence (2.2) can be recast as a special case of (2.1). What sets it apart is the fact that the variance of the features $\Phi_{\mathbf{w}}(\mathbf{x}) = e^{i\langle \mathbf{w}, \mathbf{x} \rangle}$ is relatively evenly spread. (2.2) extends immediately to Fourier transformations on other symmetry groups (Berg et al., 1984).

- The conditional independence kernel of Watkins (2000) is one of the first instances of (2.1). Here $\mathcal{X}, \mathcal{C}$ are domains of biological sequences, $\Phi_c(\mathbf{x}) = \mathbf{P}(\mathbf{x}\,|c)$ denotes the probability of observing $\mathbf{x}$ given the ancestor c, and $\mathbf{P}(c)$ denotes a distribution over ancestors.

While in many cases straightforward sampling may suffice, it can prove disastrous whenever $\Phi_c(\mathbf{x})$ has only a small number of significant terms. For instance, for the pair-HMM kernel most strings c are *unlikely* ancestors of $\mathbf{x}$ and $\mathbf{x}'$, hence $\mathbf{P}(\mathbf{x}\,|c)$ and $\mathbf{P}(\mathbf{x}'\,|c)$ will be negligible for most c. As a consequence the number of strings required to obtain a good estimate is prohibitively large—we need to reduce $\overline{\Phi}$ further.

2.2.2 Locally Sensitive Hashing

The basic idea of randomised projections (Indyk and Motawani, 1998) is that due to concentration of measures the inner product $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$ can be approximated

by $\sum_{i=1}^{n} \langle v_i, \Phi(\mathbf{x}) \rangle \langle v_i, \Phi(\mathbf{x}') \rangle$ efficiently, provided that the distribution generating the vectors v_i satisfies basic regularity conditions. For example, $v_i \sim \mathcal{N}(0, I)$ is sufficient, where I is an identity matrix. This allows one to obtain Chernoff bounds and $O(\epsilon^{-2} \log m)$ rates of approximation, where m is the number of instances. The main cost is to store v_i and perform the $O(nm)$ multiply-adds, thus rendering this approach too expensive as a preprocessing step in many applications.

Achlioptas (2003) proposes a random projection approach that uses symmetric random variables to project the original feature onto a lower dimension feature space. This operation is simple and fast and the author shows it does not sacrifice the quality of the embedding. Moreover, it can be directly applied to online learning tasks. Unfortunately, the projection remains dense resulting in relatively poor computational and space performance in our experiments.

2.2.3 Sparsification

Li et al. (2007) propose to sparsify $\Phi(\mathbf{x})$ by randomisation while retaining the inner products. One problem with this approach is that when performing optimisation for linear function classes, the weight vector $\mathbf{w}$ which linearly parameterises $\Phi(\mathbf{x}_i)$ remains large and dense, thus obliterating a significant part of the computational savings gained in sparsifying Φ.

2.2.4 Count-Min Sketch

Cormode and Muthukrishnan (2004) propose an ingenious method for representing data streams. Denote by $\mathcal{I}$ an index set. Moreover, let $h : \mathcal{I} \to \{1, \ldots, n\}$ be a hash function and assume that there exists a distribution over h such that they are pairwise independent. That is, any pair of h and h' are independent to each other.

Assume that we draw d hash functions h_i at random and let $S \in \mathbb{R}^{n \times d}$ be a sketch matrix. For a stream of symbols s update $S_{h_i(s),i} \leftarrow S_{h_i(s),i} + 1$ for all $1 \leq i \leq d$. To retrieve the (approximate) counts for symbol s' compute $\min_i S_{h_i(s'),i}$. (Hence the name count-min sketch). The idea is that by storing counts of s according to several hash functions we can reduce the probability of collision with another particularly large symbol. Cormode and Muthukrishnan (2004) show that only $O(\epsilon^{-1} \log 1/\delta)$ storage is required for an ϵ-good approximation, where $1 - \delta$ is the confidence.

Cormode and Muthukrishnan (2004) discuss approximating inner products and the extension to signed rather than nonnegative counts. However, the bounds degrade for real-valued entries. Even worse, for the hashing to work, one needs to take the minimum over a set of inner product candidates.

2.2.5 Random Feature Mixing

Ganchev and Dredze (2008) provide empirical evidence that using hashing can eliminate alphabet storage and reduce the number of parameters without severely impacting model performance. In addition, Langford et al. (2007) released the "Vowpal Wabbit" fast online learning software which uses a hash representation similar to the one discussed here.

2.2.6 Hash Kernel on Strings

We propose a hash kernel (Shi et al., 2009a) to deal with the issue of computational efficiency by a very simple algorithm: high-dimensional vectors are compressed by adding up all coordinates which have the same hash value—one only needs to perform as many calculations as there are nonzero terms in the vector. The hash kernel can jointly hash both labels and features, thus the memory footprint is essentially independent of the number of classes used.

2.3 Hash Kernels

Our goal is to design a possibly biased approximation which a) approximately preserves the inner product, b) which is generally applicable, c) which can work on data streams, and d) which increases the density of the feature matrices (the latter matters for fast linear algebra on CPUs and graphics cards).

2.3.1 Kernel Approximation

As before denote by $\mathcal{I}$ an index set and let $h : \mathcal{I} \to \{1, \ldots, n\}$ be a hash function that maps $\mathcal{J}$ to $\{1, \ldots, n\}$ uniformly. Finally, assume that $\Phi(\mathbf{x})$ is indexed by $\mathcal{I}$ and that we may compute $\Phi_i(\mathbf{x})$ for all nonzero terms efficiently. In this case we define the hash kernel as follows:

$$\overline{k}(\mathbf{x}, \mathbf{x}') = \left\langle \overline{\Phi}(\mathbf{x}), \overline{\Phi}(\mathbf{x}') \right\rangle \text{ with } \overline{\Phi_j}(\mathbf{x}) = \sum_{i \in \mathcal{I}; h(i)=j} \Phi_i(\mathbf{x}) \tag{2.3}$$

We are accumulating all coordinates i of $\Phi(\mathbf{x})$ for which $h(i)$ generates the same value j into coordinate $\overline{\Phi}_j(\mathbf{x})$. Our claim is that hashing preserves information as well as randomised projections with significantly less computation. Before providing an analysis let us discuss two key applications: efficient hashing of kernels on strings and cases where the number of classes is very high, such as categorisation in an ontology. In (2.3) it seems that $\Phi(\mathbf{x})$ needs to be pre-computed as some feature vector such as term frequency (Jones, 1972). However, in practice we no longer need to build up vocabulary nor to compute $\Phi(\mathbf{x})$ explicitly as it will be clear in Section 2.6.1. The implementation of hash we used is Murmur Hash (Appleby, 2008). However, as long as the hash is good in the sense that the hash maps uniformly, it does not matter which one you use. It is just like in sampling technique, it does not matter what random number generator you use.

2.3.2 Strings

Denote by $\mathcal{X} = \mathcal{I}$ the domain of strings on some alphabet. Moreover, assume that $\Phi_i(\mathbf{x}) := \lambda_i \#_i(\mathbf{x})$ denotes the number of times the substring i occurs in $\mathbf{x}$, weighted by some coefficient $\lambda_i \geq 0$. This allows us to compute a large family of kernels via

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i \in \mathcal{I}} \lambda_i^2 \#_i(\mathbf{x}) \#_i(\mathbf{x}'). \tag{2.4}$$

Teo and Vishwanathan (2006) propose a storage efficient $O(|\,\mathbf{x}\,| + |\,\mathbf{x}'\,|)$ time algorithm for computing k for a *given* pair of strings $\mathbf{x}, \mathbf{x}'$. Here $|\,\mathbf{x}\,|$ denotes the length of the string. Moreover, a weighted combination $\sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x})$ can be computed in $O(|\,\mathbf{x}\,|)$ time after $O(\sum_i |\,\mathbf{x}_i\,|)$ preprocessing.

The big drawback with string kernels using suffix arrays/trees is that they require large amounts of working memory. Approximately a factor of 50 additional storage is required for processing and analysis. Moreover, updates to a weighted combination are costly. This makes it virtually impossible to apply (2.4) to millions of documents. Even for modest document lengths this would require Terabytes of RAM.

Hashing allows us to reduce the dimensionality. Since for every document $\mathbf{x}$ only a relatively small number of terms $\#_i(\mathbf{x})$ will have nonzero values—at most $O(|\,\mathbf{x}\,|^2)$ but in practice we will restrict ourselves to substrings of a bounded length l leading to a cost of $O(|\,\mathbf{x}\,| \cdot l)$—this can be done efficiently in a single pass over $\mathbf{x}$. Moreover, we can compute $\overline{\Phi}(\mathbf{x})$ as a pre-processing step and discard $\mathbf{x}$ altogether.

Note that this process spreads out the features available in a document *evenly* over the coordinates of $\overline{\Phi}(\mathbf{x})$. Moreover, note that a similar procedure can be used to obtain good estimates for a TF/IDF reweighting (Jones, 1972) of the counts obtained, thus rendering preprocessing as memory efficient as the actual computation of the kernel.

2.3.3 Multiclass Classification

Classification can sometimes lead to a very high dimensional feature vector even if the underlying feature map $\mathbf{x} \to \Phi(\mathbf{x})$ may be acceptable. For instance, for a bag-of-words representation (Lewis, 1998) of documents with 10^4 unique words and 10^3 classes this involves up to 10^7 coefficients to store the parameter vector directly when $\Phi(x, y) = e_y \otimes \Phi(\mathbf{x})$, where $\otimes$ is the tensor product and e_y is a vector whose y-th entry is 1 and the rest are zero. The dimensionality of e_y is the number of classes.

Note that in the above case $\Phi(\mathbf{x}, y)$ corresponds to a sparse vector which has nonzero terms only in the part corresponding to e_y. That is, by using the joint index (i, y) with $\Phi(\mathbf{x}, y)_{(i,y')} = \Phi_i(\mathbf{x})\delta_{y,y'}$ we may simply apply (2.3) to the joint index to obtain hashed versions of multiclass vectors. We have

$$\overline{\Phi_j}(\mathbf{x}, y) = \sum_{i \in \mathcal{I}; h(i,y)=j} \Phi_i(\mathbf{x}).$$

In some cases it may be desirable to compute a compressed version of $\Phi(\mathbf{x})$, that is, $\overline{\Phi}(\mathbf{x})$ first and subsequently expand terms with y. In particular for strings this can be useful since it means that we need not parse $\mathbf{x}$ for every potential value of y. While this deteriorates the approximation in an additive fashion it can offer significant computational savings since all we need to do is permute a given feature vector as opposed to performing any summations.

2.3.4 Streams

Some features of observations arrive as a stream. For instance, when performing estimation on graphs, we may obtain properties of the graph by using an MCMC sampler. The advantage is that we need not store the entire data stream but rather just use summary statistics obtained by hashing.

2.4 Analysis

We show that the penalty we incur from using hashing to compress the number of coordinates only grows *logarithmically* with the number of observations m and with the number of classes M, which will be shown in Theorem 3. While we are unable to obtain the excellent $O(\epsilon^{-1})$ rates offered by the count-min sketch, our approach retains the inner product property thus making hashing accessible to linear estimation.

2.4.1 Information Loss

One of the key fears of using hashing in machine learning is that hash collisions harm performance. When a collision occurs, information is lost, which may reduce the achievable performance for a predictor.

Definition 1 *(Information Loss) A hash function h causes information loss on a distribution D with a loss function L if the expected minimum loss before hashing is less than the expected minimum loss after hashing:*

$$\min_f \mathbb{E}_{(\mathbf{x},y)\sim D}\left[L(f(\mathbf{x}),y))\right] < \min_g \mathbb{E}_{(\mathbf{x},y)\sim D}\left[L(g(h(\mathbf{x})),y))\right].$$

Redundancy in features is very helpful in avoiding information loss. The redundancy can be explicit or systemic such as might be expected with a bag-of-words or substring representation. In the following we analyse explicit redundancy where a feature is mapped to two or more values in the space of size n. This can be implemented with a hash function by (for example) appending the string $i \in \{1, \ldots, c\}$ to feature f and then computing the hash of $f \circ i$ for the i-th duplicate.

The essential observation is that the information in a feature is only lost if all duplicates of the feature collide with other features. Given this observation, it's unsurprising that increasing the size of n by a constant multiple c and duplicating features c times makes collisions with all features unlikely. It's perhaps more surprising that when keeping the size of n *constant* and duplicating features, the probability of information loss can go *down*.

Theorem 2 *For a random function mapping l features duplicated c times into a space of size n, for all loss functions L and distributions D on n features, the probability (over the random function) of no information loss is at least:*

$$1 - l[1 - (1 - c/n)^c + (lc/n)^c].$$

For the proof see Appendix A.

To see the implications consider $l = 10^5$ and $n = 10^8$. Without duplication, a birthday paradox collision is virtually certain. However, if $c = 2$, the probability of information loss is bounded by about 0.404, and for $c = 3$ it drops to about 0.0117.

2.4.2 Rate of Convergence

As a first step note that any convergence bound only depends *logarithmically* on the size of the kernel matrix as follows.

Theorem 3 *Assume that the probability of deviation between the hash kernel and its expected value is bounded by an exponential inequality via*

$$\mathbf{P}\left[\left|\overline{k}^h(\mathbf{x}, \mathbf{x}') - \mathbb{E}_h\left[\overline{k}^h(\mathbf{x}, \mathbf{x}')\right]\right| > \epsilon\right] \leq c\exp(-c'\epsilon^2 n)$$

for some constants c, c' depending on the size of the hash and the kernel used. In this case the error ϵ arising from ensuring the above inequality, with probability at least $1 - \delta$, for m observations and M classes for a joint feature map $\Phi(\mathbf{x}, y)$, is bounded by

$$\epsilon \leq \sqrt{(2\log(m+1) + 2\log(M+1) - \log\delta + \log c - 2\log 2)/nc'}.$$

For the proof see Appendix A.

2.5 Graphlet Kernels

Denote by G a graph with vertices $V(G)$ and edges $E(G)$. Several methods have been proposed to perform classification on such graphs. Most recently, Przulj (2007) proposed to use the distribution over graphlets, that is, subgraphs, as a characteristic property of the graph. Unfortunately, brute force evaluation does not allow calculation of the statistics for graphlets of size more than 5, since the cost for exact computation scales exponentially in the graphlet size.

In the following we show that sampling and hashing can be used to make the analysis of larger subgraphs tractable in practice. For this denote by $S \subseteq G$ an induced subgraph of G, obtained by restricting ourselves to only $V(S) \subseteq V(G)$ vertices of G and let $\#_S(G)$ be the number of times S occurs in G. This suggests that the feature map $G \to \Phi(G)$, where $\Phi_S(G) = \#_S(G)$ will induce a useful kernel: adding or removing an edge (i, j) only changes the properties of the subgraphs using the pair (i, j) as part of their vertices.

2.5.1 Counting and Sampling

Depending on the application, the distribution over the counts of subgraphs may be significantly skewed. For instance, in sparse graphs we expect the fully disconnected subgraphs to be considerably overrepresented. Likewise, whenever we are dealing with almost complete graphs, the distribution may be skewed towards the other end (*i.e.*, most subgraphs will be complete). To deal with this, we impose weights $\beta(k)$ on subgraphs containing k edges $|E(S)|$.

To deal with the computational complexity issue together with the issue of reweighting the graphs S we simply replace explicit counting with sampling from the distribution

$$\mathbf{P}(S|G) = c(G)\beta(|E(S)|) \tag{2.5}$$

where $c(G)$ is a normalisation constant. Samples from $\mathbf{P}(S|G)$ can be obtained by a Markov-Chain Monte Carlo approach.

Lemma 4 *The following MCMC sampling procedure has the stationary distribution (2.5).*

1. *Choose a random vertex, say $i \in V(S)$ uniformly.*
2. *Add a vertex j from $G \backslash S_i$ to S_i with probability $c(S_i, G)\beta(|E(S_{ij})|)$.*

Here S_i denotes the subgraph obtained by removing vertex i from S, and S_{ij} is the result of adding vertex j to S_i.

Note that sampling over j is easy: all vertices of G which do not share an edge with S_i occur with the same probability. All others depend only on the number of joining edges. This allows for easy computation of the normalisation constant $c(S_i, G)$.

Proof We may encode the sampling rule via

$$T(S_{ij}|S, G) = \frac{1}{k} c(S_i, G)\beta(|E(S_{ij})|)$$

where $c(S_i, G)$ is a suitable normalisation constant. Next we show that T satisfies the detailed balance equations (Gilks et al., 1995) and therefore can be used as a proposal distribution with acceptance probability 1.

$$\frac{T(S_{ij}|S, G)\,\mathbf{P}(S)}{T(S|S_{ij}, G)\,\mathbf{P}(S_{ij})} = \frac{k^{-1}c(S_i, G)\beta(|E(S_{ij})|)c(G)\beta(|E(S)|)}{k^{-1}c(S_{ij,j}, G)\beta(|E(S_{ij,ji})|)c(G)\beta(|E(S_{ij})|)} = 1.$$

This follows since $S_{ij,j} = S_i$ and likewise $S_{ij,ji} = S$. That is, adding and removing the same vertex leaves a graph unchanged. ■

In summary, we obtain an algorithm that will readily draw samples S from $\mathbf{P}(S|G)$ to characterise G.

2.5.2 Dependent Random Variables

The problem with sampling from a MCMC procedure is that the random variables are *dependent* on each other. This means that we cannot simply appeal to Chernoff bounds when it comes to averaging. Before discussing hashing we briefly discuss averages of dependent random variables:

Definition 5 (Bernoulli Mixing) *Denote by Z a stochastic process indexed by $t \in \mathbb{Z}$ with probability measure $\mathbf{P}$ and let Σ_n be the σ-algebra on Z_t with $t \in \mathbb{Z}\backslash 1, \ldots, n-1$. Moreover, denote by $\mathbf{P}_-$ and $\mathbf{P}_+$ the probability measures on the negative and positive indices t respectively. The mixing coefficient β is*

$$\beta(n, \mathbf{P}_X) := \sup_{A \in \Sigma_n} \left| \mathbf{P}(A) - \mathbf{P}_- \times \mathbf{P}_+(A) \right|.$$

If $lim_{n\to\infty}\beta(n, \mathbf{P}_z) = 0$ we say that Z is β-mixing.

That is, $\beta(n, \mathbf{P}_X)$ measures how much dependence a sequence has when cutting out a segment of length n. Nobel and Dembo (1993) show how such mixing processes can be related to iid observations.

Theorem 6 *Assume that $\mathbf{P}$ is β-mixing. Denote by $\mathbf{P}^*$ the product measure obtained from $\ldots \mathbf{P}_t \times \mathbf{P}_{t+1} \ldots$ Moreover, denote by $\Sigma_{l,n}$ the σ-algebra on $Z_n, Z_{2n}, \ldots, Z_{ln}$. Then the following holds:*

$$\sup_{A \in \Sigma_{l,n}} |\mathbf{P}(A) - \mathbf{P}^*(A)| \leq l\beta(n, \mathbf{P}).$$

This allows us to obtain bounds for expectations of variables drawn from $\mathbf{P}$ rather than $\mathbf{P}^*$.

Theorem 7 *Let $\mathbf{P}$ be a distribution over a domain $\mathcal{X}$ and denote by $\Phi : \mathcal{X} \to \mathcal{H}$ a feature map into a Hilbert Space with $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle \in [0, 1]$. Moreover, assume that there is a β-mixing MCMC sampler of $\mathbf{P}$ with distribution $\mathbf{P}^{\mathrm{MC}}$ from which we draw l observations $\mathbf{x}_{in}$ with an interleave of n rather than sampling from $\mathbf{P}$*

directly. Averages with respect to $\mathbf{P}^{\mathrm{MC}}$ *satisfy the following with probability at least* $1-\delta$*:*

$$\left\| \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathbf{P}(\mathbf{x})}[\Phi(\mathbf{x})] - \frac{1}{l}\sum_{i=1}^{l}\Phi(\mathbf{x}_{in}) \right\| \leq l\beta(n,\mathbf{P}^{\mathrm{MC}}) + \frac{2+\sqrt{\log\frac{2}{\delta}}}{\sqrt{l}}.$$

Proof Theorem 6, the bound on $\|\Phi(\mathbf{x})\|$, and the triangle inequality imply that the expectations with respect to $\mathbf{P}^{\mathrm{MC}}$ and $\mathbf{P}^{*}$ only differ by $l\beta$. This establishes the first term of the bound. The second term is given by a uniform convergence result in Hilbert Spaces from Altun and Smola (2006). ■

Hence, sampling from a MCMC sampler for the purpose of approximating inner products is sound, provided that we only take sufficiently independent samples (*i.e.*, a large enough n) into account. The translation of Theorem 7 into bounds on inner products is straightforward, since

$$|\langle \mathbf{x}, y\rangle - \langle \mathbf{x}', y'\rangle| \leq \|\mathbf{x}-\mathbf{x}'\|\,\|y\| + \|y-y'\|\,\|\mathbf{x}\| + \|\mathbf{x}-\mathbf{x}'\|\,\|y-y'\|.$$

2.5.3 Hashing and Subgraph Isomorphism

Sampling from the distribution over subgraphs $S \in G$ has two serious problems in practice which we will address in the following: firstly, there are several graphs which are isomorphic to each other. This needs to be addressed with a graph isomorphism tester, such as Nauty (McKay, 1984). For graphs up to size 12 this is a very effective method. Nauty works by constructing a lookup table to match isomorphic objects.

However, even after the graph isomorphism mapping we are still left with a sizeable number of distinct objects. This is where a hash map on data streams comes in handy. It obviates the need to store any intermediate results, such as the graphs S or their unique representations obtained from Nauty. Finally, we combine the convergence bounds from Theorem 7 with the guarantees available for hash kernels to obtain the approximate graph kernel.

Note that the two randomisations have very different purposes: the sampling over graphlets is done as a way to approximate the *extraction* of features whereas the compression via hashing is carried out to ensure that the representation is computationally efficient.

Data Sets	#Train	#Test	#Labels
RCV1	781,265	23,149	2
DMOZ L2	4,466,703	138,146	575
DMOZ L3	4,460,273	137,924	7,100

Table 2.1: Text data sets. #X denotes the number of observations in X.

2.6 Experiments

To test the efficacy of our approach we applied hashing to the following problems: first we used it for classification on the Reuters RCV1 data set (Lewis et al., 2004) as it has a relatively large feature dimensionality. Secondly, we applied it to the DMOZ ontology (see Section 2.6.2) of topics of webpages* where the number of topics is high. The third experiment—Biochemistry and Bioinformatics Graph Classification uses our hashing scheme, which makes comparing all possible subgraph pairs tractable, to compare graphs (Vishwanathan et al., 2007a). On publicly available data sets like MUTAG and PTC as well as on the biologically inspired data set DD used by Vishwanathan et al. (2007a), our method achieves the best known accuracy.

In both RCV1 and DMOZ, we use linear kernel SVM with stochastic gradient descent (SGD) as the workhorse. We apply our hash kernels and random projection (Achlioptas, 2003) to the SGD linear SVM. We don't apply the approach in Rahimi and Recht (2008) since it requires a shift-invariant kernel $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$, such as a RBF kernel, which is not applicable in this case. In the third experiment, existing randomisation approaches are not applicable since enumerating all possible subgraphs is intractable. Instead we compare hash kernels with existing graph kernels: random walk kernel, shortest path kernel and graphlet kernel (see Borgwardt et al. 2007).

2.6.1 Reuters Articles Categorisation

We use the Reuters RCV1 binary classification data set (Lewis et al., 2004). 781,265 articles are used for training by stochastic gradient descent (SGD) and 23,149 articles are used for testing. Conventionally one would build a bag of words representation first and calculate exact term frequency / inverse document frequency (TF/IDF) counts from the contents of each article as features. The

*DMOZ L2 denotes non-parent topic data in the top 2 levels of the topic tree and DMOZ L3 denotes non-parent topic data in the top 3 levels of the topic tree.

Algorithm	Pre	TrainTest	Error %
BSGD	303.60s	10.38s	6.02
VW	303.60s	87.63s	5.39
VWC	303.60s	5.15s	5.39
HK	0s	25.16s	5.60

Table 2.2: Runtime and error on RCV1. BSGD: Bottou's SGD. VW: Vowpal Wabbit without cache. VWC: Vowpal Wabbit using cache file. HK: hash kernel with feature dimension 2^{20}. Pre: preprocessing time. TrainTest: time to load data, train and test the model. Error: misclassification rate. Apart from the efficacy of the hashing operation itself, the gain in speed is also due to a multi-core implementation—hash kernel uses 4-cores to access the disc for online hash feature generation. For learning and testing evaluation, all algorithms use single-core.

problem is that the TF calculation needs to maintain a very large dictionary throughout the whole process. Moreover, it is impossible to extract features online since the entire vocabulary dictionary is usually unobserved during training. Another disadvantage is that calculating exact IDF requires us to preprocess all articles in a first pass. This is not possible as articles such as news may vary daily.

However, it suffices to compute TF and IDF approximately as follows: using hash features, we no longer require building the bag of words. Every word produces a hash key which is the new dimension index of the word. The frequency is recorded in the dimension index of its hash key. Therefore, every article has a frequency count vector as TF. This TF is a much denser vector which requires no knowledge of the vocabulary. IDF can be approximated by scanning a smaller *part* of the training set.

We compare the hash kernel with Leon Bottou's Stochastic Gradient Descent SVM[†] (BSGD), Vowpal Wabbit (Langford et al., 2007) (VW) and Random Projections (RP) (Achlioptas, 2003). Our hash scheme is generating features online. BSGD is generating features offline and learning them online. VW uses BSGD's preprocessed features and creates further features online. Caching speeds up VW considerably. However, it requires one run of the original VW code for this purpose. RP uses BSGD's preprocessed features and then creates the new projected

[†]Code can be found at `http://leon.bottou.org/projects/sgd`.

Alg.	Dim	Pre	TrainTest	orgTrainSize	newTrainSize	Error %
RP	2^8	748.30s	210.23s	423.29Mb	1393.65Mb	29.35%
	2^9	1079.30s	393.46s	423.29Mb	2862.90Mb	25.08%
	2^{10}	1717.30s	860.95s	423.29Mb	5858.48Mb	19.86%
HK	2^8	0s	22.82s	N/A	N/A	17.00%
	2^9	0s	24.19s	N/A	N/A	12.32%
	2^{10}	0s	24.41s	N/A	N/A	9.93%

Table 2.3: Hash kernel vs. random projections with various feature dimensionalities on RCV1. RP: random projections in Achlioptas (2003). HK: hash kernel. Dim: dimension of the new features. Pre: preprocessing time. TrainTest: time to load data, train and test the model. orgTrainSize: compressed original training feature file size. newTrainSize: compressed new training feature file size. Error: misclassification rate. N/A: not applicable. In hash kernel there is no preprocessing step, so there are no original/new feature files. Features for hash kernel are built up online via accessing the string on disc. The disc access time is taken into account in **TrainTest**. Note that the TrainTest for random projection time increases as the new feature dimension increases, whereas for hash kernel the TrainTest is almost independent of the feature dimensionality.

lower dimension features. Then it uses BSGD for learning and testing. We compare these algorithms on RCV1 in Table 2.2. RP is not included in this table because it would be intractable to run it with the same feature dimensionality as HK for a fair comparison. As can be seen, the preprocessing time of BSGD and VW is considerably longer compared to the time for training and testing, due to the TF-IDF calculation which is carried out offline. For a fair comparison, we measure the time for feature loading, training and testing together. It can also be seen that the speed of online feature generation is considerable compared to disk access. Table 2.2 shows that the test errors for hash kernel, BSGD and VW are competitive.

In table 2.3 we compare hash kernels to RP with different feature dimensions. As we can see, the error reduces as the new feature dimension increases. However, the error of hash kernel is always much smaller (by about 10%) than RP given the same new dimension. An interesting thing is that the new feature file created after applying RP is much bigger than the original one. This is because the projection maps the original sparse feature to a dense feature. For example, when the feature dimensionality is 2^{10}, the compressed new feature file size is already 5.8G. Hash kernels are much more efficient than RP in terms of speed, since to

Dim	#Unique	Collision %	Error %
2^{24}	285614	0.82	5.586
2^{22}	278238	3.38	5.655
2^{20}	251910	12.52	5.594
2^{18}	174776	39.31	5.655
2^{16}	64758	77.51	5.763
2^{14}	16383	94.31	6.096

Table 2.4: Influence of new dimension on Reuters (RCV1) on collision rates (reported for both training and test set combined) and error rates. Note that there is no noticeable performance degradation even for a 40% collision rate.

compute a hash feature one requires only $O(d_{nz})$ hashing operations, where d_{nz} is the number of non-zero entries. To compute a RP feature one requires $O(dn)$ operations, where d is the original feature dimension and and n is the new feature dimension. With RP the new feature is always dense even when n is big, which further increases the learning and testing runtime. When $d_{nz} \ll d$ such as in text processing, the difference is significant. This is verified in our experiment (see Table 2.3). For example, a hash kernel (including **Pre** and **TrainTest**) with 2^{10} feature size is over 100 times faster than RP.

Furthermore, we investigate the influence of the new feature dimension on the misclassification rate. As can be seen in Table 2.4, when the feature dimension decreases, the collision and the error rate increase. In particular, a 2^{24} dimension causes almost no collisions. Nonetheless, a 2^{18} dimension which has almost 40% collisions performs equally well on the same problem. This leads to rather memory-efficient implementations.

2.6.2 DMOZ Websites Multiclass Classification

In a second experiment we perform topic categorisation using the DMOZ topic ontology. The task is to recognise the topic of websites given the short descriptions provided on the webpages. To simplify things we categorise only the leaf nodes (Top two levels: L2 or Top three levels: L3) as a flat classifier (the hierarchy could easily be taken into account by adding hashed features for each part of the path in the tree). This leaves us with 575 leaf topics on L2 and with 7100 leaf topics on L3.

Conventionally, assuming M classes and l features, training M different pa-

	HLF (2^{28})		HLF (2^{24})		HF		no hash	U base	P base
	error	mem	error	mem	error	mem	mem	error	error
L2	30.12	2G	30.71	0.125G	31.28	2.25G (2^{19})	7.85G	99.83	85.05
L3	52.10	2G	53.36	0.125G	51.47	1.73G (2^{15})	96.95G	99.99	86.83

Table 2.5: Misclassification and memory footprint of hashing and baseline methods on DMOZ. HLF: joint hashing of labels and features. HF: hash features only. no hash: direct model (not implemented as it is too large, hence only memory estimates—we have 1,832,704 unique words). U base: baseline of uniform classifier. P base: baseline of majority vote. mem: memory used for the model. Note: the memory footprint in HLF is essentially independent of the number of classes used.

	HLF		KNN			Kmeans		
	2^{28}	2^{24}	S= 3%	6%	9%	S= 3%	6%	9%
L2	69.88	69.29	50.23	52.59	53.81	42.29	42.96	42.76
L3	47.90	46.64	30.93	32.67	33.71	31.63	31.56	31.53

Table 2.6: Accuracy comparison of hashing, KNN and Kmeans. HLF: joint hashing of labels and features. KNN: apply K Nearest Neighbor on sampled training set as search set. Kmeans: apply Kmeans on sampled training set to do clustering and then take its majority class as predicted class. S is the sample size which is the percentage of the entire training set.

rameter vectors w requires $O(Ml)$ storage. This is infeasible for massively multiclass applications. However, by hashing data and labels jointly we are able to obtain an efficient joint representation which makes the implementation computationally possible.

As can be seen in Table 2.5 joint hashing of features and labels is very attractive in items of memory usage and in many cases is necessary to make large multiclass categorisation computationally feasible at all (naive online SVM ran out of memory). In particular, hashing features only produces worse results than joint hashing of labels and features. This is likely due to the increased collision rate: we need to use a smaller feature dimension to store the class dependent weight vectors explicitly.

Next we compare hash kernel with K Nearest Neighbour (KNN) and Kmeans. Running the naive KNN on the entire training set is very slow[‡]. Hence we in-

[‡]In fact the complexity of KNN is $O(N \times T)$, where N and T are the size of the training set

Data	Algorithm	Dim	Pre	TrainTest	Error %
L2	RP	2^7	779.98s	1258.12s	82.06%
	RP	2^8	1496.22s	3121.66s	72.66%
	RP	2^9	2914.85s	8734.25s	62.75%
	HK	2^7	0s	165.13s	62.28%
	HK	2^8	0s	165.63s	55.96%
	HK	2^9	0s	174.83s	50.98%
L3	RP	2^7	794.23s	18054.93s	89.46%
	RP	2^8	1483.71s	38613.51s	84.06%
	RP	2^9	2887.55s	163734.13s	77.25%
	HK	2^7	0s	1573.46s	76.31%
	HK	2^8	0s	1726.67s	71.93%
	HK	2^9	0s	1812.98s	67.18%

Table 2.7: Hash kernel vs. random projections with various feature dimensionalities on DMOZ. RP: random projections in Achlioptas (2003). HK: hash kernel. Dim: dimension of the new features. Pre: preprocessing time—generation of the random projected features. TrainTest: time to load data, train and test the model. Error: misclassification rate. Note that the TrainTest time for random projections increases as the new feature dimension increases, whereas for hash kernel the TrainTest is almost independent of the feature dimensionality. Moving the dimension from 2^8 to 2^9 the increase in processing time of RP is not linear—we suspect this is because with 2^8 the RP model has $256 \times 7100 \times 8$ bytes $\approx$ 14MB, which is small enough to fit in the CPU cache (we are using a 4-cores cpu with a total cache size of 16MB), while with 2^9 the model has nearly 28MB, no longer fitting in the cache.

troduce sampling to KNN. We first sample a subset from the entire training set as search set and then do KNN classification. To match the scheme of KNN, we use sampling in Kmeans too. Again we sample from the entire training set to do clustering. The number of clusters is the minimal number of classes which have at least 90% of the documents. Each test example is assigned to one of the clusters, and we take the majority class of the cluster as the predicted label of the test example. The accuracy plot in Figure 2.1 shows that in both DMOZ L2 and L3, KNN and Kmeans with various sample sizes get test accuracies of 30% to

and the testing set. We estimate the running time for the original KNN, in a batch processing manner ignoring the data loading time, is roughly 44 days on a PC with a 3.2GHz cpu.

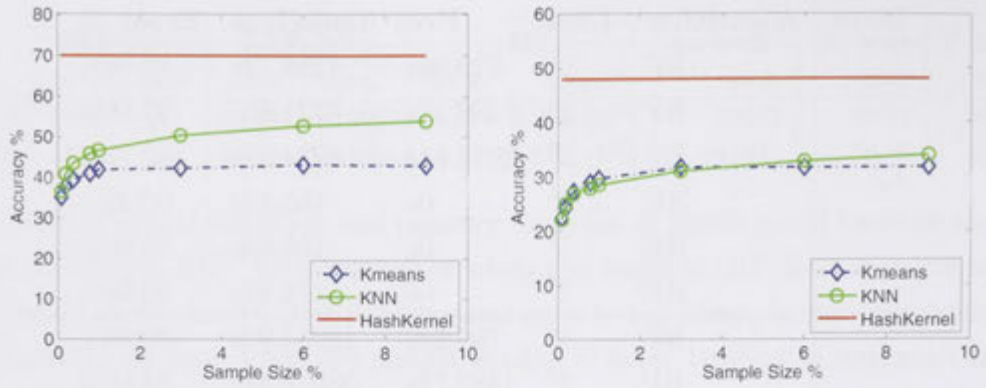


Figure 2.1: Test accuracy comparison of KNN and Kmeans on DMOZ with various sample sizes. Left: results on L2. Right: results on L3. Hash kernel (2^{28}) result is used as an upper bound.

20% less than the upper bound accuracy achieved by hash kernel. The trend of the KNN and Kmeans accuracy curve suggests that the bigger the sample size is, the less accuracy increment can be achieved by increasing it. A numerical result with selected sample sizes is reported in Table 2.6.

We also compare hash kernel with RP with various feature dimensionalities on DMOZ. Here RP generates the random projected feature first and then does online learning and testing. It uses the same 4-cores implementation as hash kernel does. The numerical result with selected dimensionalities is in Table 2.7. It can be seen that hash kernel is not only much faster but also has much smaller error than RP given the same feature dimension. Note that both hash kernel and RP reduce the error as they increase the feature dimension. However, RP can't achieve a competitive error compared to what hash kernel has in Table 2.5, simply because with large feature dimension RP is too slow—the estimated run time for RP with dimension 2^{19} on DMOZ L3 is 2000 days.

Furthermore we investigate whether such a good misclassification rate is obtained by predicting well only on a few dominant topics. We reorder the topic histogram in accordance to ascending error rate. Figure 2.2 shows that hash kernel does very well on the first one hundred topics. They correspond to easy categories such as language related sets "World/Italiano","World/Japanese","World/Deutsch".

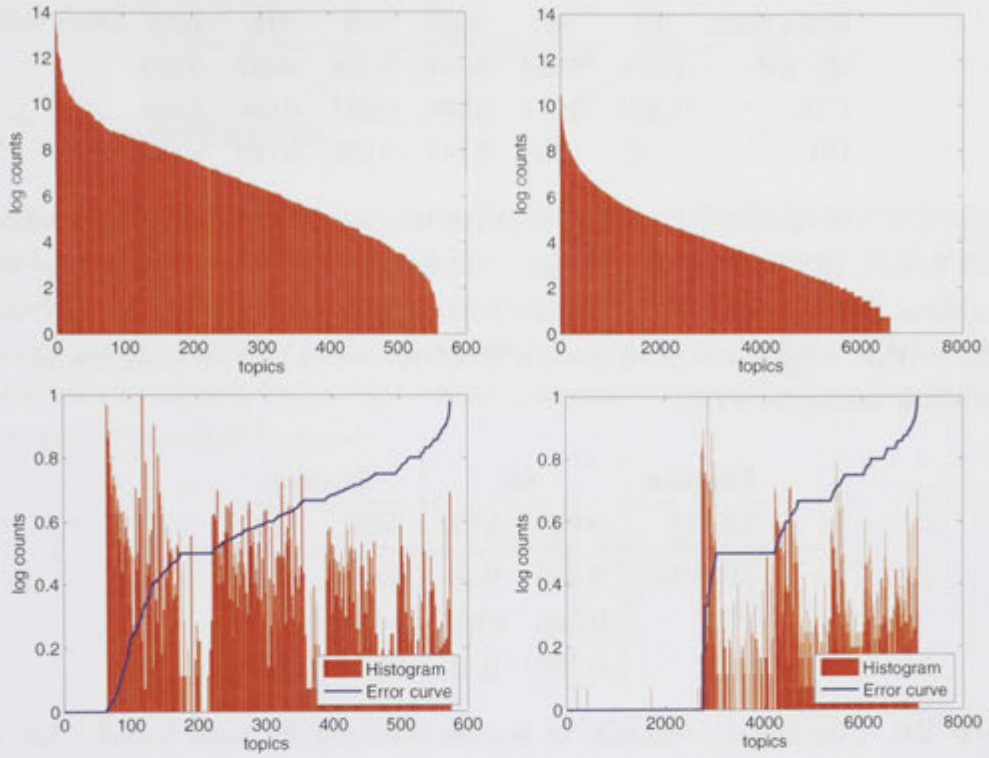


Figure 2.2: Left: results on L2. Right: results on L3. Top: frequency counts for topics as reported on the training set (the test set distribution is virtually identical). We see an exponential decay in counts. Bottom: log-counts and error probabilities on the test set. Note that the error is reasonably evenly distributed among the size of the classes (besides a number of near empty classes which are learned perfectly).

2.6.3 Biochemistry and Bioinformatics Graph Classification

For the final experiment we work with graphs. The benchmark data sets we used here contain three real-world data sets: two molecular compounds data sets, Debnath et al. (1991) and PTC (Toivonen et al., 2003), and a data set for protein function prediction task (DD) from Dobson and Doig (2003). In this work we used the unlabeled version of these graphs, see, for example, Borgwardt et al. (2007).

All these data sets are made of sparse graphs. To capture the structure of the graphs, we sampled connected subgraphs with varying number of nodes, from $n = 4$ to $n = 9$. We used graph isomorphism techniques, implemented in Nauty

Data Sets	RW	SP	GKS	GK	HK	HKF
MUTAG	0.719	0.813	0.819	0.822	0.855	0.865
PTC	0.554	0.554	0.594	0.597	0.606	0.635
DD	>24h	>24h	0.745	>24h	0.799	0.841

Table 2.8: Classification accuracy on graph benchmark data sets. RW: random walk kernel, SP: shortest path kernel, GKS = graphlet kernel sampling 8497 graphlets, GK: graphlet kernel enumerating all graphlets exhaustively, HK: hash kernel, HKF: hash kernel with feature selection. '>24h' means computation did not finish within 24 hours.

Feature	All		Selection	
STATS	ACC	AUC	ACC	AUC
MUTAG	0.855	0.93	0.865	0.912
PTC	0.606	0.627	0.635	0.670
DD	0.799	0.81	0.841	0.918

Table 2.9: Non feature selection vs feature selection for hash kernel. All: all features. Selection: feature selection; ACC: accuracy; AUC: Area under ROC.

(McKay, 1984) to obtain a canonically-labeled isomorph of each sampled subgraph. The feature vector of each example (graph) is composed of the number of times each canonical isomorph was sampled. Each graph was sampled 10000 times for each of $n = 4, 5 \ldots 9$. Note that the number of connected unlabeled graphs grows exponentially with the number of nodes, so the sampling is extremely sparse for large values of n. For this reason we normalised the counts so that for each data set each feature of $\Phi(\mathbf{x})$ satisfies $1 \geq \Phi(\mathbf{x}) \geq 0$.

We compare the proposed hash kernel (with/without feature selection) with random walk kernel, shortest path kernel and graphlet kernel on the benchmark data sets. From Table 2.8 we can see that the hash kernel even without feature selection still significantly outperforms the other three kernels in terms of classification accuracy over all three benchmark data sets.

The dimensionality of the canonical isomorph representation is quite high and many features are extremely sparse, a feature selection step was taken that removed features suspected as non-informative. To this end, each feature was scored by the absolute vale of its correlation with the target. Only features with scores above the median were retained. As can be seen in Table 2.9 feature selection on hash kernel can furthermore improve the test accuracy and area

under ROC.

2.7 Conclusion

In this chapter we showed that hashing is a computationally attractive technique which allows one to approximate kernels for very high dimensional settings efficiently by means of a sparse projection into a lower dimensional space. In particular for multiclass categorisation this makes all the difference in terms of being able to implement problems with thousands of classes in practice on large amounts of data and features.

Chapter 3

Efficient Face Recognition via Hashing

Face recognition often suffers from high dimensionality of the images as well as the large amount of training data. Typically, face images/features are mapped to a much lower dimensional space (*e.g.*, via down-sampling, or linear projection), in which the important information is hopefully preserved. Classification models are then trained on those low-dimensional features. Recently, Wright et al. (2008) propose a random ℓ_1 minimisation approach on sparse representations, which exploits the fact that the sparse representation of the training image index space helps classification and is robust to noise and occlusions. However, the ℓ_1 minimisation in Wright et al. (2008) has computational complexity $O(d^2n^{3/2})$, where d is the number of measurements and n is the size of the training image set. This makes computation expensive for large-scale datasets. Moreover, a large dense random matrix with size of d by n has to be generated beforehand and stored during the entire processing period. We propose hashing to facilitate face recognition, which has complexity of only $O(dn)$. Evaluated on the YaleB dataset (Georghiades et al., 2001), the proposed method is up to 150 times faster than the method in Wright et al. (2008). We further show an efficient way to compute the hashing matrix implicitly, so that the procedure is potentially applicable to online computing, parallel computing and embedded hardware.

In summary, our main contributions include:

- We discover the connection between hashing kernels and compressed sensing. Existing works on hash kernels (Shi et al., 2009a,b; Weinberger et al., 2009) use hashing to perform feature reduction with theoretical guarantees that learning in the reduced feature space gains much computational

power without any noticeable loss of accuracy. The deviation bound and Rademacher margin bound are independent to the line of compressed sensing. Whereas we show the other side of the coin—hashing can actually be viewed as a measurement matrix in compressed sensing, which explains why there is asymptotically no information loss. Also we provide both a theoretical guarantee and empirical evidence that recovering the original signal is possible.

- We apply hashing in the context of compressed sensing to rapid face recognition due to sparse signal recovery. Our experiments show that the proposed method achieves competitive accuracies compared with (if not better than) the state-of-the-art in Wright et al. (2008); Yang et al. (2007). Yet the proposed hashing with orthogonal matching pursuit is much faster (up to 150 times) than Wright et al. (2008); Yang et al. (2007).

- We further present bounds on hashing signal recovery rates and face recognition rates for the proposed algorithms.

We briefly review the related work in Section 3.1, and then introduce two variants of hashing methods for face recognition in Section 3.2. The theoretical analysis in Section 3.3 gives justification to our methods, and experimental results in Section 3.4 demonstrate the excellence of the proposed methods in practice.

3.1 Related work

Given the abundant literature on face recognition, we only review the work closest to ours.

3.1.1 Facial features

Inspired by the seminal work of Eigenface (Turk and Pentland, 1991) using principal component analysis (PCA), learning a meaningful distance metric has been extensively studied for face recognition. These methods try to answer the question that which features of faces are the most informative or discriminative for identifying a face from another. Eigenface using PCA, Fisherface using linear discriminant analysis (LDA), Laplacianface using locality preserving projection (LPP) (He et al., 2005) and nonnegative matrix factorization all belong to this category. These methods project the high-dimensional image data into a low-dimensional feature space. The main justification is that typically the face space

has a much lower dimension than the image space (represented by the number of pixels in an image). The task of recognizing faces can be performed in the lower-dimensional face space. These methods are equivalent to learn a Mahalanobis distance as discussed in Weinberger and Saul (2009). Therefore algorithms such as large-margin nearest neighbor (LMNN) (Weinberger and Saul, 2009) can also be applied. Kernelised subspace methods such as kernel PCA and kernel LDA have also been applied for better performances.

3.1.2 Compressed sensing

Compressive sensing (CS) (Donoho, 2006; Candés et al., 2006) addresses that if a signal can be compressible in the sense that it has a sparse representation in some basis, then the signal can be reconstructed from a limited number of measurements. Several reconstruction approaches have been successfully presented. The typical algorithm in Candés et al. (2006) is to use the so-called ℓ_1 minimisation for an approximation to the ideal non-convex ℓ_0 minimisation. Yang et al. (2007); Wright et al. (2008) apply CS to face recognition, that is, randomly mapping the down-sampled training face images to a low dimensional space and then using ℓ_1 minimisation to reconstruct the sparse representation. The person identity can then be predicted via the minimal residual among all candidates. Unfortunately, ℓ_1 minimisation for large matrices is expensive, which restricts the size of the dataset and the dimensionality of the features.

3.1.3 Hash kernels

Ganchev and Dredze (2008) provide empirical evidence that using hashing can eliminate alphabet storage and reduce the number of parameters without severely deteriorating the performance. In addition, Langford et al. (2007) release the Vowpal Wabbit fast online learning software which uses a hash representation similar to the one discussed here. We propose a hash kernel (Shi et al., 2009a) to deal with the issue of computational efficiency by a very simple algorithm: high-dimensional vectors are compressed by adding up all coordinates which have the same hash value—one only needs to perform as many calculations as there are nonzero terms in the vector. The hash kernel can jointly hash both label and features, thus the memory footprint is essentially independent of the number of classes used. Shi et al. (2009b) further extend this approach to structured data. Weinberger et al. (2009) propose an unbiased hash kernel which is applied to a

large scale application of mass personalised spam filtering.

3.1.4 Connection between hash kernels and compressed sensing

Previous works on hash kernels use hashing to perform feature reduction with a theoretical guarantee that learning in the reduced features space gains much computational power without any noticeable loss of accuracy. The deviation bound and the Rademacher bound show that hash kernels have no information loss asymptotically due to the internal feature redundancy.

Alternatively, we can view hashing as a measurement matrix (see Section 3.3.2) in compressed sensing. We provide both theoretical guarantees in Section 3.3 and empirical results in Section 3.4 to show that recovering the original signal is possible. Thus hash kernels compress the original signal/feature in a recoverable way. This explains why it works well asymptotically in the context of Shi et al. (2009a,b); Weinberger et al. (2009).

3.2 Hashing for face recognition

We show in this section that hashing can be applied to face recognition.

3.2.1 Algorithms

Consider face recognition with n frontal training face images collected from $K \in \mathbb{N}$ subjects. Let n_k denote the number of training images $(\mathbf{x}_i, c_i)$ with $c_i = k$, thus the total number of training images $n = \sum_{k=1}^{K} n_k$. Without loss of generality, we assume that all the data have been sorted according to their labels and then we collect all the vectors in a single matrix $\mathbf{A}$ with m rows and n columns, given by

$$\mathbf{A} = [\mathbf{x}_1, ..., \mathbf{x}_{n_1}, ..., \mathbf{x}_n] \in \mathbb{R}^{m,n}. \tag{3.1}$$

As in Yang et al. (2007); Wright et al. (2008), we assume that any test image lies in the subspace spanned by the training images belonging to the same person. That is for any test image $\mathbf{x}$, without knowing its label information, we assume that there exists $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n)$ such that

$$\mathbf{x} = \mathbf{A}\boldsymbol{\alpha}. \tag{3.2}$$

It is easy to see that if each subject has the same number of images in the dataset, then the α for each subject has maximally $1/K$ portion of nonzero entries. In practice, α is more sparse since often only a small subset of images from the same subjects have nonzero coefficients.

Yang et al. (2007) and Wright et al. (2008) use a random matrix $\mathbf{R} \in \mathbb{R}^{d,m}$ to map $\mathbf{A}\boldsymbol{\alpha}$, where $d \ll m$, and seek $\boldsymbol{\alpha}$ by following ℓ_1 minimisation:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \|\widetilde{\mathbf{x}} - \widetilde{\mathbf{A}}\boldsymbol{\alpha}\|_{\ell_2}^2 + \lambda \|\boldsymbol{\alpha}\|_{\ell_1}, \tag{3.3}$$

where $\widetilde{\mathbf{A}} := \mathbf{R}\mathbf{A}$, $\widetilde{\mathbf{x}} := \mathbf{R}\mathbf{x}$ and λ is the regulariser controlling the sparsity of $\boldsymbol{\alpha}$. However, they did not provide a theoretical result on the reconstruction rate and the face recognition rate. We show both of our algorithms in Section 3.3.

3.2.2 Hashing with ℓ_1

Computing $\mathbf{R}$ directly can be inefficient, therefore we propose hashing to facilitate face recognition (see Figure 3.1). Denote by $h_s(j, d)$ a hash function $h_s : \mathbb{N} \to \{1, \ldots, d\}$ uniformly, where $s \in \{1, \ldots, S\}$ is the seed. Different seeds give different hash functions.

Given $h_s(j, d)$, the hash matrix $\mathbf{H} = (H_{ij})$ is defined as

$$H_{ij} := \begin{cases} 2h_s(j, 2) - 3, & h_s(j, d) = i, \forall s \in \{1, \ldots, S\} \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

Apparently, $H_{ij} \in \{0, \pm 1\}$. Equally likely ± 1 result in an unbiased estimator (see Weinberger et al. (2009)). Let $\boldsymbol{\Phi} := \mathbf{H}\,\mathbf{A} = (\Phi_{ij}) \in \mathbb{R}^{d,n}$. We look for $\boldsymbol{\alpha}$ by

$$\min \|\boldsymbol{\alpha}\|_{\ell_1} \quad \text{subject to} \quad \|\tilde{\mathbf{x}} - \boldsymbol{\Phi}\boldsymbol{\alpha}\|_{\ell_2} \leq \epsilon, \tag{3.5}$$

where $\tilde{\mathbf{x}} = \mathbf{H}\,\mathbf{x}$. Hashing with ℓ_1 is illustrated in Algorithm 1.

3.2.3 Hashing with Orthogonal Matching Pursuit

Tropp and Gilbert (2007) propose Orthogonal Matching Pursuit (OMP) which is faster than ℓ_1 minimisation but requires more measurements than does ℓ_1 for achieving the same precision. Equipped with hashing, OMP (see Algorithm 2) is much faster than random-ℓ_1, random-OMP, and hashing-ℓ_1 without significant loss of accuracy. It is known that OMP has complexity $O(dn)$. Hashing-OMP is faster than random-OMP due to the sparsity of the hash matrix $\mathbf{H}$ (see a sparse $\mathbf{H}$ in Figure 3.2).

Algorithm 1 Hashing-ℓ_1

Input: a image matrix $\mathbf{A}$ for K subjects, a test image $\mathbf{x} \in \mathbb{R}^m$ and an error tolerance ϵ.

Compute $\tilde{\mathbf{x}}$ and $\mathbf{\Phi}$.

Solve the convex optimisation problem

$$\min \|\boldsymbol{\alpha}\|_{\ell_1} \quad \text{subject to} \quad \|\tilde{\mathbf{x}} - \mathbf{\Phi}\boldsymbol{\alpha}\|_{\ell_2} \leq \epsilon. \tag{3.6}$$

Compute the residuals $r_k(\mathbf{x}) = ||\tilde{\mathbf{x}} - \mathbf{\Phi}\alpha^k(\mathbf{x})||_{\ell_2}$ for $k = 1, \ldots, K$, where α^k is the subvector consisting of the components of α corresponding to the basis of class k.

Output: identity $c^* = \operatorname{argmin}_k r_k(\mathbf{x})$.

(a) (b)

Figure 3.1: Demonstration of the recognition procedure of Hashface+ℓ_1. (a) is the test face; (b) is the training faces corresponding to the 10 largest weighted entries in α, the absolute values of their weights are shown on the images in red.

3.2.4 Efficiency of Computation and Memory Usage

For random-ℓ_1, the random matrix $\mathbf{R}$ needs to be computed beforehand and stored throughout the entire routine. When the training set is large or the feature dimensionality is high, computing and storing $\mathbf{R}$ are expensive especially for dense $\mathbf{R}$. We will show now with hashing, $\mathbf{H}$ no longer needs to be computed beforehand explicitly. For example $\mathbf{\Phi}$ and $\tilde{\mathbf{x}}$ can be directly computed as follows without computing $\mathbf{H}$.

$$\begin{aligned} &\forall i = 1, \ldots, d, j = 1, \ldots, n \\ &\Phi_{ij} = \sum_{1 \leq s \leq S} \Big(\sum_{1 \leq t \leq m; h_s(t,d) = i} A_{jt} \xi_{st} \Big), \end{aligned} \tag{3.8}$$

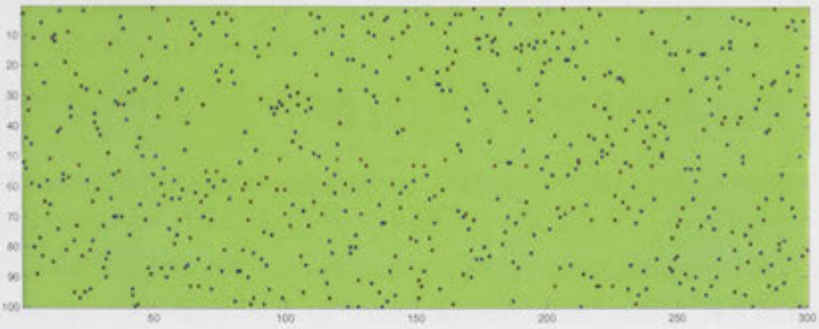


Figure 3.2: Demonstration of a hash matrix. The area with green color means the entry's value is 0, brown indicates value -1 while blue indicates 1. Best viewed in color.

Algorithm 2 Hashing-OMP

Input: a image matrix $\mathbf{A}$ for K subjects, a test image $\mathbf{x} \in \mathbb{R}^m$.

Compute $\tilde{\mathbf{x}}$ and $\mathbf{\Phi}$.

Get $\boldsymbol{\alpha}$ via OMP procedure

$$\boldsymbol{\alpha} = \text{OMP}(\tilde{\mathbf{x}}, \mathbf{\Phi}) \tag{3.7}$$

Compute the residuals $r_k(\mathbf{x}) = ||\tilde{\mathbf{x}} - \mathbf{\Phi}\alpha^k(\mathbf{x})||_{\ell_2}$ for $k = 1, \ldots, K$, where α^k is the subvector consisting of the components of α corresponding to the basis of class k.

Output: identity $c^* = \text{argmin}_k\, r_k(\mathbf{x})$.

where

$$\xi_{st} = \begin{cases} 1, & h_s(t,2) = 2 \\ -1, & \text{otherwise.} \end{cases}$$

$$\forall i = 1, \ldots, d \quad \tilde{x}_i = \sum_{1 \le s \le S} \Big(\sum_{1 \le j \le m; h_s(j,d) = i} y_j \xi_{sj} \Big). \tag{3.9}$$

It means for even very large image set, hashing with OMP (hashing-OMP) can still be implemented on hardware with very limited memory.

3.3 Analysis

In this section, we show that hashing can be used for signal recovery, which is the principle behind the application to face recognition. We further give a lower

bound on its face recognition rate under some mild assumptions.

3.3.1 Restricted isometry property and signal recovery

A n-dimensional real valued signal is called η-sparse if it has at most η many nonzero components. The following Restricted Isometry Property (RIP) (Candes and Tao, 2005; Candés, 2008) provides a guarantee for embedding a high dimensional signal into a lower dimensional space without suffering a great distortion.

Definition 8 (Restricted Isometry Property) *Let Φ be an $m \times n$ matrix and let $\eta < n$ be an integer. Suppose that there exists a constant β such that, for every $m \times \eta$ submatrix Φ_η of Φ and for every vector x,*

$$(1-\epsilon)\|x\|_{\ell_2}^2 \leq \|\Phi_\eta x\|_{\ell_2}^2 \leq (1+\epsilon)\|x\|_{\ell_2}^2. \tag{3.10}$$

Then, the matrix Φ is said to satisfy the η-restricted isometry property with restricted isometry constant ϵ.

Baraniuk et al. (2007) proves that the RIP holds with high probability for some random matrices by the well-known Johnson-Lindenstrauss Lemma (see Baraniuk et al. (2007) for detail). The main difference is that Johnson-Lindenstrauss Lemma concerns finite many points whereas RIP concerns all (infinite many) points. With RIP, it is possible to reconstruct the original sparse signal by randomly combining the entries by the following theorem (Tropp and Gilbert, 2007; Candes and Tao, 2005; Rudelson and Veshynin, 2005).

Theorem 9 (Recovery via Random Map) *For any η-sparse signal $\boldsymbol{\alpha} \in \mathbb{R}^n$ and two constants $z_1, z_2 > 0$, let $m \geq z_1 \eta \log(n/\eta)$, and draw m row vectors $\mathbf{r}_1, \ldots, \mathbf{r}_m$ independently from the standard Gaussian distribution on $\mathbb{R}^n$. Denote the stacked vectors $\{\mathbf{r}_i\}_{i=1}^m$ as the matrix $\mathbf{R} \in \mathbb{R}^{m,n}$ and take m measurements $x_i = \langle \mathbf{r}_i, \boldsymbol{\alpha} \rangle, i = 1, \ldots, m$, i.e., $\mathbf{x} = \mathbf{R}\,\boldsymbol{\alpha}$. Then with probability at least $1 - e^{-z_2 m}$, the signal α can be recovered via*

$$\alpha^* = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{R}\boldsymbol{\alpha}\|_{\ell_2}^2 + \lambda \|\boldsymbol{\alpha}\|_{\ell_1}. \tag{3.11}$$

The condition on m in the theorem above comes from the RIP condition. This immediately leads to following corollary when recovery is on a specific basis $\mathbf{A}$.

Corollary 10 (Recovery on a Specific Basis) *For any η-sparse signal $\boldsymbol{\alpha} \in \mathbb{R}^n$ and two constants $z_1, z_2 > 0$, let $d \geq z_1 \eta \log(n/\eta)$, and draw d row vectors $\mathbf{r}_1, \ldots, \mathbf{r}_d$ independently from the standard Gaussian distribution on $\mathbb{R}^m$. Denote the stacked vectors $\{\mathbf{r}_i\}_{i=1}^d$ as the matrix $\mathbf{R} \in \mathbb{R}^{d,m}$. For any matrix $\mathbf{A} \in \mathbb{R}^{m,n}$ with unit length columns, with probability at least $1 - e^{-z_2 d}$, the signal α can be recovered via*

$$\alpha^* = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \|\mathbf{R}\,\mathbf{x} - (\mathbf{R}\,\mathbf{A})\boldsymbol{\alpha}\|_{\ell_2}^2 + \lambda \|\boldsymbol{\alpha}\|_{\ell_1}. \tag{3.12}$$

For the proof see Appendix A.

3.3.2 Recovery with hashing

Can one reconstruct the signal via hashing rather than Gaussian random mapping? The answer is affirmative. Achlioptas (2003) constructs an embedding with the property that all elements of the projection matrix $\mathbf{U}$ belong in $\{\pm 1, 0\}$ and shows that such an embedding has a Johnson-Lindenstrauss Lemma type of distance preservation property. Due to uniformity, a hashing matrix $\mathbf{H}$ with $S = d$ is such a projection matrix $\mathbf{U}$ ignoring scaling. Since the distance preservation property implies RIP (Baraniuk et al., 2007), signal recovery still holds by replacing the gaussian matrix with $\mathbf{U}$, and it leads to the corollary below.

Corollary 11 (Hashing ℓ_1 Recovery) *For any η-sparse signal $\boldsymbol{\alpha} \in \mathbb{R}^n$ and two constants $z_1, z_2 > 0$ depending on ϵ, given hash matrix $\mathbf{H}$, let $d \geq z_1 \eta \log(n/\eta)$, for any matrix $\mathbf{A} \in \mathbb{R}^{m,n}$, with probability at least $1 - e^{O(-z_2 d)}$, the signal α can be recovered via*

$$\alpha^* = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \|\mathbf{H}\,\mathbf{x} - (\mathbf{H}\,\mathbf{A})\boldsymbol{\alpha}\|_{\ell_2}^2 + \lambda \|\boldsymbol{\alpha}\|_{\ell_1}. \tag{3.13}$$

Here the big O notation is to take scaling into account.

Tropp and Gilbert (2007) show that the OMP recovery theorem holds for all admissible measurement matrices such as Gaussian random matrices and Bernoulli random matrices. Applying OMP to the hashing matrix $\mathbf{H}$, we get the following theorem:

Theorem 12 (Hashing OMP Recovery) *For any η-sparse signal $\boldsymbol{\alpha} \in \mathbb{R}^n$ and confidence $\delta > 0$, given hash matrix $\mathbf{H}$, let $d \geq 16\eta^2 \log(n/\delta)$, for any matrix $\mathbf{A} \in \mathbb{R}^{m,n}$, take the measurements such that $\mathbf{H}\,\mathbf{x} = (\mathbf{H}\,\mathbf{A})\boldsymbol{\alpha}$. Then with probability at least $1 - \delta$, the signal α can be recovered via Algorithm 2.*

For the proof see Appendix A.

3.3.3 Recognition rates

A commonly used assumption is that any test face image can be represented as a weighted sum of face images belonging to the same person, which has been used in Wright et al. (2008); Yang et al. (2007). Ideally, once we achieve the exact weights, the classification should be perfect. However, because the similarity of human face appearance and noise, it is no longer true. So we propose a weakened assumption below.

Assumption 13 *There exists a high dimensional representation in the training face images index space, in which the classification can be conducted with recognition rate at least q.*

The following theorem provides bounds on the recognition rate for any test image via hashing.

Theorem 14 (Recognition Rate via Hashing) *The recognition rates via Algorithm 1 and 2 are, at least $(1 - e^{O(-z_2 d)})q$, and $(1 - \delta)q$, respectively, under Assumption 13.*

Proof We know that with probability at least $1 - e^{O(-z_2 d)}$, the signal can be recovered via Corollary 11. With Assumption 13, we know that even the $e^{O(-z_2 d)}$ portion of not-perfectly-recovered signals are all misclassified, the classification accuracy is still greater than or equal to $(1-e^{O(-z_2 d)})q$. Similarly for Algorithm 2. ■

Note that the bound in the above theorem is possible to further tighten by salvaging the portion of not-perfectly-recovered signals for classification. Indeed, predictions on those signals are usually not completely wrong.

3.4 Experiments

To compare the proposed hashing approaches with random-ℓ_1 (Yang et al., 2007; Wright et al., 2008), we use the same databases, namely, the Extended YaleB and AR as used in Wright et al. (2008). The Extended YaleB database (Georghiades et al., 2001) contains 2,414 frontal-face images from 38 individuals. The cropped and normalised 192×168 face images were captured under various laboratory-controlled lighting conditions. Each subject has 62 to 64 images. Thus we randomly select 32, 15, 15 of them (no repetition) as the training, validation and

testing sets. The AR database consists of over 4,000 front images for 126 individuals. Each individual has 26 images. The pictures of each individual were taken in two different days (Martinez and Benavente, 1998). Unlike Extended YaleB, the faces in AR contain more variations such as illumination change, expressions and facial disguises. 100 subjects (50 male and 50 female) are selected randomly. And for each individual, 13, 7 and 6 images (since there are 26 images in total for each individual) are chosen as training, validation and testing set respectively.

3.4.1 Comparisons on accuracy and efficiency

We run the experiment 10 times on each method and report the average accuracy with the standard deviations (STD) as well as the running time. In each round we run the experiment, the databases are split according to above scheme and different algorithms are performed on the same training, validation and test data set. The number of hash functions L is tuned via model selection assessed on the validation set. Given a feature dimension Dim in the reduced feature space, L is the rounded up integer of $u\times$ Dim. For hashing-ℓ_1 $u \in \{0.02, 0.04, 0.06, ...0.38, 0.40\}$ and for hashing-OMP $u \in \{0.05, 0.10, 0.15, ...0.95, 1.00\}$. The error tolerance ε for random-ℓ_1 is fixed to 0.05 which is identical to the value adopted in Yang et al. (2007).

We evaluate our methods and the state-of-the-arts on the YaleB and AR databases shown in Table 3.1. The best accuracies are highlighted in bold. As we can see, when Dim = 300, hashing-ℓ_1 gets the best accuracies on both datasets. An example is given in Figure 3.3. Figure 3.3 (d) (e) show that the hashing-ℓ_1 weight vector is more sparse than random-ℓ_1. We conjecture that the sparsity is a distinct pattern for classification, which may help to improve the performance as observed in Shi et al. (2009b). Overall, hashing has competitive accuracy with random-ℓ_1.

Hashing-OMP is significantly faster than random-ℓ_1 (from 30 to 150 times shown in Table 3.2). This is further verified in Figure 3.4, which shows that as the feature dimensionality increases, the running time of hashing-OMP is almost constant whereas that of random-ℓ_1 increases dramatically. In real world applications, the speed of the algorithms is a big issue. Hence we further compare hashing-OMP with random-ℓ_1 by restricting their running time to the same level. This way, hashing-OMP gets much better accuracies than random-ℓ_1 shown in Table 3.3. In fact, one may further improve the hashing-OMP accuracy by increasing the feature dimensionality, for Figure 3.4 suggests that the running time

		DIM-50	DIM-100	DIM-200	DIM-300
AR	HASH-OMP	0.658 ± 0.063	0.778 ± 0.066	0.937 ± 0.032	0.969 ± 0.019
	RANDOM-OMP	0.689 ± 0.077	0.784 ± 0.060	0.835 ± 0.036	0.908 ± 0.034
	EIGEN-OMP	0.449 ± 0.131	0.449 ± 0.112	0.606 ± 0.068	0.671 ± 0.040
	HASH-ℓ_1	0.727 ± 0.064	0.915 ± 0.037	0.961 ± 0.029	$\mathbf{0.985 \pm 0.013}$
	RANDOM-ℓ_1	0.855 ± 0.047	0.915 ± 0.042	0.929 ± 0.028	0.958 ± 0.016
	EIGEN-ℓ_1	0.705 ± 0.094	0.751 ± 0.061	0.758 ± 0.035	0.806 ± 0.050
	EIGEN-KNN	0.500 ± 0.102	0.537 ± 0.101	0.555 ± 0.097	0.558 ± 0.096
	FISHER-KNN	0.740 ± 0.045	0.920 ± 0.026	0.977 ± 0.011	0.981 ± 0.011
	EIGEN-SVM	$\mathbf{0.903 \pm 0.048}$	$\mathbf{0.959 \pm 0.021}$	0.976 ± 0.017	0.979 ± 0.011
	FISHER-SVM	0.896 ± 0.043	0.953 ± 0.020	$\mathbf{0.979 \pm 0.013}$	0.980 ± 0.012
YaleB	HASH-OMP	0.806 ± 0.057	0.856 ± 0.050	0.939 ± 0.022	0.964 ± 0.016
	RANDOM-OMP	0.821 ± 0.059	0.908 ± 0.039	0.945 ± 0.033	0.944 ± 0.029
	EIGEN-OMP	0.289 ± 0.075	0.669 ± 0.078	0.882 ± 0.053	0.911 ± 0.048
	HASH-ℓ_1	0.899 ± 0.030	0.951 ± 0.021	0.977 ± 0.017	$\mathbf{0.982 \pm 0.013}$
	RANDOM-ℓ_1	$\mathbf{0.928 \pm 0.036}$	$\mathbf{0.966 \pm 0.018}$	$\mathbf{0.980 \pm 0.017}$	0.979 ± 0.016
	EIGEN-ℓ_1	0.822 ± 0.072	0.911 ± 0.049	0.936 ± 0.037	0.945 ± 0.036
	EIGEN-KNN	0.589 ± 0.101	0.662 ± 0.109	0.702 ± 0.100	0.714 ± 0.096
	FISHER-KNN	0.891 ± 0.050	0.920 ± 0.038	0.948 ± 0.029	0.954 ± 0.030
	EIGEN-SVM	0.890 ± 0.063	0.919 ± 0.041	0.940 ± 0.036	0.953 ± 0.029
	FISHER-SVM	0.880 ± 0.068	0.913 ± 0.040	0.939 ± 0.035	0.948 ± 0.031

Table 3.1: Comparison on accuracy for Hashing-OMP, Random-ℓ_1 and Eigen-ℓ_1 (using Eigenface). On both datasets, Hashing-ℓ_1 achieves the best classification accuracy for $Dim = 300$. When the dimensionality is low, sparse representation based algorithms do not perform as well as SVM.

		DIM-50	DIM-100	DIM-200	DIM-300
AR	HASH-OMP	$\mathbf{11.55 \pm 0.22}$	$\mathbf{24.8 \pm 0.17}$	$\mathbf{78.25 \pm 0.41}$	$\mathbf{101.15 \pm 1.34}$
	RANDOM-OMP	12.05 ± 0.23	80.25 ± 0.93	812.55 ± 0.74	1323.45 ± 2.00
	EIGEN-OMP	12.45 ± 0.24	77.25 ± 0.32	299.55 ± 1.54	422.1 ± 2.03
	HASH-ℓ_1	714.55 ± 2.96	1740.5 ± 12.69	6125.85 ± 99.22	15718.9 ± 290.25
	RANDOM-ℓ_1	814.35 ± 5.44	2276.95 ± 10.28	11266 ± 73.18	31731 ± 292.63
	EIGEN-ℓ_1	751.95 ± 7.10	2637.9 ± 37.68	8758.3 ± 132.26	19632.9 ± 477.55
YaleB	HASH-OMP	$\mathbf{10.05 \pm 0.02}$	$\mathbf{67.4 \pm 0.80}$	$\mathbf{61.45 \pm 0.34}$	$\mathbf{138.05 \pm 0.24}$
	RANDOM-OMP	10.75 ± 0.18	74.3 ± 0.11	944.25 ± 0.53	2944.45 ± 2.90
	EIGEN-OMP	10.8 ± 0.19	75 ± 0.30	190.65 ± 0.49	291.35 ± 0.78
	HASH-ℓ_1	724.45 ± 2.53	1713.3 ± 14.69	5191.9 ± 120.27	9536.8 ± 311.48
	RANDOM-ℓ_1	823.25 ± 5.63	2401 ± 19.56	8655.6 ± 71.23	21887.8 ± 164.97
	EIGEN-ℓ_1	742.55 ± 5.42	2006.6 ± 38.53	4621.65 ± 143.30	8444.65 ± 273.76

Table 3.2: Comparison on the running time(ms) for Hashing-OMP, Random-ℓ_1 and Eigen-ℓ_1. Hashing-OMP is faster than other methods.

RUNTIME(MS)	HASH-OMP	10.05 ± 0.020	46.65 ± 2.394	85.4 ± 3.891	340.95 ± 4.080
	RANDOM-ℓ_1	**N/A**	58.35 ± 1.152	97.15 ± 7.926	329.4 ± 2.480
ACCURACY	HASH-OMP	$\mathbf{0.658 \pm 0.063}$	$\mathbf{0.687 \pm 0.060}$	$\mathbf{0.835 \pm 0.037}$	$\mathbf{0.998 \pm 0.034}$
	RANDOM-ℓ_1	**N/A**	0.0571 ± 0.010	0.2 ± 0.047	0.653 ± 0.068
DIMENSION	HASH-OMP	50	85	180	1000
	RANDOM-ℓ_1	**N/A**	5	10	25

Table 3.3: Comparison on accuracies given running time constraints for Hashing-OMP and Random-ℓ_1 on AR. "Dimension" shows the dimensionality under which the two approaches could achieve similar running speed. "Running time" shows the real running times that should be similar to each other for a certain running speed. N/A means that it was impossible to achieve that speed.

curve for hashing-OMP is almost flat.

3.4.2 Predicting via α

Algorithm 1 uses the residuals to predict the label. Alternatively we can learn a classifier on the sparse α directly. To investigate this, we estimated α via Algorithm 1 (*i.e.*, ℓ_1 minimisation) on the test set and the validation set of the AR dataset. Then we split the union of the two sets into 10 folds. We ran 10 fold cross-validation (8 for training, 1 for testing, and 1 for validation) with SVM. We used both the original α and the normalised one denoted as $\alpha_{[0,1]}$, which

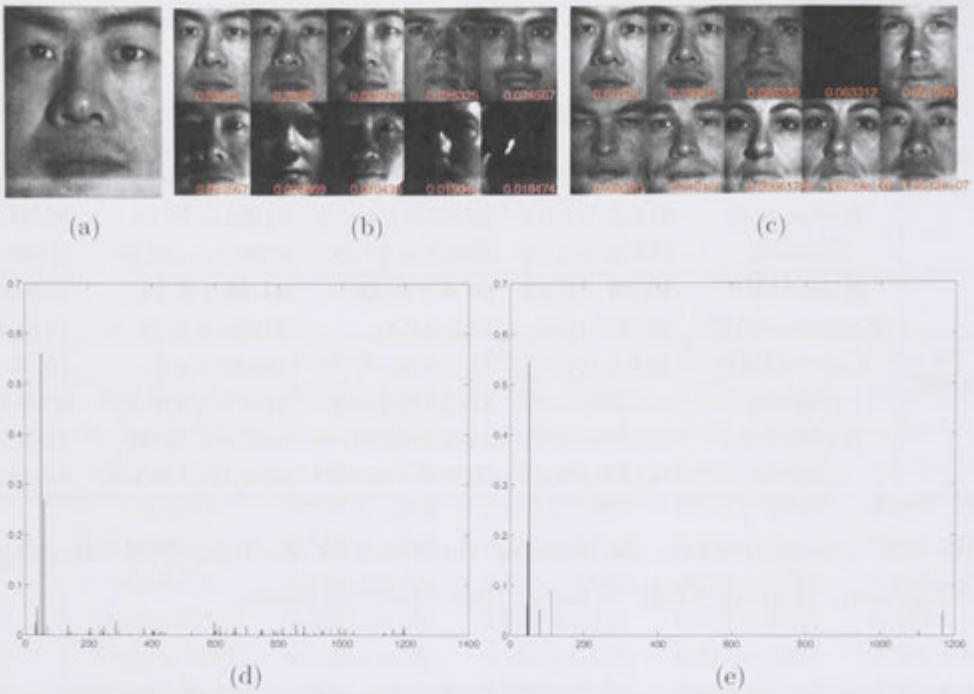


Figure 3.3: The comparison of the recognition procedure of Hashing -ℓ_1 and Random-ℓ_1 on YaleB. (a) is the test face; (b) and (c) are the top 10 weighted training faces for random-ℓ_1 and hashing-ℓ_1 respectively. The absolute value of the weights are shown in red (view in color); (d) and (e) are the bar charts corresponding to the absolute value of top 100 largest weighted entries in the weight α for random-ℓ_1 and hashing-ℓ_1 respectively.

is normalised to $[0, 1]$. Because α has both positive and negative entries, the normalisation step introduces many nonzero entries to $\alpha_{[0,1]}$. As we can see in Table 3.4 and Table 3.1, when Dim $= 50$, SVM on α or $\alpha_{[0,1]}$ gets better results than hashing-OMP and hashing-ℓ_1. When Dim ≥ 100 hashing-OMP and ℓ_1 beat SVM. The experiment suggests that when the feature dimensionality is low (*e.g.* ≤ 50), predicting via α is a good idea; when the feature dimensionality is high, predicting via residuals is better.

3.5 Conclusion

We have proposed a new face recognition methodology with hashing, which speeds up the state-of-the-art in Wright et al. (2008) by up to 150 times, with comparable recognition rates. Both theoretical analysis and experiments justify the excellence

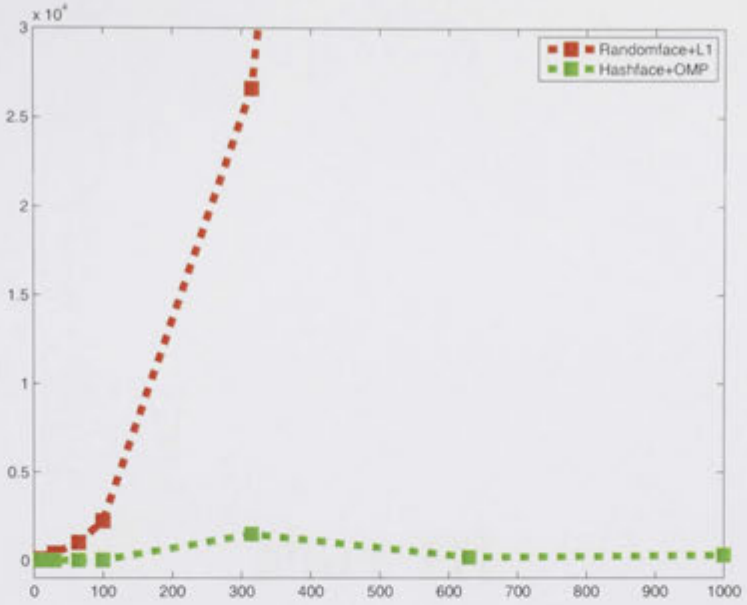


Figure 3.4: The running time curves of Hashing-OMP and Random-ℓ_1 on AR. The horizontal axis represents the dimensionality and the vertical axis is the running time in ms.

	Dim 50	Dim 100	Dim 200	Dim 300
Accuracy on α	0.865 ± 0.006	0.876 ±0.010	0.875 ± 0.007	0.835 ± 0.009
Accuracy on $\alpha_{[0,1]}$	0.853 ± 0.006	0.877 ± 0,011	0.878± 0.007	0.849 ± 0.010

Table 3.4: Test accuracy via predicting on α on AR dataset with 10 fold cross-validation.

of the proposed method.

As hashing can deal with data with structures in the input such as graphs and face images, the next part of the thesis moves on to an even more challenging task — dealing with data with structures in the output.

Part III

Structured Learning in Practice

Part III

Structured Learning In Practice

Chapter 4

Structured Learning Background

In this chapter, we will explain the background of structured learning including some basic notions and some popular existing methods. And the methods here are not our contributions.

4.1 Structured Label

Previous chapters assume that $(\mathbf{x}, y)$ are I.I.D. However, in many cases, $(\mathbf{x}, y)$ are no longer I.I.D. Structured labels are used to deal with these cases. One often models those correlated ys in a structured output $\mathbf{y}$ with the assumption that $(\mathbf{x}, \mathbf{y})$ are I.I.D. drawn from $\mathbf{P}(\mathbf{x}, \mathbf{y})$. Here the output $\mathbf{y}$ can be any object associated with $\mathbf{x}$. For example, for automated paragraph breaking problem, the input $\mathbf{x}$ is a document, and the output $\mathbf{y}$ is a sequence whose the entries denote the beginning positions of the paragraphs. For image segmentation, the input $\mathbf{x}$ is an n by m image, and the output $\mathbf{y}$ is a 2-D lattice $\{\mathbf{y}^{i,j}\}_{1 \le i \le n; 1 \le j \le m}$, where $\mathbf{y}^{i,j}$ denotes class id of the pixel $\mathbf{x}^{i,j}$. The learning is called "structured learning" when some interdependency structure between different parts of the output is exploited. In this case, the output $\mathbf{y}$ is no longer a scalar.

The dependencies within $\mathbf{y}$ are often modelled as directed graphs, undirected graphs and factor graphs. In this thesis, we mainly focus on undirected graphical models for their rich representations of potentials and features — potentials do not need to be normalised locally. And we categorise the most popular structured learning algorithms into two categories — probabilistic approaches and Max Margin approaches. The probabilistic approaches estimate the underlying data distribution, hence require an expensive normalisation step or computing an expectation of features. And the Max Margin approaches estimate a discriminative

function directly, often requiring only an argmax operation which is commonly done via dynamic programming. The two types of approach have their own advantages and disadvantages which we will discuss for each algorithm. It will be seen that many algorithms from both categories can be viewed in a unified framework, Empirical Risk Minimisation (ERM) (Guyon et al., 1992; Shawe-Taylor et al., 1996).

4.2 Empirical Risk Minimisation

Many machine learning algorithms are essentially minimising a regularised empirical risk functional. That is, one would like to solve

$$\min_{\mathbf{w}} J(\mathbf{w}) := \lambda\Omega(\mathbf{w}) + R_{emp}(\mathbf{w}),$$
$$\text{where } R_{emp}(\mathbf{w}) := \frac{1}{m}\sum_{i=1}^{m} l(\mathbf{x}_i, \mathbf{y}_i, \mathbf{w})$$

is the empirical risk and $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m) \in \mathcal{X} \times \mathcal{Y}$ is the training sample of input-output pairs and $\mathbf{w}$ is a parameter vector. The model complexity is controlled by regulariser $\lambda\Omega(\mathbf{w})$ (with $\lambda > 0$), which usually is (piecewise) differentiable and cheap to compute. For instance, let the regulariser $\Omega(\mathbf{w}) = \frac{1}{2}||\,\mathbf{w}\,||^2$, and the loss $\ell(\mathbf{x}_i, y_i, \mathbf{w})$ be the binary hinge loss, $[1 - y\,\langle \mathbf{w}, \mathbf{x}_i \rangle]_+$, we recover the soft margin linear SVM.

Solving the ERM problem, we learn a discriminant function $F \in \mathcal{F} : \mathcal{X} \to \mathcal{Y}$ over input-output pairs from which we can derive a prediction by maximising F over the response variable $\mathbf{y}$ for a given input $\mathbf{x}$. That is,

$$H(\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \mathbf{w}).$$

Throughout the thesis, we assume the problem we are dealing with has structured output $\mathbf{y} \in \mathcal{Y}$, of which the binary class and multiclass are just special cases.

4.3 Probabilistic Approaches

Among probabilistic approaches, two principles are most commonly used — Maximum a Posteriori (MAP) principle and Maximum Entropy (ME) principles. As we shall see, many probabilistic approaches adopt one of the two principles with some additional assumptions and constraints.

4.3.1 Maximum a Posteriori and Maximum Entropy Principles

Maximum a Posteriori A likelihood function $\mathcal{L}(\mathbf{w})$ is the modelled probability or density for the occurrence of a sample configuration $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)$ given the probability density $\mathbf{P}_\mathbf{w}$ parameterised by $\mathbf{w}$. That is,

$$\mathcal{L}(\mathbf{w}) = \mathbf{P}_\mathbf{w}\Big((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\Big).$$

Maximum a Posteriori (MAP) estimates $\mathbf{w}$ by maximising $\mathcal{L}(\mathbf{w})$ times a prior $P(\mathbf{w})$. That is,

$$\mathbf{w}^* = \operatorname*{argmax}_\mathbf{w} \mathcal{L}(\mathbf{w}) P(\mathbf{w}). \tag{4.1}$$

Assuming $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{1 \le i \le m}$ are I.I.D. samples from $\mathbf{P}_\mathbf{w}(\mathbf{x}, \mathbf{y})$, (4.1) becomes

$$\begin{aligned} \mathbf{w}^* &= \operatorname*{argmax}_\mathbf{w} \prod_{1 \le i \le m} \mathbf{P}_\mathbf{w}(\mathbf{x}_i, \mathbf{y}_i) P(\mathbf{w}) \\ &= \operatorname*{argmin}_\mathbf{w} \sum_{1 \le i \le m} -\ln \mathbf{P}_\mathbf{w}(\mathbf{x}_i, \mathbf{y}_i) - \ln P(\mathbf{w}). \end{aligned}$$

Maximum Likelihood (ML) is a special case of MAP when $P(\mathbf{w})$ is uniform. Alternatively, one can replace the joint distribution $\mathbf{P}_\mathbf{w}(\mathbf{x}, \mathbf{y})$ by the conditional distribution $\mathbf{P}_\mathbf{w}(\mathbf{y} \mid \mathbf{x})$ that gives a discriminative model called Conditional Random Fields (CRFs) which will be introduced in Section 4.3.3.

Maximum Entropy ME estimates $\mathbf{w}$ by maximising the entropy. That is,

$$\mathbf{w}^* = \operatorname*{argmax}_\mathbf{w} \sum_{\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}} -\mathbf{P}_\mathbf{w}(\mathbf{x}, \mathbf{y}) \ln \mathbf{P}_\mathbf{w}(\mathbf{x}, \mathbf{y}).$$

It is well-known that the dual of maximum likelihood is maximum entropy (Altun and Smola, 2006), subject to moment matching constraints on the expectations of features taken with respect to the distribution. An example is Maximum Entropy Discrimination Markov Networks (Zhu and Xing, 2009), which belongs to both categories. We will introduce it in Section 4.4.3 after introducing the margin concept.

4.3.2 Generative Markov Models

Generative markov models are usually modelled as directed graphs, thought they can also be modelled as undirected graphs such as Markov Random Fields. The

arrow of an edge points from a parent node to its child node. The joint probability (or density) of a graph with children nodes x_C and parent nodes x_P is then decomposed to $\mathbf{P}_{\mathbf{w}}(x_C|x_P)P\mathbf{w}(x_P)$. The model is estimated via maximising the joint likelihood given observations. The advantage of this method is that the overall probability is always a valid probability since local ones are readily normalised. Incorporating new nodes or variables into existing models can be easily done by simply multiplying the probability of the new variables, for the product of the probabilities is always a valid joint probability as well. Also, for a learnt model, the probability for any subset of variables is readily computed which gives nice interpretation on the importance of each variable. However, the normalised potentials in each node raise a bias problem observed in Lafferty et al. (2001). Furthermore, the potentials and features (which are crucial in structured estimation) are not as rich as those in the undirected graphical models.

4.3.3 Conditional Random Fields

Conditional Random Fields (CRFs) (Lafferty et al., 2001) assume the conditional distribution over $\mathcal{Y} \mid \mathcal{X}$ has a form of exponential families, *i.e.*,

$$\mathbf{P}(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \frac{\exp(\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle)}{Z(\mathbf{w} \mid \mathbf{x})},$$

where

$$Z(\mathbf{w} \mid \mathbf{x}) = \sum_{\mathbf{y}' \in \mathcal{Y}} \exp(\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}') \rangle), \tag{4.2}$$

and

$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_{i \in \mathcal{V}} \Phi_1(\mathbf{x}, \mathbf{y}^{(i)}) + \sum_{(ij) \in \mathcal{E}} \Phi_2(\mathbf{x}, \mathbf{y}^{(ij)}). \tag{4.3}$$

via the Hammersley – Clifford theorem if only node and edge features are considered. More generally speaking, the global feature can be decomposed into local features on cliques (fully connected subgraphs). Denote $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ as $\mathbf{X}$, $(\mathbf{y}_1, \ldots, \mathbf{y}_m)$ as $\mathbf{Y}$. The classical approach is to maximise the conditional likelihood of $\mathbf{Y}$ on $\mathbf{X}$, incorporating a prior on the parameters. This is a Maximum a Posteriori (MAP) estimator, which consists of maximising

$$\mathbf{P}(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) \propto P(\mathbf{w})\, \mathbf{P}(\mathbf{Y} \mid \mathbf{X}; \mathbf{w}).$$

From the i.i.d. assumption we have

$$\mathrm{P}(\mathbf{Y} \mid \mathbf{X}; \mathbf{w}) = \prod_{i=1}^{m} \mathrm{P}(\mathbf{y}_i \mid \mathbf{x}_i; \mathbf{w}),$$

and we impose a Gaussian prior on $\mathbf{w}$

$$P(\mathbf{w}) \propto \exp\left(\frac{-||\,\mathbf{w}\,||^2}{2\sigma^2}\right).$$

Risk Maximising the posterior distribution can also be seen as minimising the negative log-posterior, which becomes our risk function $\ell(\mathbf{w} \mid \mathbf{X}, \mathbf{Y})$

$$\begin{aligned}\ell(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) &= -\ln(P(\mathbf{w})\,\mathrm{P}(\mathbf{Y} \mid \mathbf{X}; \mathbf{w})) + c \\ &= \frac{||\,\mathbf{w}\,||^2}{2\sigma^2} \underbrace{- \sum_{i=1}^{m} (\langle \Phi(\mathbf{x}_i, \mathbf{y}_i), \mathbf{w} \rangle) - \ln(Z(\mathbf{w} \mid \mathbf{x}_i))}_{:=\ell_{_L}} + c,\end{aligned}$$

where c is a constant and $\ell_{_L}$ denotes the negative log-likelihood. Now learning is equivalent to

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \ell(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}).$$

Gradient Above is a convex optimisation problem on $\mathbf{w}$ since $\ln Z(\mathbf{w} \mid \mathbf{x})$ is a convex function of $\mathbf{w}$ (Wainwright and Jordan, 2003). The solution can be obtained by gradient descent since $\ln Z(\mathbf{w} \mid \mathbf{x})$ is also differentiable. We have

$$\nabla_{\mathbf{w}} \ell_{_L}(\mathbf{w} \mid \mathbf{X}, \mathbf{Y}) = -\sum_{i=1}^{m} (\Phi(\mathbf{x}_i, \mathbf{y}_i) - \nabla_{\mathbf{w}} \ln(Z(\mathbf{w} \mid \mathbf{x}_i)).$$

It follows from direct computation that

$$\nabla_{\mathbf{w}} \ln Z(\mathbf{w} \mid \mathbf{x}) = \mathbb{E}_{\mathbf{y} \sim \mathbf{P}(\mathbf{y} \mid \mathbf{x}; \mathbf{w})}[\Phi(\mathbf{x}, \mathbf{y})].$$

Since our sufficient statistics $\Phi(\mathbf{x}, \mathbf{y})$ are decomposed over nodes and edges (eq. 4.3), it is straightforward to show that the expectation also decomposes into expectations on nodes $\mathcal{V}$ and edges $\mathcal{E}$

$$\begin{aligned}&\mathbb{E}_{\mathbf{y} \sim \mathbf{P}(\mathbf{y} \mid \mathbf{x}; \mathbf{w})}[\Phi(\mathbf{x}, \mathbf{y})] = \\ &\sum_{i \in \mathcal{V}} \mathbb{E}_{\mathbf{y}^{(i)} \sim \mathbf{P}(\mathbf{y}^{(i)} \mid \mathbf{x}; \mathbf{w})}[\Phi_1(\mathbf{x}, \mathbf{y}^{(i)})] + \sum_{(ij) \in \mathcal{E}} \mathbb{E}_{\mathbf{y}^{(ij)} \sim \mathbf{P}(\mathbf{y}^{(ij)} \mid \mathbf{x}; \mathbf{w})}[\Phi_2(\mathbf{x}, \mathbf{y}^{(ij)})],\end{aligned}$$

where the node and edge expectations can be computed either exactly by variable elimination or approximately using for example loopy belief propagation. This is the main computational problem with MAP estimation, which in Galleguillos et al. (2008) is circumvented through sampling.

4.4 Max Margin Approaches

4.4.1 Structured Support Vector Machines

Tsochantaridis et al. (2004, 2005) provide a general framework for structured output using maximum margin. They look for a hyperplane that separates the correct labelling $\mathbf{y}_i$ of each observation $\mathbf{x}_i$ in the training set from all the incorrect labellings $\mathcal{Y} - \mathbf{y}_i$ with some margin that depends on the label cost Δ additively*. In order to allow some outliers, they use slack variables ξ_i and maximise the minimum margin, $F(\mathbf{x}_i, \mathbf{y}_i) - \max_{\mathbf{y} \in \mathcal{Y} - \mathbf{y}_i} F(\mathbf{x}_i, \mathbf{y})$, across training instances i. Equivalently,

$$\min_{\mathbf{w}, \xi} \ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i \quad \text{s.t.} \tag{4.4a}$$

$$\forall i, \mathbf{y} \ \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \mathbf{y}) \rangle \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i. \tag{4.4b}$$

To solve this optimisation problem efficiently, one can investigate its dual given by†

$$\begin{aligned} \min_{\alpha} \ & \frac{1}{2} \sum_{i,j,\mathbf{y},\mathbf{y}'} \alpha_{i\mathbf{y}} \alpha_{j\mathbf{y}'} \langle \Phi(\mathbf{x}_i, \mathbf{y}), \Phi(\mathbf{x}_j, \mathbf{y}') \rangle \\ & - \sum_{i,\mathbf{y}} \Delta(\mathbf{y}_i, \mathbf{y}) \alpha_{i\mathbf{y}} \\ \forall i, \mathbf{y} \ & \sum_{\mathbf{y}} \alpha_{i\mathbf{y}} \leq C, \ \alpha_{i\mathbf{y}} \geq 0. \end{aligned} \tag{4.5}$$

Here, there exists one parameter $\alpha_{i\mathbf{y}}$ for each training instance $\mathbf{x}_i$ and its possible labelling $\mathbf{y} \in \mathcal{Y}$. Solving this optimisation problem presents a formidable challenge since $\mathcal{Y}$ generally scales exponentially with the number of variables within each variable $\mathbf{y}$. This essentially makes it impossible to find an optimal solution via enumeration. Instead, one may use a column generation algorithm (see Tsochantaridis et al., 2005) to find an approximate solution in polynomial time. The key idea is to find the most violated constraints (4.4b) for the current set of parameters and satisfy them up to some precision. In order to do this, one needs

*There is an alternative formulation that is multiplicative in Δ. For details see Tsochantaridis et al. (2005).

†Note that one can express the optimisation and estimation problem in terms of kernels $k((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) := \langle \Phi(\mathbf{x}, \mathbf{y}), \Phi(\mathbf{x}', \mathbf{y}') \rangle$. We refer the reader to Tsochantaridis et al. (2005) for details

Algorithm 3 Max-Margin Training Algorithm

Input: data $\mathbf{x}_i$, labels $\mathbf{y}_i$, sample size m, tolerance ϵ
Initialise $S_i = \emptyset$ for all i, and $\mathbf{w} = 0$.
repeat
  for $i = 1$ **to** m **do**
    $\mathbf{w} = \sum_i \sum_{y \in S_i} \alpha_{i\mathbf{y}} \Phi(\mathbf{x}_i, \mathbf{y})$
    $\overline{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}) \rangle + \Delta(\mathbf{y}_i, \mathbf{y})$
    $\xi = [\max_{\mathbf{y} \in S_i} \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}) \rangle]_+ + \Delta(\mathbf{y}_i, \mathbf{y}))$
    if $\langle \mathbf{w}, \Phi(\mathbf{x}_i, \overline{\mathbf{y}}) \rangle + \Delta(\mathbf{y}_i, \mathbf{y}) > \xi + \epsilon$ **then**
      Increase constraint set $S_i \leftarrow S_i \cup \overline{\mathbf{y}}$
      Optimise (4.5) wrt $\alpha_{i\mathbf{y}}, \forall \mathbf{y} \in S_i$.
    end if
  end for
until S has not changed in this iteration

to find

$$\overline{\mathbf{y}_i} = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \Delta(\mathbf{y}_i, \mathbf{y}) + \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}) \rangle , \tag{4.6}$$

which can usually be done via dynamic programming due to the decomposition of Δ and Φ such as in (4.3). For the training procedure see Algorithm 3.

4.4.2 Max Margin Markov Network

Max Margin Markov Network (M3N) (Taskar et al., 2004) essentially shares the same primal (4.4) and dual (4.5) with structured SVMs except using Linear Programming (LP) for inference to find the most-violated $\overline{\mathbf{y}_i}$ in (4.6). The dual formula in (4.5) can be transformed into (see Taskar, 2004, Chapter 5)

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2} \| \sum_{i,\mathbf{y}} \alpha_{i\mathbf{y}} [\Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \mathbf{y})] \|^2 + \sum_{i,\mathbf{y}} \Delta(\mathbf{y}_i, \mathbf{y}) \alpha_{i\mathbf{y}} \\ \forall i, \mathbf{y} \quad & \sum_{\mathbf{y}} \alpha_{i\mathbf{y}} = C, \ \alpha_{i\mathbf{y}} \geq 0. \end{aligned}$$

Taskar et al. (2004) discover that the dual variable $\frac{\alpha_{i\mathbf{y}}}{C}$ can be viewed as a distribution over $\mathbf{y}$ given $\mathbf{x}$. Thus the dual object becomes

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2} \| \sum_{i} \mathbb{E}_{\mathbf{y} \sim \alpha_{i\mathbf{y}}} [\Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \mathbf{y})] \|^2 + \sum_{i} \mathbb{E}_{\mathbf{y} \sim \alpha_{i\mathbf{y}}} \Delta(\mathbf{y}_i, \mathbf{y}) \\ \forall i, \mathbf{y} \quad & \sum_{\mathbf{y}} \frac{\alpha_{i\mathbf{y}}}{C} = 1, \ \alpha_{i\mathbf{y}} \geq 0. \end{aligned} \tag{4.7}$$

Denote $\mathbf{y} \sim \mathbf{y}^{(a)}$ as the value of the component $\mathbf{y}^{(a)}$ is consistent with that in $\mathbf{y}$. Decomposing global features into local node and edge features as (4.3), we get

$$\begin{aligned}
&\mathbb{E}_{\mathbf{y}\sim\alpha_{i\mathbf{y}}}\,\Phi(\mathbf{x}_i,\mathbf{y}) \\
&= \sum_{\mathbf{y}} \alpha_{i\mathbf{y}} \Phi(\mathbf{x}_i,\mathbf{y}) \\
&= \sum_{\mathbf{y}} \alpha_{i\mathbf{y}} \sum_{a\in\mathcal{V}} \Phi_1(\mathbf{x}_i,\mathbf{y}^{(a)}) + \sum_{(ab)\in\mathcal{E}} \Phi_2(\mathbf{x}_i,\mathbf{y}^{(ab)}) \\
&= \sum_{a\in\mathcal{V}} \sum_{\mathbf{y}:\mathbf{y}\sim\mathbf{y}^{(a)}} \alpha_{i\mathbf{y}}(\mathbf{y}) \Phi_1(\mathbf{x}_i,\mathbf{y}^{(a)}) \\
&\quad + \sum_{(ab)\in\mathcal{E}} \sum_{\mathbf{y}:\mathbf{y}\sim\mathbf{y}^{(ab)}} \alpha_{i\mathbf{y}}(\mathbf{y}) \Phi_2(\mathbf{x}_i,\mathbf{y}^{(ab)}) \\
&= \sum_{a\in\mathcal{V}} \sum_{\mathbf{y}^{(a)}} \mu_{\mathbf{x}_i}(\mathbf{y}^{(a)}) \Phi_1(\mathbf{x}_i,\mathbf{y}^{(a)}) \\
&\quad + \sum_{(ab)\in\mathcal{E}} \sum_{\mathbf{y}^{(ab)}} \mu_{\mathbf{x}_i}(\mathbf{y}^{(ab)}) \Phi_2(\mathbf{x}_i,\mathbf{y}^{(ab)}),
\end{aligned}$$

where marginals

$$\begin{aligned}
\mu_{\mathbf{x}_i}(\mathbf{y}^{(a)}) &= \sum_{\mathbf{y}:\mathbf{y}\sim\mathbf{y}^{(a)}} \alpha_{i\mathbf{y}}(\mathbf{y}) \\
\mu_{\mathbf{x}_i}(\mathbf{y}^{(ab)}) &= \sum_{\mathbf{y}:\mathbf{y}\sim\mathbf{y}^{(ab)}} \alpha_{i\mathbf{y}}(\mathbf{y}).
\end{aligned}$$

Similarly if $\Delta(\mathbf{y}_i,\mathbf{y}) = \sum_{a\in\mathcal{V}} \Delta(\mathbf{y}_i,\mathbf{y}^{(a)})$, then

$$\mathbb{E}_{\mathbf{y}\sim\alpha_{i\mathbf{y}}}\,\Delta(\mathbf{y}_i,\mathbf{y}) = \sum_{a\in\mathcal{V}} \mu_{\mathbf{x}_i}(\mathbf{y}^{(a)}) \Delta(\mathbf{y}_i,\mathbf{y}^{(a)}).$$

Thus we only need to know the marginals $\mu_{\mathbf{x}_i}(\mathbf{y}^{(a)}), \mu_{\mathbf{x}_i}(\mathbf{y}^{(ab)})$ over nodes and edges to compute the dual in (4.7) instead of the entire joint distribution $\alpha_{i\mathbf{y}}$. To ensure the marginals resulting from a valid distribution $\alpha_{i\mathbf{y}}(\mathbf{y})$, one must ensure following consistency constraint

$$\sum_{\mathbf{y}^{(b)}} \mu_{\mathbf{x}_i}(\mathbf{y}^{(ab)}) = \mu_{\mathbf{x}_i}(\mathbf{y}^{(a)}), \forall (a,b) \sim \mathcal{E}, \forall i.$$

For graphical models with higher order features, higher order consistency are required.

The inference can also be done by marginals of $P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x})$ over nodes and edges (see Sontag et al., 2008). Again consistency constraints are needed. This way

of marginalization was originally proposed for CRFs; however, it is applicable to M3N as well. The objectives in both CRFs and M3N become a LP problem.

The major drawback for M3N or more generally any LP based approach, is that existing LP solvers (even commercial ones) can not handle millions of variables.

4.4.3 Maximum Entropy Discrimination Markov Networks

Jaakkola et al. (2000) propose a Maximum Entropy Discrimination (MED) scheme that maximises the entropy — or minimises the KL divergence $\mathrm{KL}(Q(\mathbf{w})||P(\mathbf{w})) = \int \ln \frac{Q(\mathbf{w})}{P(\mathbf{w})} dQ(\mathbf{w})$ between the posterior Q and the prior P — with a constraint that the expected margin with respect to the posterior $Q(\mathbf{w})$ over model parameter $\mathbf{w}$ is not less than certain threshold (that is a weighted max margin constraint or weighted hinge loss via the posterior) for binary classification. Zhu and Xing (2009) later extend it to the structured case, called Maximum Entropy Discrimination Markov Networks (MEDN) as follows:

$$\min_{\mathbf{w},\xi} \ \mathrm{KL}(Q(\mathbf{w})||P(\mathbf{w})) + C\sum_{i=1}^{m}\xi_i \quad \text{s.t.}$$
$$\forall i, \mathbf{y} \int \Big[\langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \mathbf{y}) \rangle - \Delta(\mathbf{y}_i, \mathbf{y}) \Big] dQ(\mathbf{w}) \geq -\xi_i.$$

Again $\mathbf{y}$ can be replaced by the most-violated $\overline{\mathbf{y}}_i$. Apparently letting $\mathbf{y}$ be scalar y, MEDN recovers MED. Zhu and Xing (2009) show that letting $P(\mathbf{w})$ be a zero mean, identity variance gaussian over $\mathbf{w}$, MEDN recovers M3N.

4.5 Conclusion

Here we have categorised the most popular structured learning algorithms into probabilistic approaches and Max Margin approaches. Two types of approaches have their own advantages and disadvantages as we discussed. In fact, many algorithms from both categories can be viewed in a unified framework, Empirical Risk Minimisation.

Chapter 5

Automatic Paragraph Segmentation

Automatic paragraph segmentation (APS) is closely related to some well known problems such as text segmentation, discourse parsing, topic shift detection and is relevant for various important applications in speech-to-text and text-to-text tasks.

In speech-to-text applications, the output of a speech recognition system, such as the output of systems creating memos and documents for the Parliament House, is usually raw text without any punctuation or paragraph breaks. Clearly, such text requires paragraph segmentations. In text-to-text processing, such as summarisation, the output text does not necessarily retain the correct paragraph structure and may require post-processing. There is psycholinguistic evidence as cited by Sporleder and Lapata (2006) showing that insertion of paragraph breaks improves readability. Moreover, it has been shown that different languages may have cross-linguistic variations in paragraph boundary placement (Zhu, 1999), which indicates that machine translation can also benefit from APS. APS can also recover the paragraph breaks that are often lost in OCR applications.

There has been growing interest within the NLP community for APS in recent years. Previous methods such as Sporleder and Lapata (2006); Genzel (2005) treat the problem as a binary classification task, where each sentence is labeled as the beginning of a paragraph or not. They focus on the use of features of the sentence itself, such as surface features, language modelling features and syntactic features. The effectiveness of features is investigated across languages and/or domains. However, these approaches ignore the inherent sequential nature of APS. Clearly, consecutive sentences within the same paragraph depend on each other.

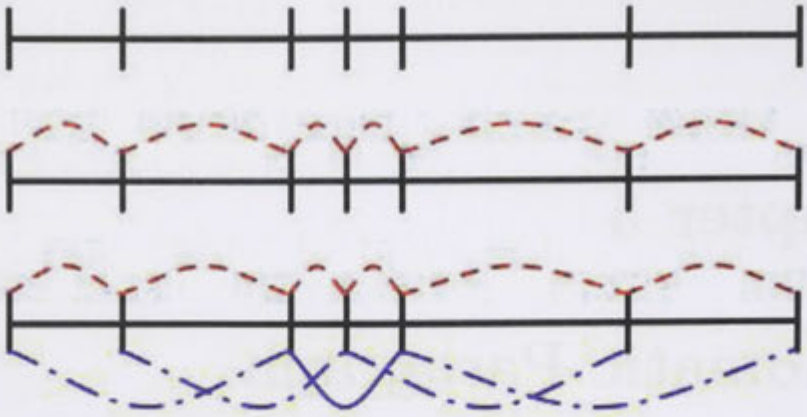


Figure 5.1: **Top**: sequence (horizontal line) with segment boundaries (vertical lines). This corresponds to a model where we estimate each segment boundary independently of all other boundaries. **Middle**: simple semi-Markov structure. The position of the segment boundaries only depends on the position of its neighbours, as denoted by the (red) dash arcs. **Bottom**: a more sophisticated semi-Markov structure, where each boundary depends on the position of two of its neighbours. This may occur, e.g., when the decision of where to place a boundary depends on the content of two adjacent segments. The longer range interaction is represented by the additional (blue) arcs.

Moreover, paragraphs should exhibit certain properties such as coherence, which should be explored within an APS system. One cannot incorporate such properties/features when APS is treated as a binary classification problem. To overcome this limitation, we cast APS as a sequence prediction problem, where the performance can be significantly improved by optimising the choice of labelling over a whole sequence of sentences, rather than individual sentences.

Sequence prediction is one of the most prominent examples of structured prediction. This problem is generally formalised such that there exists one variable for each observation in the sequence and the variables form a Markov chain such as a Hidden Markov Model (HMM). Segmentation of a sequence has been studied as a class of sequence prediction problems with common applications such as protein secondary structure prediction, Named Entity Recognition and segmentation of FAQ's. The exceptions to this approach are Sarawagi and Cohen (2004); Raetsch and Sonnenburg (2006), which show that Semi-Markov models (SMMs) (Janssen and Limnios, 1999), which are a variation of Markov models, are a natural formulation for sequence segmentation. The advantage of these

models, depicted in Figure 5.1, is their ability to encode features that capture properties of a segment as a whole, which is not possible in an HMM model. In particular, these features can encode similarities between two sequence segments of arbitrary lengths, which can be very useful in tasks such as APS.

In this chapter, we present a Semi-Markov model for APS and propose a max-margin training procedure on these methods. This training method is a generalisation of the Max-Margin methods for Hidden Markov Models (HMMs) (Altun et al., 2003) to SMMs. It follows the recent literature on discriminative learning of *structured prediction* (Lafferty et al., 2001; Collins, 2002; Taskar et al., 2004). Our method inherits the advantages of discriminative techniques, namely the ability to encode arbitrary (overlapping) features and not making implausible conditional independence assumptions. It also has advantages of SMM models, namely the ability to encode features at segment level. We present a linear time inference algorithm for SMMs and outline the learning method. Experimental evaluation on datasets used previously on this task (Sporleder and Lapata, 2006) shows improvement over the state-of-the art methods on APS.

5.1 Modelling Sequence Segmentation

In sequence segmentation, our goal is to solve the estimation problem of finding a segmentation $\mathbf{y} \in \mathcal{Y}$, given an observation sequence $\mathbf{x} \in \mathcal{X}$. For example, in APS $\mathbf{x}$ can be a book which is a sequence of sentences. In a Semi-Markov model, there exists one variable for each subsequence of observations (i. e. multiple observations) and these variables form a Markov chain. This is opposed to an HMM where there exists one variable for each observation. More formally, in SMMs, $\mathbf{y} \in \mathcal{Y}$ is a sequence of segment labellings $s_i = (b_i, l_i)$ where b_i is a non-negative integer denoting the beginning of the i^{th} segment which ends at position $b_{i+1} - 1$ and whose label is given by l_i (Sarawagi and Cohen, 2004). Since in APS the label of the segments is irrelevant, we represent each segment simply by the beginning position $\mathbf{y} := \{b_i\}_{i=0}^{L-1}$ with the convention that $b_0 = 0$ and $b_L = N$ where N is the number of observations in $\mathbf{x}$. Here, L denotes the number of segments in $\mathbf{y}$. So the first segment is $[0, b_1)$, and the last segment is $[b_{L-1}, N)$, where $[a, b)$ denotes all the sentences from a to b including a but excluding b.

We cast this estimation problem as finding a discriminant function $F(\mathbf{x}, \mathbf{y})$ such that for an observation sequence $\mathbf{x}$ we assign the segmentation that receives

the best score with respect to F,

$$\mathbf{y}^*(\mathbf{x}) := \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y}). \tag{5.1}$$

As in many learning methods, we consider functions that are linear in some feature representation Φ,

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle. \tag{5.2}$$

Here, $\Phi(\mathbf{x}, \mathbf{y})$ is a feature map defined over the joint input/output space as detailed in Section 5.3.

5.1.1 Max-Margin Training

We now present a maximum margin training procedure for predicting structured output variables, of which sequence segmentation is an instance. One of the advantages of this method is its ability to incorporate the cost function that the classifier is evaluated with. Let $\Delta(\mathbf{y}, \bar{\mathbf{y}})$ be the cost of predicting $\bar{\mathbf{y}}$ instead of $\mathbf{y}$. For instance, Δ is usually the 0-1 loss for binary and multiclass classification. However, in segmentation, this may be a more sophisticated function such as the symmetric difference of $\mathbf{y}$ and $\bar{\mathbf{y}}$ as discussed in Section 5.2. Then, one can argue that optimising a loss function that incorporates this cost can lead to better generalisation properties*.

We follow the general framework of Tsochantaridis et al. (2004) and look for a hyperplane that separates the correct labelling $\mathbf{y}_i$ of each observation sequence $\mathbf{x}_i$ in our training set from all the incorrect labellings $\mathcal{Y} - \mathbf{y}_i$ with some margin that depends additively on Δ as in (4.4)†. In order to allow for some outliers, we use slack variables ξ_i and maximise the minimum margin, $F(\mathbf{x}_i, \mathbf{y}_i) - \max_{\mathbf{y} \in \mathcal{Y} - \mathbf{y}_i} F(\mathbf{x}_i, \mathbf{y})$, across training instances i. For the dual form see (4.5). In order to find the most violated constraint in (4.4b), we propose an extension of the Viterbi algorithm in Section 5.4 for Semi Markov models.

To adapt the Structured SVMs framework to the segmentation setting, we need to address three issues: a) we need to specify a loss function Δ for segmentation, b) we need a suitable feature map Φ as defined in Section 5.3, and c) we need to find an algorithm to solve (4.6) efficiently. The max-margin training of SMMs was also presented in Raetsch and Sonnenburg (2006)

*For a theoretical analysis of this approach see Tsochantaridis et al. (2004).

†There is an alternative formulation that is multiplicative in Δ. We prefer 4.4 due to computational issues.

5.2 Cost Function

To measure the discrepancy between $\mathbf{y}$ and some alternative sequence segmentation $\mathbf{y}'$, we simply count the number of segment boundaries that have a) been missed and b) been wrongly added. Note that this definition allows for errors exceeding 100% - for instance, if we were to place considerably more boundaries than can actually be found in a sequence.

The number of errors is given by the symmetric difference between $\mathbf{y}$ and $\mathbf{y}'$, when segmentations are viewed as sets. This can be written as

$$\begin{aligned}\Delta(\mathbf{y}, \mathbf{y}') &= |\,\mathbf{y}\,| + |\,\mathbf{y}'\,| - 2|\,\mathbf{y} \cap \mathbf{y}'\,| \\ &= |\,\mathbf{y}\,| + \sum_{i=1}^{l'} \left[1 - 2\,\{b'_i \in \mathbf{y}\}\right]. \end{aligned} \tag{5.3}$$

Here $|\cdot|$ denotes the cardinality of the set. (5.3) plays a vital role in solving (4.6), since it allows us to decompose the loss in $\mathbf{y}'$ into a constant and functions depending on the segment boundaries b'_i only. Note that in the case where we want to segment *and* label, we simply would need to check that the positions are accurate *and* that the labels of the segments match.

5.3 Feature Representation

SMMs can extract three kinds of features from the input/output pairs: a) node features, i. e. features that encode interactions between attributes of the observation sequence and the (label of a) *segment* (rather than the label of each observation as in HMMs), b) features that encode interactions between neighbouring labels along the sequence and c) edge features, *i.e.*, features that encode properties of segments. The first two types of features are commonly used in other sequence models, such as HMMs and CRFs. The third feature type is specific to Semi-Markov models. In particular, these features can encode properties of a whole segment or similarities between two sequence segments of arbitrary lengths. The cost of this expressibility is simply a constant factor of the complexity of Markov models, if the maximum length of a segment is bounded. These types of features are particularly useful in the face of sparse data.

As in HMMs, we assume stationarity in our model and sum over the features of each segment to get $\Phi(\mathbf{x}, \mathbf{y})$. Then, Φ corresponding to models of the middle

structure given in Figure 5.1 are given by

$$\Phi(\mathbf{x}, \bar{\mathbf{y}}) := (\Phi_0, \sum_{i=1}^{\bar{l}-1} \Phi_1(\bar{n}_i, \mathbf{x}), \sum_{i=1}^{\bar{l}} \Phi_2(\bar{b}_{i-1}, \bar{b}_i, \mathbf{x})).$$

We let $\Phi_0 = \bar{l} - 1$, the number of segments. The node features Φ_1 capture the dependency of the current segment boundary to the observations, whereas the edge features Φ_2 represent the dependency of the current segment to the observations. To model the bottom structure in Figure 5.1, one can design features that represent the dependency of the current segment to its adjacent segments as well as the observations, $\Phi_3(\mathbf{x}, b_{i-2}, b_{i-1}, b_i)$. The specific choices of the feature map Φ are presented in Section 5.5.

5.4 Column Generation on SMMs

Tractability of Algorithm 3 depends on the existence of an efficient algorithm that finds the most violated constraint (4.4b) via (4.6). Both the cost function of Section 5.2 and the feature representation of Section 5.3 are defined over a short sequence of segment boundaries. Therefore, using the Markovian property, one can perform the above maximisation step efficiently via a dynamic programming algorithm. This is a simple extension of the Viterbi algorithm. The inference given by (5.1) can be performed using the same algorithm, setting Δ to a constant function.

We first state the dynamic programming recursion for $F + \Delta$ in its generality. We then give the pseudocode for $\Phi_3 = \emptyset$.

Denote by $T(t_-, t_+; x)$ the largest value of $\Delta(\mathbf{y}, p) + F(\mathbf{x}, p)$ for any *partial* segmentation p that starts at position 0 and which ends with the segment $[t_-, t_+)$. Moreover, let M be a upper bound on the length of a segment. The recursive step of the dynamic program is given by

$$T(t_-, t_+; \mathbf{x}) = \max_{\max(0, t_- - M) \leq k < t_-} T(k, t_-; \mathbf{x}) + g(k, t_-, t_+)$$

where we defined the increment $g(k, t_-, t_+)$ as

$$\langle \Phi_0(\mathbf{x}), \Phi_1(\mathbf{x}, t_+), \Phi_2(\mathbf{x}, t_-, t_+), \Phi_3(\mathbf{x}, k, t_-, t_+), \mathbf{w} \rangle + 1 - 2\{(t_-, t_+) \in \mathbf{y}\}$$

where by convention $T(i, i') = -\infty$ if $i < 0$ for all labels. Since T needs to be computed for all values of $t_+ - M \leq t_- < t_+$, we need to compute $O(|\mathbf{x}|M)$ values, each of which requires an optimisation over M possible values. That is, storage

Algorithm 4 Column Generation

Input: sequence $\mathbf{x}$, segmentation $\mathbf{y}$, max-length of a segment M
Output: score s, segment boundaries $\mathbf{y}'$
Initialise vectors $T \in \mathbb{R}^m$ and $R \in \mathcal{Y}^m$ to 0
for $i = 1$ **to** l **do**
  $R_i = \underset{\max(0,i-M)\leq j<i}{\operatorname{argmax}} \; T_j + g(j,i)$
  $T_i = T_{R_i} + g(R_i, i)$
end for
$s = T_m + |\,\mathbf{y}\,|$
$\mathbf{y}' = \{m\}$
repeat
  $i = \mathbf{y}'_{\text{first}}$
  $\mathbf{y}' \leftarrow \{R_i, \mathbf{y}'\}$
until $i = 0$

requirements are $O(|\,\mathbf{x}\,|M)$, whereas the computation scales with $O(|\,\mathbf{x}\,|M^2)$. If we have a good bound on the maximal sequence length, this can be dealt with efficiently. Finally, the recursion is set up by $T(0, 0, \mathbf{x}) = |\,\mathbf{y}\,|$.

See Algorithm 4 for a pseudocode, when $\Phi_3 = \emptyset$. The segmentation corresponding to (4.6) is found by constructing the path traversed by the argument of the max operation generating T.

5.5 Local Features

We now specify the features described in Section 5.3 for APS. Note that the second type of features do not exist for APS since we ignore the labellings of segments.

Node Features Φ_1

Node features $\Phi_1(b_j, \mathbf{x})$ represent the information of the current segment boundary and some attributes of the observations around it (which we define as the current, preceding and successive sentences). These are sentence level features, which we adapt from Genzel (2005) and Sporleder and Lapata (2006)[‡]. For the b_jth sentence, $\mathbf{x}(b_j)$, we use the following features

[‡]Due to space limitations, we omit the motivations for these features and refer the reader to the literature cited above.

- Length of $\mathbf{x}(b_j)$.
- Relative Position of $\mathbf{x}(b_j)$.
- Final punctuation of $\mathbf{x}(b_j)$.
- Number of capitalised words in $\mathbf{x}(b_j)$.
- Word Overlap of $\mathbf{x}(b_j)$ with the next one

$$W_{over}(\mathbf{x}(b_j), \mathbf{x}(b_j + 1)) = \frac{2 \mid x(b_j) \cap \mathbf{x}(b_j + 1) \mid}{\mid \mathbf{x}(b_j) \mid + \mid \mathbf{x}(b_j + 1) \mid}.$$

- First word of $\mathbf{x}(b_j)$.
- Bag Of Words (BOW) features: Let the bag of words of a set of sentences S be

$$B(S) = (c_0, c_1, ..., c_i, ..., c_{N-1}),$$

where N is the size of the dictionary and c_i is the frequency of word i in S.

  - BOW of $\mathbf{x}(b_j)$, $B(\{\mathbf{x}(b_j)\})$
  - BOW of $\mathbf{x}(b_j)$ and the previous sentence $B(\{\mathbf{x}(b_j - 1), \mathbf{x}(b_j)\})$
  - BOW of $\mathbf{x}(b_j)$ and the succeeding sentence $B(\{\mathbf{x}(b_j), \mathbf{x}(b_j + 1)\})$
  - The inner product of the two items above

- Cosine Similarity of $\mathbf{x}(b_j)$ and the previous sentence

$$CS(\mathbf{x}(b_j - 1), \mathbf{x}(b_j)) = \frac{\langle B(\mathbf{x}(b_j - 1)), B(\mathbf{x}(b_j)) \rangle}{\mid B(\mathbf{x}(b_j - 1)) \mid \times \mid B(\mathbf{x}(b_j)) \mid}$$

- Shannon's Entropy of $\mathbf{x}(b_j)$ computed by using a language model as described in Genzel and Charniak (2003).
- Quotes(Q_p, Q_c, Q_i). Q_p and Q_c are the number of pairs of quotes in the previous(Num_p) and current sentence (Num_c), $Q_p = 0.5 \times Num_p$ and $Q_c = 0.5 \times Num_c$.

5.5.1 Edge Features Φ_2

Below is the set of features $\Phi_2(b_j, b_{j+1}, \mathbf{x})$ encoding information about the current segment. These features represent the power of the Semi-Markov models. Note that Φ_3 features also belong to edge features category. In this chapter, we did not use Φ_3 feature due to computational issues.

- Length of The Paragraph: This feature expresses the assumption that one would want to have a balance across the lengths of the paragraphs assigned to a text. Very long and very short paragraphs should be uncommon.

- Cosine Similarity of the current paragraph and neighbouring sentences: Ideally, one would like to measure the similarity of two consecutive paragraphs and search for a segmentation that assigns low similarity scores (in order to facilitate changes in the content). This can be encoded using $\Phi_3(\mathbf{x}, b_{j-1}, b_j, b_{j+1})$ features. When such features are computationally expensive, one can measure the similarity of the current paragraph with the preceding sentence as

$$CS(P, \mathbf{x}(b_j - 1)) = \frac{\langle BOW(P), BOW(\mathbf{x}(b_j - 1)) \rangle}{\mid BOW(P) \mid \times \mid BOW(\mathbf{x}(b_j - 1)) \mid}$$

  where P is the set of sentences in the current paragraph, $[b_j, b_{j+1})$. A similar feature is used for $CS(P, \mathbf{x}(b_{j+1}))$.

- Shannon's Entropy of the Paragraph: The motivation for including features encoding the entropy of the sentences is the observation that the entropies of a paragraph's initial sentences are lower than the others (Genzel and Charniak, 2003). The motivation for including features encoding the entropy of the paragraphs, on the other hand, is that the entropy rate should remain more or less constant across paragraphs, especially for long texts like books. We ignore the sentence boundaries and use the same technique that we use to compute the entropy of a sentence.

5.5.2 Feature Rescaling

Most of the features described above are binary. There are also some features such as the entropy whose value could be very large. We rescale all the non-binary

valued features so that they do not override the effect of the binary features. The scaling is performed as follows:

$$u_{new} = \frac{u - \min(u)}{\max(u) - \min(u)}$$

where u_{new} is the new feature and u is the old feature. $\min(u)$ is the minimum of u, and $\max(u)$ is the maximum of u. An exception to this is the rescaling of BOW features which is given by

$$B(\mathbf{x}(b_j))_{new} = B(\mathbf{x}(b_j))/\langle B(\mathbf{x}(b_j)), B(\mathbf{x}(b_j))\rangle.$$

5.6 Experiments

We collected four sets of data for our experiments. The first corpus, which we call SB, consists of manually annotated text from the *same book The Adventures of Bruce-Partington Plans* by Arthur Conan-Doyle. The second corpus, which we call SA, again consists of manually annotated text but from 10 different books by the *same author* Conan-Doyle. Our third corpus consists of German (GER) and English (ENG) texts. The German data consisting of 12 German novels was used by Sporleder and Lapata (2006). This data uses automatically assigned paragraph boundaries, with the labelling error expected to be around 10%. The English data contains 12 well known English books from Project Gutenberg (`http://www.gutenberg.org/wiki/Main_Page`). For this dataset the paragraph boundaries were marked manually.

All corpora were approximately split into training (72%), validation (21%), and test set (7%) (see Table 5.1). The table also reports the accuracy of the baseline classifier, denoted as BASE, which either labels all sentences as paragraph boundaries or non-boundaries, choosing whichever scheme yields a better accuracy.

We evaluate our system using accuracy, precision, recall, and the F_1-score given by $(2 \times Precision \times Recall)/(Precision + Recall)$ and compare our results to Sporleder and Lapata (2006) who used BoosTexter (Schapire and Singer, 2000) as a learning algorithm. To the best of our knowledge, BoosTexter (henceforth called BT) is the leading method published for this task so far. In order to evaluate the importance of the edge features and the resultant large-margin constraint, we also compare against a standard binary Support Vector Machine (SVM) which uses node features alone to predict whether each sentence is the beginning of a

	TOTAL	TRAIN	DEV	TEST	BASE
SB	59,870	43,678	12,174	3,839	53.70
SA	69,369	50,680	14,204	4,485	58.62
ENG	123,261	88,808	25,864	8,589	63.41
GER	370,990	340,416	98,610	31,964	62.10

Table 5.1: Number of sentences and % accuracy of the baseline classifier (BASE) on various datasets used in our experiments.

DATASET	ALGO.	ACC.	REC.	PREC.	F_1
ENG	SMM	**75.61**	**46.67**	**77.78**	**58.33**
	SVM	58.54	26.67	40.00	32.00
	BT	65.85	33.33	55.56	41.67
GER	SMM	70.56	46.81	65.67	54.66
	SVM	39.92	**100.00**	38.68	55.79
	BT	**72.58**	54.26	**67.11**	**60.00**

Table 5.2: Test results on ENG and GER data after model selection.

paragraph or not. For a fair comparison, all classifiers used the linear kernel and the same set of node features.

We perform model selection for all three algorithms by choosing the parameter values that achieve the best F_1-score on the development set. For both the SVM as well as our algorithm, SMM, we tune the parameter C (see (4.4a)) which measures the trade-off between training error and margin. For BT, we tune the number of Boosting iterations, denoted by N.

In our first experiment, we compare the performance of our algorithm, SMM, on the English and German corpus to a standard SVM and BoosTexter. As can be seen in Table 5.2, our algorithm outperforms both SVM and BT on the ENG corpus and performs very competitively on the GER corpus, achieving accuracies close to those of BT. The SVM does not take into account edge features and hence does not perform well on this task.

We observed a large discrepancy between the performance of our algorithm on the development and the test datasets. The situation is similar for both SVM and BT. For instance, BT when trained on the ENG corpora, achieves an optimal F_1-score of 18.67% after $N = 100$ iterations. For the same N value, the test performance is 41.67%. We conjecture that this discrepancy is because the books

that we use for training and test are written by different authors. While there is some generic information about when to insert a paragraph break, it is often subjective and part of the authors style. To test this hypothesis, we performed experiments on the SA and SB corpus, and present results in Table 5.3. Indeed, the F_1-scores obtained on the development and test corpus closely match for text drawn from the same book (whilst exhibiting better overall performance), differs slightly for text drawn from different books by the same author, and has a large deviation for the GER and ENG corpus.

DATASET	ACC.	REC.	PREC.	F_1-SCORE
SB (DEV)	92.81	86.44	92.73	89.47
SB (TEST)	96.30	96.00	96.00	96.00
SA (DEV)	82.24	61.11	82.38	70.17
SA (TEST)	81.03	79.17	76.00	77.55
ENG (DEV)	69.84	18.46	78.63	29.90
ENG (TEST)	75.61	46.67	77.78	58.33
GER (DEV)	73.41	41.61	38.46	39.98
GER (TEST)	70.56	46.81	65.67	54.66

Table 5.3: Comparison on different APS datasets on SMM.

In our next experiment, we investigate the effect of the offset (the weight assigned to the constant feature Φ_0) on the performance of our algorithm. We fix the best value of C from the development dataset as above, but now we vary the offset parameter. If we now use the best offset, tuned for accuracy or F_1-score, as the case may be, the performance on the test set changes. This is shown in Tables 5.4 and 5.5.

DATASET	ACC.	REC.	PREC.	F_1-SCORE
ENG (DEV)	70.90	26.15	72.10	38.38
ENG (TEST)	73.17	60.00	64.29	62.07
GER (DEV)	80.95	19.25	69.71	30.17
GER (TEST)	68.55	24.47	76.67	37.10

Table 5.4: Performance on development and test set after tuning the offset for the best accuracy.

DATASET	ACC.	REC.	PREC.	F_1-SCORE
ENG (DEV)	55.02	98.46	43.31	60.16
ENG (TEST)	39.02	93.33	36.84	52.28
GER (DEV)	64.29	63.35	32.70	43.14
GER (TEST)	75.40	73.40	65.71	69.35

Table 5.5: Performance on development and test set after tuning the offset for best F_1-score.

DATASET	ALGO.	ACC.	REC.	PREC.	F_1-SCORE
ENG	SMM	**77.71±6.18**	33.44±13.98	**64.33±21.85**	**40.12±11.22**
	SVM	66.95±5.28	**37.06±9.95**	37.78±14.24	34.72±4.86
	BT	75.44±7.27	23.43±12.75	52.03±28.26	29.47±12.58
GER	SMM	76.68±3.71	**50.87±10.80**	**60.96±10.87**	**55.15±10.08**
	SVM	67.22±7.50	19.88±8.60	34.48±10.99	24.70±8.45
	BT	**77.29±2.40**	47.06±16.71	59.49±12.62	51.85±14.30
SB	SMM	86.46±8.41	73.62±19.43	86.47±10.22	78.46±15.11
	SVM	63.73±10.05	41.47±13.19	50.31±19.36	43.48±10.61
	BT	**87.99±6.24**	**77.51±14.06**	**87.13±10.38**	**81.23±9.55**
SA	SMM	**82.96±6.22**	**65.60±14.06**	**78.53±11.63**	**71.13±12.57**
	SVM	58.26±8.90	49.92±13.15	38.64±15.37	41.05±10.42
	BT	78.41±7.35	57.75±15.39	70.08±18.90	62.46±15.23

Table 5.6: Performance of various algorithms on our test corpus.

Thus far, following Sporleder and Lapata (2006) we worked with a single random split of the data into training, development, and test set. In our final experiment we test the statistical significance of our results by performing 10-fold cross validation. For this experiment, we randomly pick $1/3^{\text{rd}}$ of the data from each corpus and tune parameters on this development set. The parameters are now fixed, and the rest of the data is used to perform 10-fold cross validation. The results are summarised in Table 5.6. While the performance of our algorithm is relatively unchanged on the large GER dataset, there are large variations on the relatively small ENG, SA, and SB datasets. This is to be expected because 10-fold cross-validation on small samples can skew the relative distribution of the examples used for training and testing.

5.7 Conclusion

We presented a competitive algorithm for paragraph segmentation which uses the ideas from large margin classifiers and graphical models to extend the semi-Markov formalism to the large margin case. We obtain an efficient dynamic programming formulation for segmentation which works in linear time in the length of the sequence. Experimental evaluation shows that our algorithm is competitive when compared to state-of-the-art methods.

Chapter 6

Action Segmentation and Recognition

A challenging task in human action understanding is to segment and recognise a video sequence of continuous elementary actions *e.g.* running and walking. This has a wide range of applications in surveillance, video retrieval and intelligent interfaces. The difficulty comes from high variability of appearances, shapes and possible occlusions. The task is typically done in two steps: 1) segmenting and then 2) classifying the segments. Using semi-Markov model (SMM), we can segment and classify the video simultaneously.

6.1 Max Margin Approach

As commonly used in Schuldt et al. (2004); Dollar et al. (2005); Wong et al. (2007); Jhuang et al. (2007); Nowozin et al. (2007), we assume only one person appears in a given video sequence $\mathbf{x}$ (and we allow people in different video to be different) performing actions labeled as $\mathbf{y} = \{(b_k, l_k)\}_{k=0}^{l-1}$ consisting of pairs (b_k, l_k) that indicates the beginning position b_k and its corresponding action l_k for the kth segment $[b_k, b_{k+1})$. Denote $F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$ the discriminant function. For an unseen video sequence $\mathbf{x}$, we predict the label via

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}} \ \log p(\mathbf{y} \,|\, \mathbf{x}, \mathbf{w}) = \underset{\mathbf{y}}{\operatorname{argmax}} \, F(\mathbf{x}, \mathbf{y}; \mathbf{w}). \tag{6.1}$$

Learning the SMM discriminatively from training data is essentially a regularised empirical risk minimisation problem (Tsochantaridis et al., 2005; Taskar et al., 2004) with respect to $\mathbf{w}$ as in (4.4). The minimisation can be done by the cutting

Algorithm 5 Bundle Method

Input: sequence $\mathbf{x}_i$ and true label $\mathbf{y}_i$ for example i, sample size m, precision $\epsilon > 0$
Initialise $\mathbf{w} = 0$
repeat
  for $i = 1$ **to** m **do**
    $\overline{\mathbf{y}_i} = \text{argmax}_{\mathbf{y}} \, \Delta(\mathbf{y}_i, \mathbf{y}) + F(\mathbf{x}_i, \mathbf{y}; \mathbf{w})$
    Compute the empirical loss $R_{\text{emp}}(\mathbf{w})$ (6.2) and the gradient $\nabla_{\mathbf{w}} R_{\text{emp}}(\mathbf{w})$ (6.3).
    $\mathbf{w} \leftarrow \text{BMRM}(R_{\text{emp}}(\mathbf{w}), \nabla_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}))$
  end for
until $R_{\text{emp}}(\mathbf{w}) \leq \epsilon$

plane method as we did for the APS problem in the previous chapter. Alternatively we can do it by Bundle Methods for Regularised Risk Minimisation (BMRM) (Teo et al., 2007; Smola et al., 2007). Similar to the cutting plane method, we need to compute the most violated label $\overline{\mathbf{y}}$ which can be efficiently obtained by a viterbi-like dynamic programming. BMRM requires two inputs: the empirical risk

$$R_{\text{emp}}(\mathbf{w}) := \frac{1}{m} \sum_i \Delta(\mathbf{y}_i, \overline{\mathbf{y}_i}) - \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \overline{\mathbf{y}_i}) \rangle \,, \tag{6.2}$$

and its gradient

$$\nabla_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}) = -\frac{1}{m} \sum_i \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \overline{\mathbf{y}_i}). \tag{6.3}$$

Empirical studies in Section 6.4 show that the bundle method often delivers superior results to those of the cutting plane method as observed in Teo et al. (2007); Smola et al. (2007).

6.2 Viterbi-Like Inference

For both learning algorithms (cutting plane and BMRM), we need to infer

$$\overline{\mathbf{y}_i} = \underset{\mathbf{y} \in \mathcal{Y}}{\text{argmax}} \; \Delta(\mathbf{y}_i, \mathbf{y}) + F(\mathbf{x}_i, \mathbf{y}; \mathbf{w}). \tag{6.4}$$

This can be done by dynamic programming due to decomposition of the feature Φ and the cost function Δ. $\Phi(\mathbf{x}, \mathbf{y})$ can be decomposed into local features as

$$\Phi(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=0}^{l-1} \Phi_1(\mathbf{x}, n_i, c_i), \sum_{i=0}^{l-1} \Phi_2(\mathbf{x}, n_i, n_{i+1}, c_i), \sum_{i=0}^{l-1} \Phi_3(\mathbf{x}, n_i, n_{i+1}, c_i, c_{i+1}) \right).$$

Here local features Φ_1 and Φ_2 capture the observation-label dependencies within the current action segment: Φ_1 encodes the information on the boundary frame and Φ_2 encodes the overall characteristics of the entire segment. The interaction between two neighboring segments is encoded in Φ_3. Similarly, F can also be decomposed into three components $f_i(\mathbf{x}, \mathbf{y}) = \left\langle \mathbf{w}^{(i)}, \phi_i(\mathbf{x}, \mathbf{y}) \right\rangle, \forall i = \{1, 2, 3\}$ as

$$\sum_{i=0}^{l-1} \Big(f_1(\mathbf{x}, n_i, c_i) + f_2(\mathbf{x}, n_i, n_{i+1}, c_i) + f_3(\mathbf{x}, n_i, n_{i+1}, c_i, c_{i+1}) \Big).$$

Here $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \mathbf{w}^{(3)}$ are components of $\mathbf{w}$. We use the Hamming distance to measure the label cost $\Delta(\mathbf{y}, \mathbf{y}')$ between alternative action sequence labels as

$$\sum_{k=0}^{m-1} (1 - \mathbf{1}(\mathbf{y}^{(k)} = \mathbf{y}'^{(k)})),$$

where $\mathbf{1}(.)$ is the indicator function. Apparently, the label cost is decomposable as well.

The decomposition of features and label cost function leads to a Viterbi-like dynamic programming procedure proposed in Algorithm 6. Intuitively, it iteratively asks for the best segmentation and classification for all previous frames, assuming the current frame is a boundary frame with a known action class, and then reuses and stores partial information as needed. However, we don't know whether the current frame is a boundary frame nor the true class of that segment. The good news is that, in the end, we know the last frame is an ending frame, and we will know the most likely class of the last frame by mamimising F. Then we can back track all previous boundaries and classes since the information is stored already.

For any segment i, we denote its related boundaries as $n_- := n_{i-1}$ and $n := n_i$. Similarly the related labels are $c_- := c_{i-1}$ and $c := c_i$. Now, we maintain a partial score $S(\mathbf{x}, n, c)$ that sums up to segment i (*i.e.* starts at position 0 and ends with the segment $[n_-, n)$ with labels c_- (for n_-) and c (for n), respectively), and it is defined as

$$\max_{c_-, \max\{0, n-M\} \le n_- < n} \Big\{ S(\mathbf{x}, n_-, c_-) + g(\mathbf{x}, n_-, n, c_-, c) \Big\}. \tag{6.5}$$

Here the increment

$$g(\mathbf{x}, n_-, n, c_-, c) = f_1(\mathbf{x}, n_-, c_-) + f_2(\mathbf{x}, n_-, n, c_-)$$
$$+ f_3(\mathbf{x}, n_-, n, c_-, c) + 1 - \sum_{k=n_-}^{n-1} \mathbf{1}(\mathbf{y}_k = c_-).$$

Algorithm 6 Viterbi-Like Inference

Input: sequence $\mathbf{x}$ of length l, its true label $\mathbf{y}$, and maximum length of a segment M

Output: score s, the most violated label $\overline{\mathbf{y}}$

Initialise matrices $S \in \mathbb{R}^l \times C$, $J \in \mathbb{Z}^l$, and $L \in \mathbb{Z}^l$ to 0, $\overline{\mathbf{y}} = \emptyset$

for $i = 1$ **to** l **do**

  for $c_i = 1$ **to** C **do**

    $(J_i, L_i) = \underset{j, c_j}{\operatorname{argmax}} S(j, c_j) + g(j, i, c_j, c_i)$

    $S(i, c_i) = S(j^*, c^*_{j^*}) + g(j^*, i, c^*_{j^*}, c_i)$

  end for

end for

$c^*_l = \underset{c_l}{\operatorname{argmax}} S(l, c_l)$

$s = S(l, c^*_l)$

$\overline{\mathbf{y}} \leftarrow \{(l, c^*_l)\}$

$i \leftarrow l$

repeat

  $\overline{\mathbf{y}} \leftarrow \{(J_i, L_i), \overline{\mathbf{y}}\}$

  $i \leftarrow J_i$

until $i = 0$

It is easy to verify that in the end, the sum of two terms in the RHS of (6.4) amounts to $S(l, c_l)$. This algorithm can also be used for inference in the prediction phase by letting

$$g(\mathbf{x}, n_-, n, c_-, c) = f_1(\mathbf{x}, n_-, c_-) + f_2(\mathbf{x}, n_-, n, c_-) + f_3(\mathbf{x}, n_-, n, c_-, c).$$

This inference algorithm is very efficient — time complexity $O(lMC^2)$, linear to the sequence length l and memory complexity $O(l(C + 2))$. Our C++ implementation* processes the video sequences at 20 frames per second (FPS) on average on an desktop with Intel Pentium 4 3.0GHz processor and 512M memory.

6.3 Feature Representation

Neuro-psychological findings such as Phillips et al. (2002) suggest that the visual and motor cortices of human perception system are more responsible than the

*Source code can be downloaded from http://users.rsise.anu.edu.au/~qshi/code/smm_release.tgz.

semantic ones for retrieval and recognition of visual action patterns. This motivates us to represent action features Φ by a set of local features that capture the salient aspect of spatial and temporal video gradients.

The foreground object in each image is obtained using an efficient background subtraction method in Cheng et al. (2006). By applying the SIFT (Lowe, 2004) key points detector, the object is represented as a set of key feature points extracted from the foreground with each point having a 128-dimensional feature vector. Importantly, SIFT features bear these properties that are critical in our context, as being relatively invariant to illumination and view-angle changes, as well as being insensitive to the objects' color appearance by capturing local image textures in the gradient domain. In addition, from each feature point, we construct an additional 60-dimensional shape context features (Belongie et al., 2002) that roughly encode how each point "sees" the remaining points. The two sets of features are then concatenated with proper scaling to form a 188-dimensional vector. This point-set object representation are further transformed into a 50-dimensional codebook using K-means, similar to the visual vocabulary approach in Sivic and Zisserman (2003). Therefore, once a new frame is presented, its key points will be projected into the existing codebook space with clustering assignments. Thus the object is now represented as a 50-dimensional histogram vector. Typical results of the codebook representation are illustrated in Figure 6.4 (bottom row), where we randomly choose four codebook clusters and impose the assigned feature point locations on individual images. Figure 6.4 convincingly shows that each cluster of points is able to pick up patches in certain human body areas, over time and across different people. For example, the white triangles tend to stay on the hips.

With this codebook representation, we now construct feature functions Φ_1, Φ_2 and Φ_3 as follows.

Boundary Frame Features $\Phi_1(\mathbf{x}, n_i, c_i) = \varphi_1(\mathbf{x}, n_i) \otimes c_i$, where $\otimes$ denotes a tensor product (similar to (11) and (12) of Tsochantaridis et al. (2005)). φ_1 is a concatenation of two features. The first is a constant 1 which acts as the bias or offset term. The second part is obtained from a local window with width w_s centered on the boundary frame. When $w_s = 1$ it becomes a single histogram vector.

Node Features on Segments Node features are devised to capture the characteristics of the segment. Φ_2 is defined as $\Phi_2(\mathbf{x}, n_i, n_{i+1}, c_i) = \varphi_2(\mathbf{x}, n_i, n_{i+1}) \otimes c_i$.

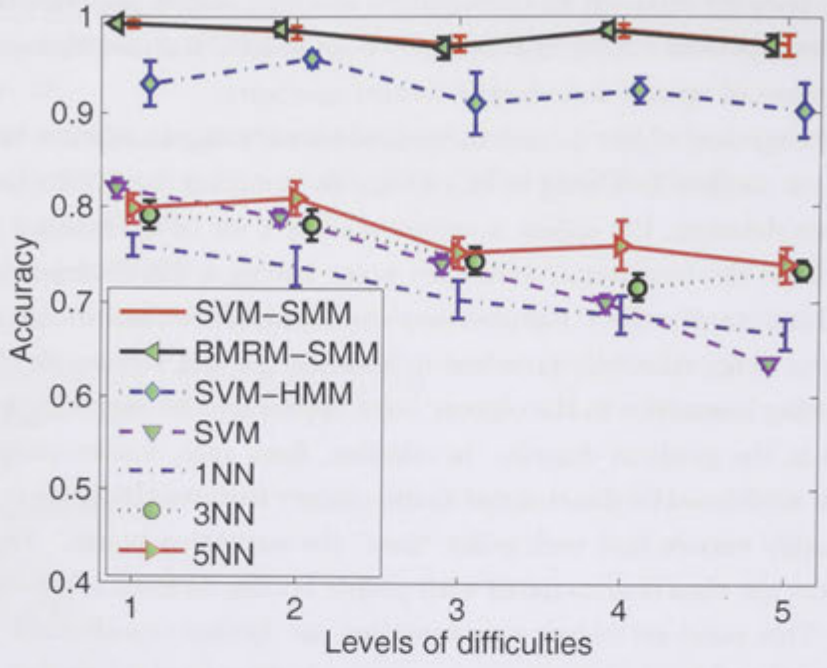


Figure 6.1: Comparing seven methods for action recognition on the synthetic dataset.

$\varphi_2(\mathbf{x}, n_i, n_{i+1})$ contains three components: the length of this segment, the mean and the variance of the histogram vector of the segment (*e.g.* over frames from n_i to $n_{i+1} - 1$).

Edge Features on Neighboring Segments As in practice we have prior knowledge about the minimum length of an action segment, we define the minimum duration of a segment as d to reduce the complexity of the Viterbi algorithm. $\Phi_3(\mathbf{x}, n_i, n_{i+1}, c_i, c_{i+1}) = \varphi_3(\mathbf{x}, n_i, n_{i+1}) \otimes c_i \otimes c_{i+1}$, and it is a concatenation of the following components: a) the mean of the histogram vector from frames n_i to $n_{i+1} - 1$, and b) from frames n_{i+1} to $n_{i+1} + d$, as well as c) the variance of the histogram vector from n_i to $n_{i+1} - 1$, and d) from n_{i+1} to $n_{i+1} + d$.

6.4 Experiments

During the following experiments, the proposed discriminative SMM approach is compared to three algorithms: KNN (where K=1, 3, 5), SVM multiclass and SVM-HMM (Tsochantaridis et al., 2005). In particular, two variants of discriminative SMM are considered, namely the one with cutting plane method (SVM-SMM) and the one with bundle method (BMRM-SMM).

By default, we set $\epsilon = 1e-4$, $M = 3$, and $w_s = 3$. The trade-off parameter η of each method (SVM multiclass, SVM-HMM, SVM-SMM and BMRM-SMM) is tuned separately using cross-validation. Moreover, we evaluate the action recognition and segmentation performance separately: A frame-wise recognition rate is utilised to benchmark the recognition performance for each of the comparison algorithms. To measure segmentation performance, we adopted the F_1-score, which is often used in information retrieval tasks, and is given by $(2 \times \text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall})$.

6.4.1 Synthetic dataset

We start with a controlled experiment where we are able to quantitatively measure the performance of comparison algorithms by varying the difficulty level of problems from easy to difficult. We do this by constructing a two-person two-action synthetic dataset consisting of five trials, where each trial has a set of ten sequences and corresponds to a certain level of difficulty[†]. Here a sequence of a person P is sampled from a semi-Markov model/process with their own Gaussian emission probabilities $\mathcal{N}(\mu_{c,P}, \sigma_{c,P})$ and duration parameters $\lambda_{c,P}$ for the two actions $c = 1, 2$, respectively. We sample 150 frames from the SMM model for each given person and action. Now, we build five trials as follows: For each trial, five sequences are generated from each person's model, and in the end we have ten sequences. Across trials, we vary the level of difficulty by moving μ_2 toward μ_1 and fixing other parameters of the models.

Figure 6.1 displays the action recognition results on this dataset, where 5-fold cross-validation is utilised. Here both discriminative SMM variants consistently outperform others: In fact, both SVM-SMM and BMRM-SMM give almost the same recognition accuracy regardless of the level of difficulty. They are followed by SVM-HMM while the other methods (namely SVM and KNNs) have inferior performance. This is due to SMM exploiting the contextual information from neighboring nodes up to neighboring segments.

6.4.2 KTH dataset

The KTH dataset (Schuldt et al., 2004) contains 25 individuals performing six activities: *running*, *walking*, *jogging*, *boxing*, *handclapping* and *handwaving*, where each sequence contains a single action with multiple action cycles. Figure 6.2

[†]This dataset can be downloaded at http://users.rsise.anu.edu.au/~qshi/code/smm_release.tgz.

Figure 6.2: Sample frames of one person engaging in six types of actions in the KTH dataset.

Method	Brief Description	Accuracy
Ke et al. (2005)	cascade classifier, spatio-temporal	63.0
Schuldt et al. (2004)	SVM, local space time features	71.7
Schindler and van Gool (2008)	SVM, bag of snippets, shape and motion	92.7
Dollar et al. (2005)	SVM, "cuboid" features	81.2
Nowozin et al. (2007)	linear SVM, "cuboid" features	87.0
	subsequence boosting, "cuboid" features	84.7
Wong et al. (2007)	WX-SVM, "cuboid" features	91.6
Our SVM	baseline SVM, "cuboid" features	85.1
Our BMRM-SVM	discriminative SMM, "cuboid" features	**95.0**

Table 6.1: Action recognition rates on KTH dataset.

displays exemplar frames of one person performing each of the six activities.

To make direct comparisons to existing methods in literature presented in Table 6.1, in this experiment we adopt the "cuboid" (Dollar et al., 2005) feature (instead of SIFT) that captures the local spatio-temporal characteristics using Gabor filters. More specifically, this detector is tuned to fire whenever variations in local image intensities contain distinguishing spatio-temporal characteristics. At each detected interest point location, a 3D cuboid is then extracted and represented as a flattened vector that contains the spatio-temporal windowed information including normalised pixel values, brightness gradient and windowed optical flow.

We adopt the same train and test sets splits as that of Nowozin et al. (2007), only here our models are trained on the joined train+validation sets: Each model

truth vs. predict	boxing	handclapping	handwaving	jogging	running	walking
boxing	0.91	0.09	0.00	0.00	0.00	0.00
handclapping	0.00	0.96	0.00	0.00	0.04	0.00
handwaving	0.00	0.00	1.00	0.00	0.00	0.00
jogging	0.00	0.00	0.00	0.89	0.00	0.11
running	0.00	0.00	0.00	0.08	0.92	0.00
walking	0.00	0.00	0.00	0.12	0.00	0.88

Table 6.2: Confusion matrix of BMRM-SMM on the KTH dataset.

Figure 6.3: Sample frames of subjects each performing one of the four actions: slow walk, fast walk, incline walk and walk with a ball, in an action sequence of the CMU MoBo dataset.

tuning parameters η of our methods are selected using 5-fold cross-validation on the joined sets, then a single model is trained on the joined sets, and the final accuracy is reported on the test set. Table 6.1 shows the results of our methods: Our SVM baseline (85.1%) is comparable to similar methods (e.g. SVM of Dollar et al. (2005); Nowozin et al. (2007)) reported in literature, while our BMRM-SMM (95.0%) performs favorably comparing to these state-of-the-art methods. We attribute this to the contextual information that we are able to exploit through the use of Φ_2 features in our SMM framework. Table 6.2 displays the confusion matrix of the BMRM-SMM method, where the *handwaving* action can be perfectly identified from the rest of the actions. On the other hand, there are a few mistakes among the three easy-to-be-confused categories: *walking*, *jogging*, and *running*.

	1NN	SVM	HMM	SVM-HMM	SVM-SMM	BMRM-SMM
Acc	0.65 ± 0.02	0.67 ± 0.03	0.68 ± 0.08	0.75 ± 0.06	0.75 ± 0.03	**0.78 ± 0.07**
F_1	0.16 ± 0.05	0.15 ± 0.03	n/a ± n/a	0.43 ± 0.01	**0.59 ± 0.03**	**0.59 ± 0.03**

Table 6.3: Accuracies (Acc) and F_1 scores on CMU MoBo dataset.

1NN	3NN	5NN	SVM	SVM-HMM	SVM-SMM	BMRM-SMM
0.82 ± 0.02	0.80 ± 0.03	0.77 ± 0.03	0.84 ± 0.03	0.87 ± 0.02	0.91± 0.02	**0.94± 0.01**

Table 6.4: Action recognition rates on the WBD dataset.

6.4.3 CMU MoBo dataset

This dataset (R.Gross and Shi, 2001) contains 24 individuals[‡] walking on a treadmill. As illustrated in Figure 6.3, each subject performs in a video clip one of the four different actions: *slow* walk, *fast* walk, *incline* walk and slow walk *with a ball*. Each sequence has been pre-processed to contain several cycles of a *single* action and we additionally manually label the boundary positions of these cycles. The task on this dataset is to automatically partition a sequence into atomic action cycles, as well as predict the action label of this sequence.

Table 6.3 presents the results averaged over 6-fold cross-validation. The results of 3NN and 5NN are omitted here as they are very similar to 1NN. We also experiment with generative HMM solely on the task of action recognition (predicting action label for each sequence), where one HMM is trained for each action type using the BaumWelch algorithm. It performs slightly better than the baseline methods including KNN (K=1,3,5) and SVM, but is still inferior to SVM-HMM (Tsochantaridis et al., 2005), its discriminative counterpart. Note that both SMM variants (SVM-SMM and BMRM-SMM) significantly outperform the other methods including SVM-HMM on action label prediction as well as on segmentation of action cycles.

6.4.4 WBD: A Dataset of Continuous Actions

In addition to the existing datasets (such as the MoBo and the KTH datasets), where each sequence contains exactly one type of action, we construct a Walk-Bend-Draw (WBD) dataset of continuous actions. Some exemplar frames are displayed in Figure 6.4. This is an indoor video dataset containing three subjects,

[‡]The dataset originally consists of 25 subjects. We drop the last person since we experienced technical problems obtaining the sequences of this individual walking with balls.

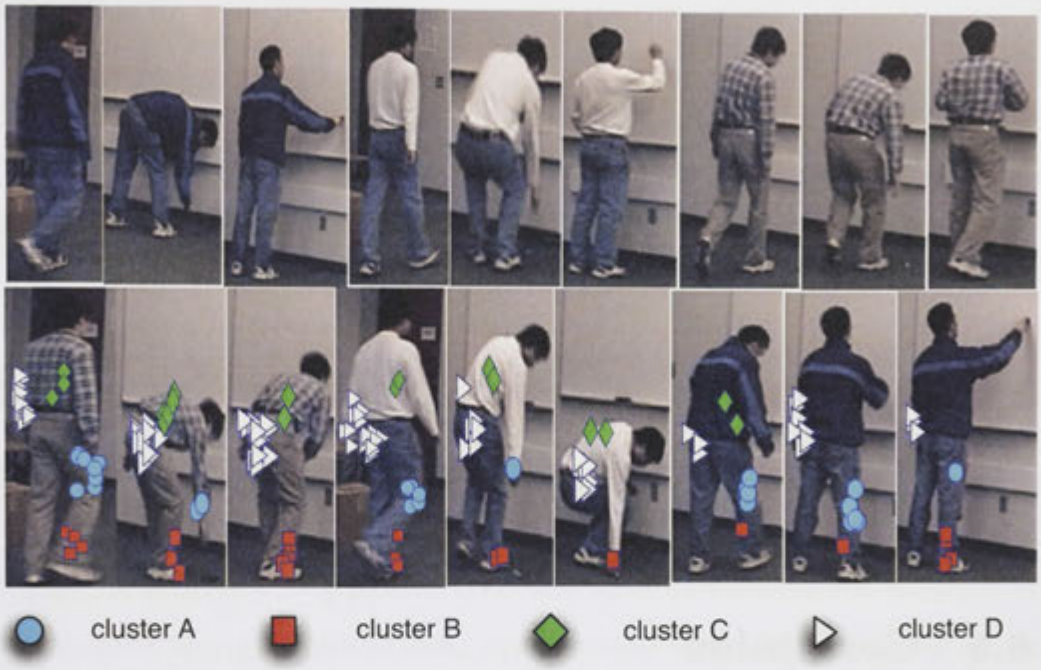


Figure 6.4: A Walk-Bend-Draw (WBD) dataset. **Top** shows some sample frames of the dataset . **Bottom** displays the assignments of image feature points on four randomly chosen codebook clusters over time and across person.

each performing six action sequences at 30 FPS at a resolution of 720×480, and each sequence consists of three continuous actions: slow *walk*, *bend* body and *draw* on board, and on average each action lasts about 2.5 seconds. We subsample each sequence to obtain 30 key frames, and manually label the ground truth actions.

The comparison results, obtained using 6-fold cross-validation, are summarised in table 6.4. Both discriminative SMM variants consistently deliver the best results, while here BMRM-SMM slightly outperforms SVM-SMM. They are then followed by SVM-HMM, SVM, and the KNN methods, in an order that is consistent with the experimental results for the synthetic dataset. Furthermore, Tables 6.5 and 6.6 display the confusion matrices of the two SMM variants: SVM-SMM vs. BMRM-SMM. where the two actions – *walk* and *draw* – seem to be rarely confused with each other, nevertheless both sometimes are mis-interpreted as bend. This is to be expected, as although *walk* and *draw* appear to be more similar to human observer in isolated images, it nevertheless can be learned by discriminative SMM methods that *walk*, *bend* and *draw* are usually conducted in order.

truth vs. predict	walk	bend	draw
walk	0.93	0.07	0.00
bend	0.05	0.93	0.02
draw	0.01	0.09	0.89

Table 6.5: Confusion matrix of SVM-SMM on WBD.

truth vs. predict	walk	bend	draw
walk	0.91	0.09	0.00
bend	0.03	0.93	0.04
draw	0.00	0.04	0.96

Table 6.6: Confusion matrix of BMRM-SMM on WBD.

6.5 Conclusion

We present a novel discriminative semi-Markov approach to human action analysis, where we intend to simultaneously segment and recognise continuous action sequences. We then devise a Viterbi-like dynamic programming algorithm that is able to efficiently solve the inference problem, and show the induced learning problem can be cast as a convex optimisation problem with many constraints, that can be subsequently solved and we present two such solvers. Empirical simulations demonstrate that our approach is competitive to and often outperforms the state-of-the-art methods.

Our approach can be extended in several directions. It is promising to explore the dual representation in order to incorporate matching cost between point sets. On future work we also plan to apply this approach to closely related problems, such as detecting unusual actions from a large video dataset.

Chapter 7

Hybrid Models on NLP and Image Categorisation

CRFs and SVMs can be seen as being representative of two different approaches to classification problems. The former is a probabilistic approach – the conditional probability of classes given each observation is explicitly modelled – while the latter is a max margin approach – classification is performed without any attempt to model probabilities.

Both approaches have their strengths and weaknesses. CRFs (Lafferty et al., 2001; Sha and Pereira, 2003) use a log loss which is known to be consistent. However, modelling $P(\mathbf{y} \mid \mathbf{x})$ often requires a large number of training examples and may sacrifice classification accuracy if the underlying distribution is complicated (Bulatov and Bousquet, 2007). In contrast, Support Vector Machines make more efficient use of training examples but are known to be inconsistent when there are more than two classes (Tewari and Bartlett, 2007; Liu, 2007).

Despite their different characteristics, CRFs and SVMs appear very similar when viewed as optimisation problems. The most salient difference is the loss used by each: CRFs are trained using a log loss while SVMs typically use a hinge loss.

In an attempt to capitalise on their relative strengths and avoid their weaknesses we propose a hybrid approach that uses a convex "blend" of these two losses. The new hybrid loss is conditionally consistent and has a generalisation bound guarantee. We postpone the detailed theoretical analysis to Chapter 8 (see the consistency of the hybrid loss in Section 8.1.2 and the generalisation bound of the hybrid loss in Section 8.2.3). We apply it to several natural language processing (NLP) applications and image categorisation before concluding with some

possible future refinements (Section 7.4).

7.1 The Hybrid Loss

The scores given to labels by a general model $f : \mathcal{X} \to \mathbb{R}^k$ can be transformed into a conditional probability distribution $p(\mathbf{x}; f) \in [0,1]^k$ by letting

$$p_{\mathbf{y}}(\mathbf{x}; f) = \frac{\exp(f_{\mathbf{y}}(\mathbf{x}))}{\sum_{\mathbf{y}\in\mathcal{Y}} \exp(f_{\mathbf{y}}(\mathbf{x}))}. \tag{7.1}$$

It is easy to show that under this interpretation the hinge loss for a probabilistic model $p = p(\cdot; f)$ is given by

$$\ell_H(p, \mathbf{y}) = \left[1 - \ln \frac{p_{\mathbf{y}}}{\max_{\mathbf{y}'\neq\mathbf{y}} p_{\mathbf{y}'}}\right]_+ \tag{7.2}$$

Another well known loss for probabilistic models, such as Conditional Random Fields, is the log loss

$$\ell_L(p, \mathbf{y}) = -\ln p_{\mathbf{y}}. \tag{7.3}$$

This loss penalises models that assign low probability to likely instances labels and, implicitly, that assign high probability to unlikely labels.

The *hybrid loss* introduced in this chapter is a loss for probabilistic models that is a convex combination of the hinge and log losses

$$\begin{aligned} \ell_\alpha(p, \mathbf{y}) &= \alpha\ell_L(p, \mathbf{y}) + (1-\alpha)\ell_H(p, \mathbf{y}) && (7.4)\\ &= -\alpha \ln(p_{\mathbf{y}}) + (1-\alpha)\left[1 - \ln \frac{p}{\max_{\mathbf{y}'\neq\mathbf{y}} p_{\mathbf{y}'}}\right]_+ && (7.5) \end{aligned}$$

where mixture of the two losses is controlled by a parameter $\alpha \in [0,1]$. Setting $\alpha = 1$ or $\alpha = 0$ recovers the log loss or hinge loss, respectively. The intention is that choosing α close to 0 will emphasise having the maximum label probability as large as possible while an α close to 1 will force models to prefer accurate probability assessments over strong classification.

7.2 Consistency and Generalisation bound

We will show in Section 8.1.2 that the hybrid loss can be conditionally consistent when the traditional hinge loss is not (see Theorem 16 in Section 8.1.3). Specifically, the hybrid loss can yield consistent predictions for instances with non-dominant labels provided the label probabilities are not too close. Also, we will show that the hybrid loss has a tighter generalisation bound than CRFs (see Theorem 19 in Section 8.2.3).

7.3 Applications

To show how the hybrid approach performs relative to the log and hinge losses, we apply it to a controlled synthetic multiclass dataset in which the dataset size and proportion of examples with non-dominant labels is carefully controlled. As might be expected, we observe that the hybrid loss outperforms the hinge loss when there are many instances with non-dominant labels and outperforms the log loss when relatively few training examples are available and most of those have dominant labels.

We also compare the hybrid loss to the log and hinge losses on several structured estimation problems and note that the hybrid loss regularly performs as well as either of the other losses.

7.3.1 Multiclass

We now run two multiclass simulations with a controlled data distribution $D(\mathbf{y}, \mathbf{x})$ having non-dominant class, *i.e.*, with instances with $D_{\mathbf{y}}(\mathbf{x}) < 1/2$ for all labels $\mathbf{y}$.

Non-dominant Distributions

Our first simulation is to see how the hinge, log and hybrid losses perform when all data are from a distribution that has no dominant class. In this case one of the labels $\mathbf{y}^*$ for each instance $\mathbf{x}$ has $D_{\mathbf{y}^*}(\mathbf{x}) = 0.46$ and the rest of the $D_{\mathbf{y}}(\mathbf{x}), \forall\, \mathbf{y} \neq \mathbf{y}^*$ are of equal value. The instances are drawn from Gaussians, $\mathbf{x} \sim N(\mu, \sigma^2)$, where $\mu = 1.2$, $\sigma = 0, 0.1, 0.2, 0.3, \ldots$. For $\sigma = 0$, we generate datasets with 100 examples from $D_{\mathbf{y}}(\mathbf{x})$ for the number of classes $|\,\mathcal{Y}\,| = 3, 4, 5, \ldots, 10$. For instance, the dataset with 10 classes has 46 examples from class $\mathbf{y}^*$, and 6 examples from each of the other classes. A good classifier should predict $\mathbf{y}^*$ as label for it is the most likely one. Thus a training error should be $1 - D_{\mathbf{y}^*}(\mathbf{x}) = 0.55$. In other words, predicting any other $\mathbf{y}$ instead of $\mathbf{y}^*$ will give higher error.The training errors for hinge, log and hybrid losses are plotted in Figure 7.1. As we can see clearly, the errors for the log and the hybrid losses remain a constant $(1 - D_{\mathbf{y}^*}(\mathbf{x}))$ as a good classifier should behave. Whereas the hinge loss error increases as the number of classes increases. This is in concordance with the consistency analysis of the three losses.

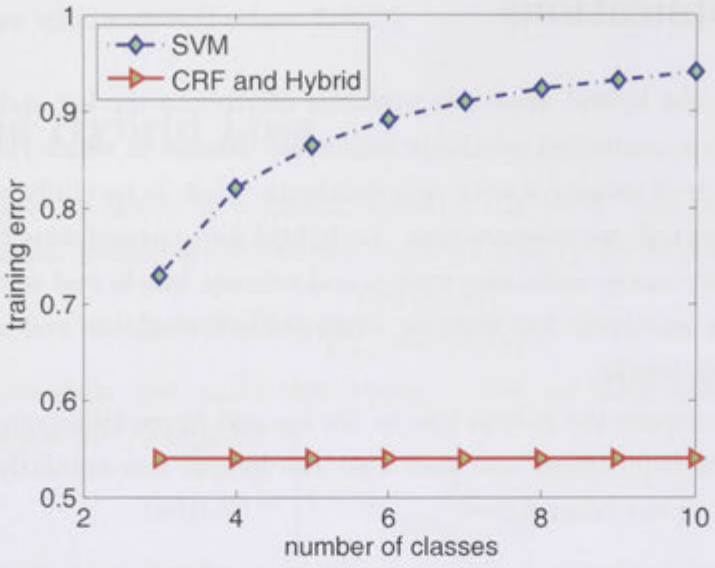
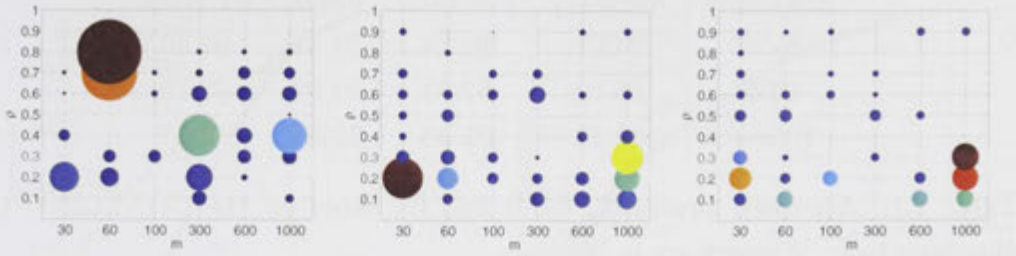


Figure 7.1: Training error curve with various number of classes. $\alpha = 0.5$ for the hybrid loss.

Mix of Non-dominant and Dominant Distributions

Minimising a consistent loss will always outperform an inconsistent one when non-dominant distributions are present if the entire data distribution is available. However, in practice, we often only have access to a small sample of the entire data distribution. Our second simulation is to study how the three losses perform given various training set sizes (denoted by m) and various proportions of instances with non-dominant distributions (denoted by ρ). We generate a 5 class data set with 100 feature dimensions as follows. In the non-dominant class case, the observation $\mathbf{x}$ is fixed and its conditional distribution is set to be $D_{\mathbf{y}^*}(\mathbf{x}) = 0.4$ and $D_{\mathbf{y}}(\mathbf{x}) = 0.15$ for $\mathbf{y} \neq \mathbf{y}^*$. In the dominant case, each dimension of the observation $\mathbf{x}$ is drawn from a one dimensional normal distribution $N(\mu = 1 + j, \sigma = 0.6)$ for the class $j = 1, \ldots, 5$. The proportion ρ ranges over the values $0.1, 0.2, 0.3, \ldots, 1$ and for each ρ, we generate the test set and the validation set with the same size 1000. Training set sizes of $m = 30, 60, 100, 300, 600, 1000$ are used. Given a mixing ratio ρ, we train models using the three losses on the training data with size m, and then apply the models to the test and validation data.

The results are summarised in Figure 7.2, from which we can see a clear trend — when the non-dominant class portion ρ is small (*e.g.*, when $\rho = 0.1$) and m is small (30,60), the hinge loss outperforms the log and hybrid losses. For larger m, the hybrid loss outperforms the log and hinge losses more often than not.



(a) Hybrid vs. Hinge (b) Hybrid vs. Log (c) Hinge vs. Log

Figure 7.2: Accuracy first vs. accuracy second. The colored dot indicates accuracy first > accuracy second. The size of the dot is proportional to the difference of two accuracies. Different color means the size is significantly different.

7.3.2 Text Chunking

Unlike the general multiclass case, structured estimation problems have a higher chance of non-dominant distributions because of the very large number of labels as well as ties or ambiguity regarding those labels. For example, in video segmentation, predicting a boundary with 1 or 2 frames offset from the human manually marked boundary is considered as "correct". Likewise in text chunking, tagging only one phrase differently while the rest are unchanged should not give totally different probability predictions – especially when there are ambiguities. Likewise in image denoising, changing a pixel's predicted graylevel (1 to 255) by 1, should not radically change the probability of the predicted whole image. Thus, because of the prevalence of non-dominant distributions, we expect that trained models using a hinge loss to perform poorly on these problems relative to training with hybrid or log losses.

CONLL2000 Text Chunking

Our first structured estimation experiment is carried out on the CONLL2000 text chunking task*. The data set has 8936 training sentences and 2012 testing sentences with 106978 and 23852 phrases (a.k.a., chunks) respectively. The task is to divide a text into syntactically correlated parts of words such as noun phrases, verb phrases, and so on. For a sentence with L chunks, its label consists of the tagging sequence of all chunks, *i.e.* $\mathbf{y} = (\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^L)$, where $\mathbf{y}^i$ is the chunking tag for chunk i. As is commonly used in this task, the label $\mathbf{y}$ is modelled as a 1D Markov chain, considering the dependency of adjacent chunking tags $(\mathbf{y}_i^j, \mathbf{y}_i^{j+1})$

*download from http://www.cnts.ua.ac.be/conll2000/chunking/

Algorithm	Accuracy	Precision	Recall	F1 Score
SVM	94.61	91.23	91.37	91.30
CRF	95.10	92.32	91.97	92.15
Hybrid	**95.11**	**92.35**	**92.00**	**92.17**

Table 7.1: Accuracy, precision, recall and F1 Score on the CONLL2000 text chunking task. Winners are in boldface.

Algorithm	Accuracy	Precision	Recall	F1 Score
SVM	94.64	87.58	88.30	87.94
CRF	95.21	90.07	88.89	89.48
Hybrid	**95.24**	**90.12**	**88.98**	**89.55**

Table 7.2: Accuracy, precision, recall and F1 Score on the baseNP chunking task. Winners are in boldface.

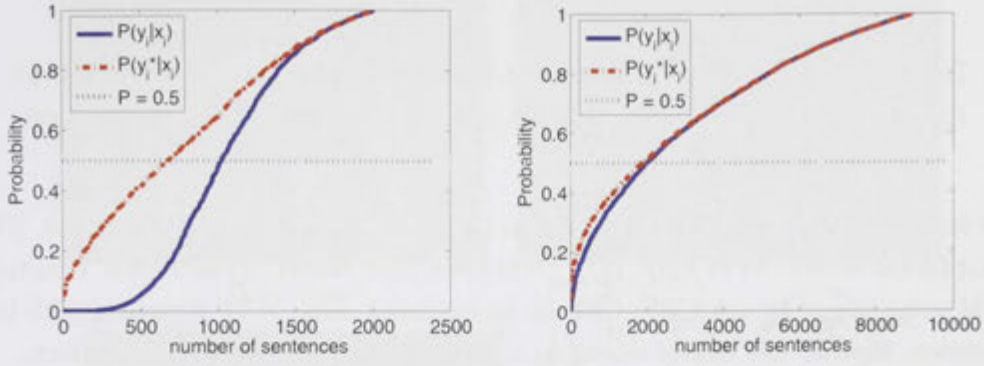
given observation $\mathbf{x}_i$. Clearly, the model has exponentially many possible labels, which suggest that there might be many non-dominant classes.

Since the true underlying distribution is unknown, we train a CRF (using the feature template from the CRF++ toolkit[†] and the CRF code[‡] from Leon Bottou) on the training set and then apply the trained model to both the testing and training datasets to get an estimate of the conditional distributions for each instance. We sort the sentences $\mathbf{x}_i$ from highest to lowest estimated probability on the true chunking label $\mathbf{y}_i$ given $\mathbf{x}_i$. The result is plotted in Figure 7.3, from which we observe the existence of many non-dominant distributions — about 1/3 of the testing sentences and about 1/4 of the training sentences.

We split the data into 3 parts: training (20%), testing (40%) and validation (40%). The regularisation parameter λ and the weight α are determined via parameter selection using the validation set. The accuracy, precision, recall and F1 Score on test set are reported in Table 7.2 for various used proportion of the training set used. As expected, when there is not an abundant amount of data, the hybrid loss outperforms both the SVM and CRF.

[†]download from http://crfpp.sourceforge.net/

[‡]download from http://leon.bottou.org/projects/sgd



(a) the testing set (b) the training set

Figure 7.3: Estimated probabilities of the true label $D_{\mathbf{y}_i}(\mathbf{x}_i)$ and most likely label $D_{\mathbf{y}_i^*}(\mathbf{x}_i)$. Sentences are sorted according to $D_{\mathbf{y}_i}(\mathbf{x}_i)$ and $D_{\mathbf{y}_i^*}(\mathbf{x}_i)$ respectively in ascending order. $D = 1/2$ is shown as the straight black dot line. About 700 sentences out of 2012 in the testing set and 2000 sentences out of 8936 in the training set have no dominant class.

baseNP Chunking

This dataset is provided in CRF++ toolkit that is mentioned before. It has 900 sentences in total. The task is to automatically classify a chunking phrase is as baseNP or not. We split the data and select λ and α in the same way as the above CONLL2000 data. We report the test accuracy, precision, recall and F1 Score in Table 7.2. The hybrid loss/model outperforms SVM and CRF on all measures.

7.3.3 Joint Image Categorisation

Our final experiment is joint image categorisation. The task is to categorise pre-segmented image areas by considering their dependency across the image segments. We use the well-known Corel dataset (Ren and Malik, 2003), which has 100 images and 7 classes: *hippo*, *polar bear*, *water*, *snow*, *vegetation*, *ground*, and *sky*. This is a very challenging task since there are 7^n many possible labels for an image with n segments.

The ground truth segmentations provided from the dataset are used as pre-segmented object regions. Therefore each image contains one or multiple objects/regions. We use 56 images for training, and 20 images for testing. The rest of the images are excluded either because they are too small or because they con-

Dataset	$\lvert\mathcal{Y}\rvert$	SVM-linear	SVM-RBF	SVM-Struct	CRF	Hybrid
Corel	7^n	58.62	65.52	**89.66**	86.21	**89.66**

Table 7.3: Image object categorisation on the Corel dataset. SVM-Linear: i.i.d. SVM using Linear kernel. SVM-RBF: i.i.d. SVM using RBF kernel. SVM-Struct: structured SVM using Linear kernel; CRF: CRF on sparse graph using MAP estimator with LBP inference. Hybrid: our hybrid model on a sparse graph. Winners are in boldface.

tain too many objects. The graphical model of an image is shown in Figure 7.4, where each segment is a node and edges capture the adjacency dependence.

Features

Any image with n segments and labels is represented as $(\mathbf{x}, \mathbf{y}) = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^n$, where the $\mathbf{x}^i$ and $\mathbf{y}^i$ are the i-th segment and corresponding label. We assume that global feature $\Phi(\mathbf{x}, \mathbf{y})$ is decomposed over singleton terms $\Phi_i(\mathbf{x}^i, \mathbf{y}^i)$, $\forall i, 1 \leq i \leq n$, as well as over pairwise terms $\Phi_{ij}(\mathbf{x}^i, \mathbf{y}^i)$, $\forall (i,j) \in \mathcal{A}_x$, where $\mathcal{A}_x$ is the set of adjacent segments in $\mathbf{x}$

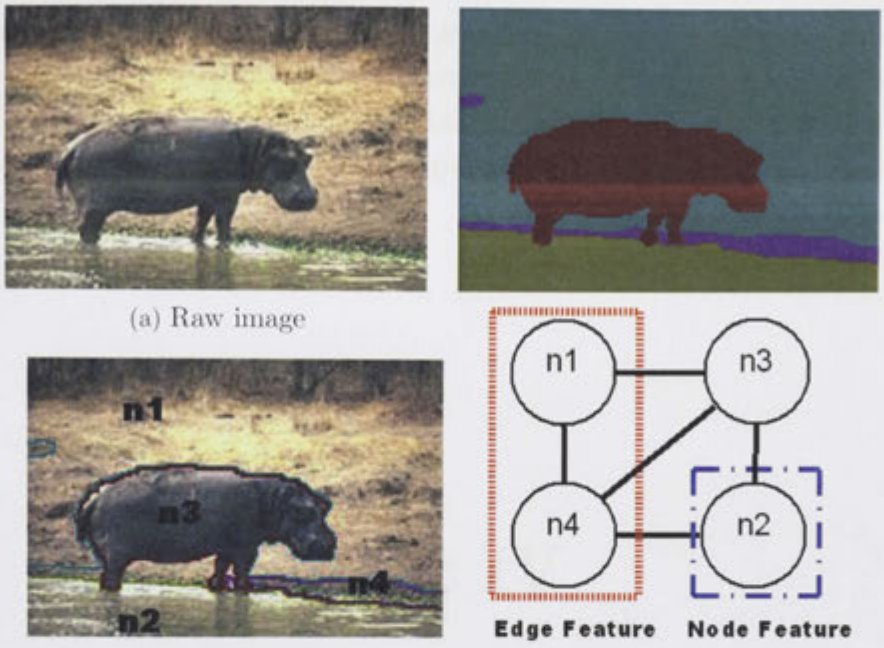
$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_i \Phi_i(\mathbf{x}^i, \mathbf{y}^i) + \sum_{(i,j)\in\mathcal{A}} \Phi_{ij}(\mathbf{x}^i, \mathbf{y}^i, \mathbf{y}^j). \tag{7.6}$$

We assume Φ_i is a tensor product of instance and label feature functions, given by $\Phi_i(\mathbf{x}, \mathbf{y}) = \varphi_i(\mathbf{x}) \otimes \mathbf{y}_i$ where φ_i is the raw node feature depending only on the observed segmented image. Similarly $\Phi_{ij}(\mathbf{x}, \mathbf{y}) = \varphi_{ij}(\mathbf{x}) \otimes \mathbf{y}_{ij}$, where φ_{ij} is the raw edge feature depending only on the observation, and $\mathbf{y}^{ij} := [\mathbf{y}^i \ \ \mathbf{y}^j]$. φ_i and φ_{ij} are assembled from

φ_1 We extract a well known texton feature vector Shotton et al. (2006) from each patch, hence every pixel is represented by a texton vector. The node feature for an object is the empirical mean of the texton vector of pixels. The raw node feature is given by $\varphi_i(\mathbf{x}) = [1 \ \varphi_1(\mathbf{x}^i)]$.

φ_2 We use the mean of the boosted texton probability density of all interior and boundary pixels of the objects as their edge feature. The raw edge feature is given by $\varphi_{ij}(\mathbf{x}) = [1 \ \varphi_2(\mathbf{x}^i) \ \varphi_2(\mathbf{x}^j)]$.

As shown in Figure 7.4, the graphical model is very general. Exactly computing CRF gradients involves computing an expectation that is NP hard. The



(a) Raw image

(c) Objects

(d) Features

Figure 7.4: An illustration of the image objects, graph and features. **(a)** The raw hippo image. **(b)** The segmentation result. **(c)** The objects. **(d)** Node and edge features: node feature encodes the object characteristics, while the edge feature encodes the interaction between objects.

common way is to run approximation such as Loopy Belief Propagation (LBP) or sampling. We use LBP for CRF here. All structured algorithms use the same node and edge features. Non-structured algorithms use the node feature only. As shown in Table 7.4, structured algorithms outperform the non-structured ones as expected. And it is interesting to see that structured SVM outperforms CRF. We conjecture that this is because the CRF decision hyperplane becomes less accurate due to the approximated gradient *i.e.*, the expectation of the feature. Whereas structured SVM needs only an argmax operation which is more efficient and perhaps more reliable.

7.4 Conclusion

We have provided theoretical and empirical motivation for the use of a novel hybrid loss for multiclass and structured prediction problems which can be used in place of the more common log loss or multiclass hinge loss. This new loss attempts to blend the strength of purely discriminative approaches to classifica-

tion, such as Support Vector machines, with probabilistic approaches, such as Conditional Random Fields. Theoretically, the hybrid loss enjoys better consistency guarantees than the hinge loss while experimentally we have seen that the addition of a purely discriminative component can improve accuracy when data is less prevalent.

Part IV

Structured Learning Theory

Chapter 8

Structured Learning Theory

Unlike the well developed statistical learning theory in non-structured cases (especially binary classification), the learning theory with structured data is still immature. There is no commonly agreed method of capacity control to explain or guarantee the performance of these algorithms. The generalisation *bounds*—even the recent structured PAC-Bayes bounds—are not yet tight for exponentially many possible labels. The original *Fisher consistency* is too coarse a notion to characterise structured surrogate losses.

In this chapter, we will extend the Fisher consistency to the structured case, and propose a refined notion to characterise the structured surrogate losses. We will review the recent development of PAC-Bayes bounds and give a bound on the generalisation error of a single structured classifier. We will also introduce Probabilistic margins (P-margins) which take the label distribution into account. It turns out that many existing algorithms can be viewed as special cases of P-margins. Hopefully the new alternative concept of margins can help understand existing algorithms as well as design new algorithms.

8.1 Fisher Consistency

Fisher consistency for classification (FCC) is an important property for algorithms, for it tells whether the algorithms yield the best optimal decision boundary given the entire data population. However, the existing FCC is too coarse — it requires that consistency holds for all data distributions. We will propose a more refined notion of Fisher consistency, namely *Conditional Fisher Consistency for Classification* (CFCC), that takes into account the true distribution of class labels. We will show how to examine CFCC and how to compute PAC-Bayes

bounds with an example of a hybrid loss which is CFCC and has a PAC-Bayes bound.

8.1.1 Losses for Structured Prediction

In classification problems *observations* $\mathbf{x} \in \mathcal{X}$ are paired with *labels* $\mathbf{y} \in \mathcal{Y}$ via some joint distribution D over $\mathcal{X} \times \mathcal{Y}$. We will write $D(\mathbf{x}, \mathbf{y})$ for the joint probability and $D(\mathbf{y} \,|\, \mathbf{x})$ for the conditional probability of $\mathbf{y}$ given $\mathbf{x}$. Since the labels $\mathbf{y}$ are finite and discrete we will also use the notation $D_{\mathbf{y}}(\mathbf{x})$ for the conditional probability to emphasise that distributions over $\mathcal{Y}$ can be thought of as vectors in $\mathbb{R}^k$ for $k = |\,\mathcal{Y}\,|$.

When the number of possible labels $k = |\,\mathcal{Y}\,| > 2$ we call the classification problem a *multiclass* classification problem. A special case of this type of problem is *structured prediction* where the set of labels $\mathcal{Y}$ has some combinatorial structure that typically means k is very large (Bakir et al., 2007). As seen in Section 7 a variety of problems, such as text tagging and image categorisation, can be construed as structured prediction problems.

Given m training samples $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$ drawn i.i.d. from D, the aim of the learner is to produce a predictor $h : \mathcal{X} \to \mathcal{Y}$ that minimises the *misclassification error* $R_D(h) = \Pr_D[h(\mathbf{x}) \neq \mathbf{y}]$. Since the true distribution is unknown, an approximate solution to this problem is typically found by minimising a regularised empirical estimate of the risk for a *surrogate loss* ℓ. Examples of surrogate losses will be discussed below.

Once a loss is specified, a solution is found by solving

$$\min_f \frac{1}{m} \sum_{i=1}^{m} \ell(f(\mathbf{x}_i), \mathbf{y}_i) + \Omega(f) \tag{8.1}$$

where each *model* $f : \mathcal{X} \to \mathbb{R}^k$ assigns a vector of scores $f(\mathbf{x})$ to each observation and regulariser $\Omega(f)$ penalises overly complex functions. A model f found in this way can be transformed into a predictor by defining $h(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} f_{\mathbf{y}}(\mathbf{x})$.

In structured prediction, the models are usually specified in terms of a parameter vector $\mathbf{w} \in \mathbb{R}^n$ and a feature map $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$ by defining $f_{\mathbf{y}}(\mathbf{x}; \mathbf{w}) = \langle \mathbf{w}, \phi(\mathbf{x}, \mathbf{y}) \rangle$ and in this case, the regulariser is $\Omega(f) = \frac{\lambda}{2} \| \mathbf{w} \|^2$ for some choice of $\lambda \in \mathbb{R}$. However much of the analysis does not assume any particular parametric model.

A common surrogate loss for multiclass problems is a generalisation of the binary class hinge loss used for Support Vector Machines (Crammer and Singer,

2000):

$$\ell_H(f, \mathbf{y}) = [1 - M(f, \mathbf{y})]_+ \tag{8.2}$$

where $[z]_+ = z$ for $z > 0$ and is 0 otherwise, and $M(f, \mathbf{y}) = f_{\mathbf{y}} - \max_{\mathbf{y}' \neq \mathbf{y}} f'_{\mathbf{y}}$ is the *margin* for the vector $f \in \mathbb{R}^k$. Intuitively, the hinge loss is minimised by models that not only classify observations correctly but also maximise the difference between the highest and second highest scores assigned to the labels.

8.1.2 Conditional Fisher Consistency For Classification

We will say that a vector $f \in \mathbb{R}^{|\mathcal{Y}|}$ is *aligned* with a distribution $D \in \Delta(\mathcal{Y})$ whenever its maximisers are also maximisers for D:

$$\operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} f_{\mathbf{y}} \subseteq \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} D_{\mathbf{y}}.$$

A loss function ℓ is called *Fisher consistent for classification* (FCC) – or *classification calibrated* – if minimising its conditional risk $L(p) = \mathbb{E}_{\mathbf{y} \sim D}[\ell(p, \mathbf{y})]$ yields a vector p^* aligned with D. This is an important property for losses since it is equivalent to the asymptotic consistency of the empirical risk minimiser for that loss (Tewari and Bartlett, 2007, Theorem 2).

The multi-class hinge loss ℓ_H is known to be inconsistent for classification when there are more than two classes (Liu, 2007; Tewari and Bartlett, 2007). The analysis in (Liu, 2007) shows that hinge loss is consistent whenever there is an instance $\mathbf{x}$ with a *non-dominant* distribution – that is, $D_{\mathbf{y}}(\mathbf{x}) < \frac{1}{2}$ for all $\mathbf{y} \in \mathcal{Y}$. Conversely, A distribution is *dominant* for an instance $\mathbf{x}$ if there is some $\mathbf{y}$ with $D_{\mathbf{y}}(\mathbf{x}) > \frac{1}{2}$.

In contrast, the log loss used to train CRFs is Fisher consistent for probability estimation – that is, the associated risk is minimised by the true conditional distribution – and thus ℓ_C is FCC since the minimising distribution is equal to $D(\mathbf{x})$ and thus aligned with $D(\mathbf{x})$.

The existing FCC is too restrictive since it requires the consistency for all D — even for some bizarre D which may never appear in real applications. Hence we introduce a more refined notion of Fisher consistency that takes into account the true distribution of class labels.

Definition 15 (Conditional Fisher Consistency For Classification) *If $D = (D_1, \ldots, D_k)$ is a distribution over the labels $\mathcal{Y}$ then we say the loss ℓ is* conditionally FCC with respect to D *whenever minimising the conditional risk w.r.t. D, $L_D(h) = \mathbb{E}_{\mathbf{y} \sim D}[\ell(h, \mathbf{y})]$ yields a predictor h^* that is consistent with D. Of*

course, if a loss ℓ is conditionally FCC w.r.t. D for all *D it is, by definition, (unconditionally) FCC.*

8.1.3 Conditional Consistency of Hybrid Loss

Recall the hybrid loss for probabilistic models that is a convex combination of the hinge and log losses in (7.4).

Theorem 16 *Let $D = (D_1, \ldots, D_k)$ be a distribution over labels and let $\mathbf{y}_1 = \max_{\mathbf{y}} D_{\mathbf{y}}$ and $\mathbf{y}_2 = \max_{\mathbf{y} \neq \mathbf{y}_1} D_{\mathbf{y}}$ be the two most likely labels. Then the hybrid loss ℓ_α is conditionally FCC for D whenever $D_{\mathbf{y}_1} > \frac{1}{2}$ or*

$$\alpha > 1 - \frac{D_{\mathbf{y}_1} - D_{\mathbf{y}_2}}{1 - 2D_{\mathbf{y}_1}}. \tag{8.3}$$

Theorem 16 can be inverted and interpreted as a constraint on the distribution D such that a hybrid loss with parameter α will yield consistent predictions. Specifically, the hybrid loss will be consistent if, for all $\mathbf{x} \in \mathcal{X}$ such that $D_{\mathbf{y}}(\mathbf{x})$ has no dominant label (*i.e.*, $D_{\mathbf{y}}(\mathbf{x}) \leq \frac{1}{2}$ for all $\mathbf{y} \in \mathcal{Y}$), the separation $D_{\mathbf{y}_1}(\mathbf{x}) - D_{\mathbf{y}_2}(\mathbf{x})$ between the top two probabilities is more than $(1 - \alpha)(1 - 2D_{\mathbf{y}_1}(\mathbf{x}))$. When this is not the case for some $\mathbf{x}$, the classification problem for that instance is, in some sense, too difficult to disambiguate. For the proof see Appendix B.

We expect that some stronger sufficient conditions on α are possible since the bounds used to establish Theorem 16 are not tight. Our conjecture is that a necessary and sufficient condition would include a dependency on the number of classes.

8.2 PAC-Bayes Bounds

The generalisation error

$$e_D = \Pr_{(\mathbf{x},\mathbf{y})\sim D} \left(\mathbf{y} \neq \operatorname*{argmax}_{\mathbf{y}' \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y}'; \mathbf{w}) \right)$$

captures the performance of the algorithm. There are several bounds such as VC bound(Vapnik, 1996), Rademacher bound(Shawe-Taylor and Cristianini, 2004) and so on that upper bound the generalisation error. Among them, PAC-Bayes bounds (McAllester, 1998; Langford et al., 2001; Germain et al., 2008; Zhu and Xing, 2009) are particularly tight.

There are two types of classifiers commonly used in PAC-Bayes bounds : Gibbs classifiers and Average classifiers. Both assume that there exist a prior P over classifiers $h \in \mathcal{H}$. The task for Gibbs classifiers is to choose a posterior Q such that the Q-weighted majority vote classifier (a Gibbs classifier) will have the smallest risk. For h parameterised by $\mathbf{w}$ via F, we simply denote the prior and posterior on $\mathbf{w}$ as $P(\mathbf{w})$ and $Q(\mathbf{w})$ to avoid too much notation. The prediction then becomes

$$G_Q(\mathbf{x}) = \mathbb{E}_{h\sim Q(h)}[h(\mathbf{x})] \tag{8.4}$$

$$= \mathbb{E}_{\mathbf{w}\sim Q(\mathbf{w})}[\operatorname*{argmax}_{\mathbf{y}'\in\mathcal{Y}} F(\mathbf{x},\mathbf{y}';\mathbf{w})]. \tag{8.5}$$

Similarly, an Average classifier predicts by

$$A_Q(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}'\in\mathcal{Y}} \mathbb{E}_{\mathbf{w}\sim Q(\mathbf{w})}[F(\mathbf{x},\mathbf{y}';\mathbf{w})]. \tag{8.6}$$

So the risk for a Gibbs classifier is

$$R(G_Q) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D(\mathbf{x},\mathbf{y})}[G_Q(\mathbf{x}) \neq \mathbf{y}] \tag{8.7}$$

$$= \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D(\mathbf{x},\mathbf{y})}\{\mathbb{E}_{\mathbf{w}\sim Q(\mathbf{w})}[\ell_{0/1}(\mathbf{x},\mathbf{y};\mathbf{w})]\}, \tag{8.8}$$

where the 0/1 loss $\ell_{0/1}(\mathbf{x},\mathbf{y};\mathbf{w}) = \mathbf{1}(\mathbf{y} \neq \operatorname{argmax}_{\mathbf{y}'\in\mathcal{Y}} F(\mathbf{x},\mathbf{y}';\mathbf{w}))$. $\mathbf{1}(s) = 1$ if the statement s is true and 0 otherwise. The risk for an Average classifier is

$$R(A_Q) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D(\mathbf{x},\mathbf{y})}[A_Q(\mathbf{x}) \neq \mathbf{y}] \tag{8.9}$$

$$= \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim D(\mathbf{x},\mathbf{y})}\{\mathbf{y} \neq \operatorname*{argmax}_{\mathbf{y}'\in\mathcal{Y}} \mathbb{E}_{\mathbf{w}\sim Q(\mathbf{w})}[F(\mathbf{x},\mathbf{y}';\mathbf{w})]\}. \tag{8.10}$$

The Gibbs classifier seems to be easier to bound since the two expectations are together. Indeed, the first PAC-Bayes bounds were proposed (McAllester, 1998, 1999) for Gibbs classifiers. Later, it is observed that $R(A_Q) \leq 2R(G_Q)$ (Langford and Shawe-Taylor, 2003; Germain et al., 2008), thus one can focus on $R(G_Q)$ only.

8.2.1 PAC-Bayes Bounds on Gibbs Classifiers

McAllester introduced PAC-Bayes analysis (McAllester, 1998, 1999) which is further refined in McAllester (2001); Langford et al. (2001); Langford (2005); Langford and Shawe-Taylor (2003). Germain et al. (2008) recently give a simplified PAC-Bayesian bound proof on Gibbs classifiers for any convex function $\mathcal{D} : [0,1] \times [0,1] \to \mathbb{R}$. Defining $\mathcal{D}(q,p) = 2(q-p)^2$, one can apply it to any general loss $\ell(\mathbf{x},y;\mathbf{w})$, thus giving the following lemma:

Theorem 17 (PAC-Bayesian bound(McAllester, 2001; Germain et al., 2008)) *For any data distribution D, for any prior P and posterior Q over w, for any $\delta \in (0, 1]$, for any loss ℓ. With probability at least $1 - \delta$ over random sample S from D with m instances, we have*

$$R(Q, \ell) \leq R_S(Q, \ell) + \sqrt{\frac{KL(Q||P) + \ln(\frac{1}{\delta} \mathbb{E}_{s\sim D^m} \mathbb{E}_{\mathbf{w}\sim P} e^{2m(R(Q,\ell) - R_S(Q,\ell))^2})}{2m}},$$

where $KL(Q||P) := \mathbb{E}_{\mathbf{w}\sim Q} \ln(\frac{Q(\mathbf{w})}{P(\mathbf{w})})$ is the Kullback-Leibler divergence between Q and P, and

$$R(Q, \ell) = \mathbb{E}_{Q,D}[\ell(\mathbf{x}, \mathbf{y}; \mathbf{w})], \tag{8.11}$$

$$R_S(Q, \ell) = \mathbb{E}_Q \frac{\sum_{i=1}^m \ell(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w})}{m}. \tag{8.12}$$

$(\mathbb{E}_{s\sim D^m} \mathbb{E}_{\mathbf{w}\sim P} e^{2m(R(Q,\ell) - R_S(Q,\ell))^2})$ is usually upper bounded by a function independent to the data distribution D. For example, for the zero-one loss, it is upper bounded by $m + 1$ (see Germain et al., 2008).

8.2.2 PAC-Bayes bounds on Average Classifiers

Langford et al. (2001) give a margin bound on average classifier for binary classification as follows:

Theorem 18 (Bound on Average Classifier for Binary Classification) *For $\mathcal{Y} = \{-1, 1\}$, for any data distribution D, for any prior P over $\mathbf{w}$, for any $\mathbf{w}$, any $\delta \in (0, 1]$ and for any $\gamma > 0$, with probability at least $1 - \delta$ over random samples S from D with m instances, we have*

$$\Pr_{(\mathbf{x},y)\sim D(\mathbf{x},y)} \left(y\, \mathbb{E}_{h\sim Q}[h(\mathbf{x})] \leq 0 \right) \leq \Pr_{(\mathbf{x},y)\sim S} \left(y\, \mathbb{E}_{h\sim Q}[h(\mathbf{x})] \leq \gamma \right)$$
$$+ O\left(\sqrt{\frac{\gamma^{-2} \frac{|\mathcal{H}|^2}{2} \ln m + \ln m + \ln \delta^{-1}}{m}} \right).$$

Zhu and Xing (2009) later extend it to structured output case for MEDN (see Section 4.4.3 for its definition), which is still an average classifier.

8.2.3 PAC-Bayes Margin bounds

Here we extend existing PAC-Bayes bounds on Averaging classifiers to a single classifier such as SVMs or CRFs in the structured output case. Define $M(\mathbf{w}', \mathbf{y}) = \min_{\mathbf{y}' \neq \mathbf{y}} \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) - \Phi(\mathbf{x}, \mathbf{y}') \rangle$, then the following theorem holds.

Theorem 19 (Generalisation Margin Bound) *For any data distribution D, for any prior P over $\mathbf{w}$, for any $\mathbf{w}$, any $\delta \in (0, 1]$ and for any $\gamma > 0$, with probability at least $1 - \delta$ over random samples S from D with m instances, we have*

$$e_D \leq \Pr_{(\mathbf{x},\mathbf{y})\sim S}(\mathbb{E}_Q(M(\mathbf{w}', \mathbf{y})) \leq \gamma) + O\left(\sqrt{\frac{\gamma^{-2}\frac{\|\mathbf{w}\|^2}{2}\ln(m|\mathcal{Y}|) + \ln m + \ln \delta^{-1}}{m}}\right).$$

For the proof see Appendix B. The big O notation bound decreases as γ increases. However, choosing a large γ will increase the empirical error $\xi = \Pr_{(\mathbf{x},\mathbf{y})\sim S}(\mathbb{E}_Q(M(\mathbf{w}', \mathbf{y})) \leq \gamma)$. Fixing ξ, one can then seek the largest possible γ to tighten the bound.

In fact, the theorem holds not only for the hybrid model, but also for the SVMs and CRFs. Moreover, for a fixed ξ, the largest possible γ in the hybrid model implicitly depends on α. This is because SVMs purely maximise the margin γ whereas the hybrid model tries to balance a large margin and small log loss according to α. For CRFs, the bound can be meaningless since γ is not maximised at all. Using the FCC analysis, one can always find the smallest α to ensure the conditionally FCC on the hybrid model. Applying the theorem gives a non-trivial generalisation bound on the hybrid model. This way, the hybrid model has strengths of both CRFs and SVMs.

8.3 Probabilistic Margins

Traditional large margin algorithms have a *geometrical* interpretation — the hyperplane separates the correctly and incorrectly labeled data in the feature space by a large margin. However, the existing margins don't take into account the probability distribution of the labels. One may argue that if you want to do classification, you don't need to waste your computational power on modelling the distribution. However, if we have some knowledge of the label distribution, taking it into account may help design more robust learning algorithms.

8.3.1 Geometrical Margins

Crammer and Singer give the definition of SVM margins for multiclass classification (Crammer and Singer, 2001), which has been further generalised in struc-

tured label case (Taskar et al., 2004; Tsochantaridis et al., 2004). It is known that the margin has a *geometrical* interpretation — the hyperplane (*i.e.* the discriminant function F) separates the correctly and incorrectly labeled data in the feature space by a large margin. Hence we call it geometrical margin. For example, the hard margin SVMs is defined as

$$\max_{\gamma} \ \gamma \quad \text{s.t.} \tag{8.13a}$$

$$\| \mathbf{w} \| = 1 \tag{8.13b}$$

$$\forall i, \mathbf{y} \ \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \mathbf{y}) \rangle \geq \gamma, \tag{8.13c}$$

where $\gamma \geq 0$, enforces the separability of input-output pairs. To allow outliers, the soft margin constraint is defined as

$$\max_{\gamma} \ \gamma - C \sum_i \xi_i \quad \text{s.t.} \tag{8.14a}$$

$$\| \mathbf{w} \| = 1 \tag{8.14b}$$

$$\forall i, \mathbf{y} \ \langle \mathbf{w}, \Phi(\mathbf{x}_i, \mathbf{y}_i) - \Phi(\mathbf{x}_i, \mathbf{y}) \rangle \geq \gamma - \xi_i. \tag{8.14c}$$

8.3.2 Probabilistic Margins

As we shall see, these margin constraints haven't made use of any information about how the data is distributed. It is natural to think what we will gain by making use of the (approximated) distribution of the data $P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x})$. We call any margins represented in terms of $P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x})$ Probabilistic Margins. For simplicity, we use $p(\mathbf{y})$ to express $P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x})$ when the context of $\mathbf{x}$ is clear.

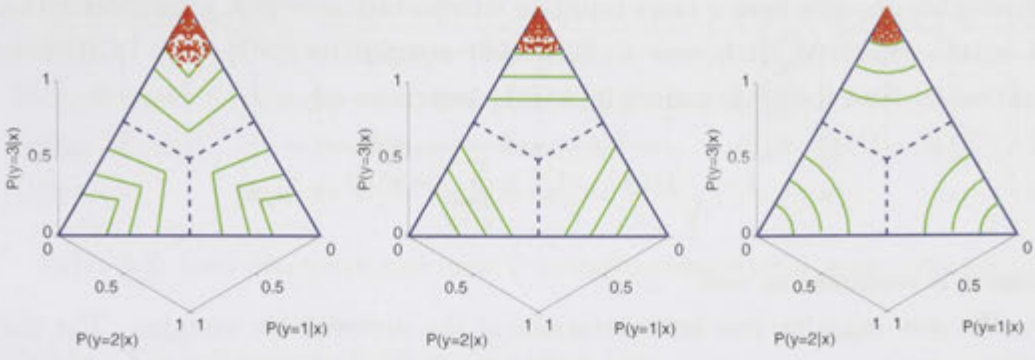
Definition 20 (Feasible sets and P-mapping) *The smallest feasible set for any* $\mathbf{y} \in \mathcal{Y}$ *is* $\{p : p(\mathbf{y}) = 1, p \in \Delta(\mathcal{Y})\}$, i.e. *the corresponding corner of the simplex. We denote it as* $\mathcal{M}_\infty(\mathbf{y})$. *A convex set* $\mathcal{M}_\gamma(\mathbf{y})$ *is a* feasible set *if and only if there exists* $\prod_\gamma : \Delta(\mathcal{Y}) \times \mathcal{Y} \to \mathcal{M}_\gamma(\mathbf{y}) \ \forall \gamma \in \mathbb{R}$, *satisfies:*

(Monotonic decrease) $\mathcal{M}_{\gamma_1}(\mathbf{y}) \supset \mathcal{M}_{\gamma_2}(\mathbf{y})$ *if and only if* $\gamma_1 < \gamma_2$; (8.15)

(Convexity) $\mathcal{M}_\gamma(\mathbf{y})$ *is still convex.* (8.16)

Such a $\prod_\gamma$ *is called P-mapping. We also define*

$$\mathcal{M}_\gamma = \bigcup_{\mathbf{y} \in \mathcal{Y}} \mathcal{M}_\gamma(\mathbf{y}) \tag{8.17}$$



(a) SVMs (b) CRFs (c) cycle

Figure 8.1: Contours of P-margins feasible sets. The green lines are the boundary of $\mathcal{M}_\gamma$ on the simplex on selected γ values. The region with red dots is $\mathcal{M}_\gamma(\mathbf{y}=3)$ with the largest γ value. 3 classes are used for demonstration purposes, although nothing prevents us from using the structured case. (a) SVMs margin. That is $\ln \frac{p(\mathbf{y})}{p(\mathbf{y}^*)} > \gamma$. $\gamma = 0.5, 1, 1.5$. (b) Lower bounded CRFs margin. That is $\ln p(\mathbf{y}) > \gamma$. $\gamma = \ln(0.6), \ln(0.7), \ln(0.8)$. (c) Cycle constraint margin. That is $(p(\mathbf{y})-1)^2 + \sum_{y' \neq y}(p(\mathbf{y}')^2) < (1-\gamma)^2, \gamma = 0.5, 0.65, 0.8$. Red dots are p points sampled in the feasible set. Due to symmetry, we only sample from the right half region of M_γ, and then display sample points symmetrically on both sides.

Intuitively, P-margins can be viewed as getting different contours of the feasible set via choosing different $\mathcal{M}_\gamma$. Thus increasing γ, the feasible set shrinks differently given different $\prod$. On the other hand, one can come up with any contour of the feasible set, as long as the Monotonic decrease and Convexity hold, it is a valid P-margin. This is very convenient.

Some P-margins are shown in Fig. 8.1. For example Fig. 8.1a shows that the contours of SVMs margin feasible sets are parallel to blue bisector lines. Whereas for Lower Bounded CRFs (LCRFs), the contours are straight lines (see Fig. 8.1b). What if we want the contours to be cycles centred at each corner as Fig. 8.1c shows? It turns out that following constraint $(p(\mathbf{y})-1)^2+\sum_{\mathbf{y}' \neq \mathbf{y}}(p(\mathbf{y}')^2) < (1-\gamma)^2$ gives exactly what we want.

Definition 21 (Realizable P-margins) *A functional $\mu : \Delta(\mathcal{Y}) \times \mathcal{Y} \to \mathbb{R}$ is called a* realizable P-margin *if and only if μ satisfies:*

$$\forall \gamma \in \mathbb{R},\ \{p : p \in \Delta(\mathcal{Y}), \mu(p, \mathbf{y}) > \gamma\} = \mathcal{M}_\gamma(\mathbf{y}).$$

Realizable margins have a more intuitive interpretation — $\mu(p, \mathbf{y})$ is greater than a certain threshold. It is easy to show that margins for SVMs and LCRFS are realisable. And the cycle constraint can be rewritten as

$$1 - \sqrt{(p(\mathbf{y}) - 1)^2 + \sum_{\mathbf{y}' \neq \mathbf{y}} (p(\mathbf{y}')^2)} > \gamma,$$

thus it is realisable as well.

We now examine two interpretations of the probabilistic margins. The first relates the p-margin to losses for probability estimation while the second gives a geometrical interpretation in terms of restrictions of models on probability simplexes. This second interpretation provides an intuition as to how the margins act as a capacity control, in a similar way to the way the original, geometric margin does for classification.

8.3.3 Losses imply P-Margins

Many existing algorithm can then be cast as

$$\max_p \gamma, \quad s.t. \tag{8.18a}$$

$$\forall i, p \in \mathcal{M}_\gamma(\mathbf{y}_i). \tag{8.18b}$$

To allow outliers, a soft margin version is obtained by a relaxed constraint

$$\max_p \gamma - C \sum_i \xi_i, \quad s.t. \tag{8.19a}$$

$$\forall i, p \in \mathcal{M}_{\gamma - \xi_i}(\mathbf{y}_i), \xi_i > 0. \tag{8.19b}$$

For realizable margins, the constraints become

$$\max_p \gamma - C \sum_i \xi_i, \quad s.t. \tag{8.20a}$$

$$\forall i, \mu(p, \mathbf{y}_i) \geq \gamma - \xi_i, \xi_i > 0. \tag{8.20b}$$

Alternatively, it can be written as

$$\min_p J(p) := \lambda \Omega(p) + \sum_{i=1}^{m} \xi_i, \quad s.t. \tag{8.21a}$$

$$\forall i, \mu(p, \mathbf{y}_i) \geq \gamma_0 - \xi_i, \xi_i > 0, \tag{8.21b}$$

where γ_0 is a fixed constant.

Algorithms	Losses	$\mu(p, \mathbf{y})$	γ_0
Hinge	$[1 - \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) - \Phi(\mathbf{x}, \overline{\mathbf{y}}) \rangle]_+$	$\ln \frac{p(\mathbf{y})}{p(\overline{\mathbf{y}})}$	1
HingeRescale	$[\Delta(\mathbf{y}, \overline{\mathbf{y}}) - \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) - \Phi(\mathbf{x}, \overline{\mathbf{y}}) \rangle]_+$	$\ln \frac{p(\mathbf{y})}{p(\overline{\mathbf{y}})}$	$\Delta(\mathbf{y}, \overline{\mathbf{y}})$
SquaredHinge	$\frac{1}{2}(1 - \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) - \Phi(\mathbf{x}, \overline{\mathbf{y}}) \rangle)^2$	$-\frac{1}{2}(\ln \frac{p(\mathbf{y})}{p(\overline{\mathbf{y}})})^2 + \ln \frac{p(\mathbf{y})}{p(\overline{\mathbf{y}})}$	$\frac{1}{2}$
CRF	$\langle \Phi(\mathbf{x}, \mathbf{y}), \mathbf{w} \rangle - \ln(Z(\mathbf{w} \mid \mathbf{x}))$	$-\ln(p(\mathbf{y}))$	0

Table 8.1: Loss functions and their P-margins constraint $\mu(p, y) \geq \gamma_0 - \xi$.

SVMs The soft margin SVM has the hinge loss

$$\begin{aligned} \ell_h(\mathbf{x}, \mathbf{y}, \mathbf{w}) &= [1 - \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) - \Phi(\mathbf{x}, \overline{\mathbf{y}}) \rangle]_+ \\ &= [1 - \ln \frac{p(\mathbf{y})}{p(\overline{\mathbf{y}})}]_+. \end{aligned}$$

So $\mu(p, \mathbf{y}) = \ln \frac{p(\mathbf{y})}{p(\overline{\mathbf{y}})}$ and $\gamma_0 = 1$.

CRFs CRFs can be also formulated as

$$\operatorname*{argmin}_{\mathbf{w}} \frac{\| \mathbf{w} \|^2}{2\sigma^2} + \sum_{i=1}^{m} \xi_i \tag{8.22}$$

$$s.t. \quad \ln(p(\mathbf{y}_i)) \geq -\xi_i, \quad \xi_i \geq 0, \quad \forall i \tag{8.23}$$

So $\mu(p, \mathbf{y}) = \ln(p(\mathbf{y}_i))$ and $\gamma_0 = 0$.

For more examples of P-margins for various algorithms, see Table 8.1.

8.4 Conclusion

We extended the Fisher consistency to the structured case, and proposed a refined notion to characterise the structured surrogate losses. We reviewed the recent development of PAC-Bayes bounds and gave a bound on the generalisation error of a single structured classifier. We also introduce P-margins which take the label distribution into account. And we show that many existing algorithms can be viewed as special cases of the new margin concept which may help understand existing algorithms as well as design new algorithms.

Part V

Conclusions and Future Directions

Part V

Conclusions and Future Directions

Chapter 9

Summary and Future Directions

In this chapter, we will summarise our contributions and discuss future directions.

9.1 Contribution Summary

In this thesis, we made several contributions — some are complete, and some are at the exploratory stage. We propose hash kernels (in Chapter 2) to facilitate efficient kernels (Shi et al., 2009a,b) which can deal with massive multi-class problems with even more than 7000 classes. We exploit the connection between hash kernels and compressed sensing, and apply hashing to face recognition which significantly speeds up the state-of-the-art (Shi et al., 2010a) (in Chapter 3). We propose a novel approach for automatic paragraph segmentation (Shi et al., 2007) (in Chapter 5), namely training Semi-Markov models discriminatively using a Max-Margin method. This method allows us to model the sequential nature of the problem and to incorporate features of a whole paragraph. We jointly segment and recognise actions in video sequences with a discriminative semi-Markov model framework (Shi et al., 2008, 2009d) (in Chapter 6). A Viterbi-like algorithm is devised to help efficiently solve the induced optimisation problem. We propose a novel hybrid loss (Shi et al., 2009c, 2010b) (in Chapter 7) which has the advantages of both CRFs and SVMs — it is consistent and has a tight PAC-Bayes bound. We apply it to various applications such as Text chunking, Named Entity Recognition and Joint Image Categorisation. We study the recent advances in PAC-Bayes bounds, and apply them to structured learning (Shi et al., 2009c, 2010b) (in Chapter 8). Moreover, we propose a more refined notion of Fisher consistency, namely *Conditional Fisher Consistency for Classification* (CFCC)(Shi et al., 2010b), that conditions on the knowledge of the true distribution of class

labels. It turns out that the hybrid loss is CFCC but not FCC. We also introduce Probabilistic margins (in Chapter 8) which take the label distribution into account. And we show that many existing algorithms can be viewed as special cases of the new margin concept which may help understand existing algorithms as well as design new algorithms.

9.2 Future Directions

9.2.1 Tightening PAC-Bayes bounds

PAC-Bayes bounds incorporate both data and the distributions of models. However, those existing bounds are often pessimistic and loose. It may be possible to develop tighter bounds using more information exacted from the data. For example, the true conditional distribution of a label for an observation is often on few labels. Hence the large quantity of the size of the label space in the PAC-Bayes bound might be replaced by some much smaller quantity which may lead to tighter bound.

9.2.2 Adaptive hybrid loss

The current hybrid model uses a single, fixed α for each training set. One interesting avenue to explore would be trying to dynamically estimate a good value of α on a per-observation basis. This may further improve the efficacy of the hybrid loss by exploiting the robustness of SVMs (low α) when the label distribution for an observation has a dominant class but switching to probability estimation via CRFs (high α) when this is not the case.

9.2.3 Compressed Sensing and Graphical model inference

Inference for large graphical models is often extremely expensive. For instance, a 1000 by 1000 pixels image can have a graphical model with 1 million nodes. Even approximate inference algorithms become very expensive. How to infer and learn models as such scales is still an open question.

Compressed Sensing (CS) (Candes and Tao, 2005; Candés et al., 2006; Donoho, 2006; Tropp and Gilbert, 2007; Song et al., 2008) in the context of information theory and signal processing discovered a surprising result — a sparse signal can be recovered by a sampling rate much smaller than the conventional Shannon-Nyquist rate. Hsu et al. (2009) apply CS to multi-label prediction problems with

large output spaces under the assumption of output sparsity. They learn a regressor from which one can predict a compressed label which is then used to recover the original label. However, their technique can not deal with dependent labels in graphical models. Cevher et al. (2008) infer a restricted Markov Random Field with only binary classes on each node. Moreover, they *implicitly* assume that the expected feature is the same as or similar to the expected label values, so that they can avoid designing potential functions in the compressed space.

We are interested in applying CS to more general graphical models. Compared to multi-class or multi-label, graphical models are already compact representations. The notion of sparsity in CS may no longer be applicable to general graphical models. We would instead like to define a novel notion — *compressibility* instead on graphical models. By random mapping or mapping according to clustering/separability, we may be able to map the original large graphical models to much smaller graphical models with careful design of potential functions which guarantee the consistency between the inference in the original graphical models and that in the compressed ones. The task of inference is to find the most likely labels which are often grouped or separated into subspaces. Exploiting the correlation between subspaces such as blocked/grouped ℓ_1 regularisation (Stojnic, 2009) or more general model-based CS (Baraniuk et al., 2009) is a possible way to recover the original best label from the compressed best label. This way, one could infer a large graphical model with millions of nodes with the cost of only thousands of nodes.

Appendix A

Appendix

This appendix contains proofs for hash kernels and compressive sensing.

Theorem 2 For a random function mapping l features duplicated c times into a space of size n, for all loss functions L and distributions D on n features, the probability (over the random function) of no information loss is at least:

$$1 - l[1 - (1 - c/n)^c + (lc/n)^c].$$

Proof The proof is essentially a counting argument with consideration of the fact that we are dealing with a hash *function* rather than a random variable. It is structurally similar to the proof for a Bloom filter (Bloom, 1970), because the essential question we address is: "What is a lower bound on the probability that all features have one duplicate not colliding with any other feature?"

Fix a feature f. We'll argue about the probability that all c duplicates of f collide with other features.

For feature duplicate i, let $h_i = h(f \circ i)$. The probability that $h_i = h(f' \circ i')$ for some other feature $f' \circ i'$ is bounded by $(l-1)c/n$ because the probability for each other mapping of a collision is $1/n$ by the assumption that h is a random function, and the union bound applied to the $(l-1)c$ mappings of other features yields $(l-1)c/n$. Note that we do not care about a collision of two duplicates of the same feature, because the feature value is preserved.

The probability that all duplicates $1 \leq i \leq c$ collide with another feature is bounded by $(lc/n)^c + 1 - (1 - c/n)^c$. To see this, let $c' \leq c$ be the number of distinct duplicates of f after collisions. The probability of a collision with the first of these is bounded by $\frac{(l-1)c}{n}$. Conditioned on this collision, the probability of the

next collision is at most $\frac{(l-1)c-1}{n-1}$, where 1 is subtracted because the first location is fixed. Similarly, for the ith duplicate, the probability is $\frac{(l-1)c-(i-1)}{n-(i-1)}$. We can upper bound each term as $\frac{lc}{n}$, implying the probability of all c' duplicates colliding with other features is at most $(lc/n)^{c'}$. The probability that $c' = c$ is the probability that none of the duplicates of f collide, which is $\frac{(n-1)!}{n^c(n-c-1)!} \geq ((n-c)/n)^c$. If we pessimistically assume that $c' < c$ implies that every duplicate collides with another feature, then

$$\begin{aligned}\mathbf{P}(\text{coll}) &\leq \mathbf{P}(\text{coll}|c' = c)\,\mathbf{P}(c' = c) + \mathbf{P}(c' \neq c) \\ &\leq (lc/n)^c + 1 - ((l-c)/l)^c.\end{aligned}$$

Simplification gives $(lc/n)^c + 1 - (1 - c/n)^c$ as claimed. Taking a union bound over all l features, we find that the probability any feature has all duplicates collide is bounded by $l[1 - (1 - c/n)^c + (lc/n)^c]$. ■

Theorem 3 Assume that the probability of deviation between the hash kernel and its expected value is bounded by an exponential inequality via

$$\mathbf{P}\left[\left|\overline{k}^h(\mathbf{x}, \mathbf{x}') - \mathbb{E}_h\left[\overline{k}^h(\mathbf{x}, \mathbf{x}')\right]\right| > \epsilon\right] \leq c\exp(-c'\epsilon^2 n)$$

for some constants c, c' depending on the size of the hash and the kernel used. In this case the error ϵ arising from ensuring the above inequality, with probability at least $1 - \delta$, for m observations and M classes for a joint feature map $\Phi(\mathbf{x}, y)$, is bounded by

$$\epsilon \leq \sqrt{(2\log(m+1) + 2\log(M+1) - \log\delta + \log c - 2\log 2)/nc'}. \tag{A.1}$$

Proof Apply the union bound to the kernel matrix of size $(mM)^2$, that is, to all $T := m(m+1)M(M+1)/4$ unique elements. Solving

$$Tc\exp(-c'\epsilon^2 n) = \delta,$$

we get the bound

$$\epsilon \leq \sqrt{\frac{\log(Tc) - \log\delta}{c'n}}.$$

Bounding $\log(Tc)$ from above

$$\log(Tc) = \log T + \log c \leq 2\log(m+1) + 2\log(M+1) + \log c - 2\log 2,$$

and substituting it into (A.1) yields the result. ■

Corollary 10 [Recovery on a Specific Basis] For any η-sparse signal $\boldsymbol{\alpha} \in \mathbb{R}^n$ and two constants $z_1, z_2 > 0$, let $d \geq z_1 \eta \log(n/\eta)$, and draw d row vectors $\mathbf{r}_1, \ldots, \mathbf{r}_d$ independently from the standard Gaussian distribution on $\mathbb{R}^m$. Denote the stacked vectors $\{\mathbf{r}_i\}_{i=1}^d$ as the matrix $\mathbf{R} \in \mathbb{R}^{d,m}$. For any matrix $\mathbf{A} \in \mathbb{R}^{m,n}$ with unit length columns, with probability at least $1 - e^{-z_2 d}$, the signal α can be recovered via

$$\alpha^* = \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\operatorname{argmin}} \|\mathbf{R}\,\mathbf{x} - (\mathbf{R}\,\mathbf{A})\boldsymbol{\alpha}\|_{\ell_2}^2 + \lambda \|\boldsymbol{\alpha}\|_{\ell_1}.$$

Proof Let $\mathbf{A}_j, j = 1, \ldots, n$ denote the j-th column vector of the matrix $\mathbf{A}$ and let $\widetilde{\mathbf{A}} := \mathbf{R}\,\mathbf{A}$, *i.e.*, the row vectors $\widetilde{\mathbf{A}}_i = \big(\langle \mathbf{r}_i, \mathbf{A}_1 \rangle, \ldots, \langle \mathbf{r}_i, \mathbf{A}_n \rangle \big)$ for $(i = 1, \ldots, d)$. Note that the inner product $\langle \mathbf{r}_i, \mathbf{A}_j \rangle = \sum_{k=1}^m \mathbf{r}_{i,k}\, \mathbf{A}_{k,j}$ is still a random variable drawn from Gaussian distribution $N(0, \sum_{k=1}^m \mathbf{A}_{k,j}^2)$. Hence $\{\widetilde{\mathbf{A}}_i\}_{i=1}^d$ are random vectors independently drawn from the Gaussian distribution in $\mathbb{R}^m$. Corollary 10 follows Theorem 9. ■

Theorem 12 [Hashing OMP Recovery] For any η-sparse signal $\boldsymbol{\alpha} \in \mathbb{R}^n$ and confidence $\delta > 0$, given hash matrix $\mathbf{H}$, let $d \geq 16\eta^2 \log(n/\delta)$, for any matrix $\mathbf{A} \in \mathbb{R}^{m,n}$, take the measurements such that $\mathbf{H}\,\mathbf{x} = (\mathbf{H}\,\mathbf{A})\boldsymbol{\alpha}$. Then with probability at least $1 - \delta$, the signal α can be recovered via Algorithm 2.

Proof Admissibility mainly relies on the coherence statistic $\mu := \max_{j<k} |\langle \mathbf{R}_j, \mathbf{R}_k \rangle|$. In a hash matrix $\mathbf{H}$, $\{-1, 1\}$ are equally likely to appear so $\mathbb{E}[\langle \mathbf{H}_j, \mathbf{H}_k \rangle] = 0$. By the hoeffding inequality, $P(|\langle \mathbf{H}_j, \mathbf{H}_k \rangle > \epsilon|) \leq 2e^{-\epsilon N/2}$. The union bound argument further gives the bound on $P(\mu) < d$ of $\mathbf{H}$ as of Bernoulli random matrix. This then leads to the same bound on the smallest singular value. Also we know the columns of $\mathbf{H}$ with multiple hash functions are independent and normalization only changes the scale, hence $\mathbf{H}$ is admissible. Admissibility implies reconstruction, so the theorem holds. ■

Appendix B

Appendix

This appendix contains proofs for structured learning.

Theorem 16 Let $D = (D_1, \ldots, D_k)$ be a distribution over labels and let $\mathbf{y}_1 = \max_{\mathbf{y}} D_{\mathbf{y}}$ and $\mathbf{y}_2 = \max_{\mathbf{y} \neq \mathbf{y}_1} D_{\mathbf{y}}$ be the two most likely labels. Then the hybrid loss ℓ_α is conditionally FCC for D whenever $D_{\mathbf{y}_1} > \frac{1}{2}$ or

$$\alpha > 1 - \frac{D_{\mathbf{y}_1} - D_{\mathbf{y}_2}}{1 - 2D_{\mathbf{y}_1}}. \tag{B.1}$$

Proof Since we a free to permute labels within $\mathcal{Y}$ we will assume without loss of generality that $D_1 = \max_{\mathbf{y} \in \mathcal{Y}} D_{\mathbf{y}}$ and $D_2 = \max_{\mathbf{y} \neq 1} D_{\mathbf{y}}$. The proof now proceeds by contradiction and assumes there is some minimiser $p = \operatorname{argmin}_{q \in \Delta(\mathcal{Y})} L_\alpha(q, D)$ that is not aligned with D. That is, there is some $\mathbf{y}^* \neq 1$ such that $p_{\mathbf{y}^*} \geq p_1$. For simplicity, and again without loss of generality, we will assume $\mathbf{y}^* = 2$.

The first case to consider is when p_2 is a maximum and $p_1 < p_2$. Here we construct a q that "flips" the values of p_1 and p_2 and leaves all the values unchanged. That is, $q_1 = p_2$, $q_2 = p_1$ and $q_{\mathbf{y}} = p_{\mathbf{y}}$ for all $\mathbf{y} = 3, \ldots, k$. Intuitively, this new point is closer to D and therefore the CRF component of the loss will be reduced while the SVM loss won't increase. The difference in conditional risks satisfies

$$\begin{aligned} L_\alpha(p, D) - L_\alpha(q, D) &= \sum_{\mathbf{y}=1}^{k} D_{\mathbf{y}}.(\ell_\alpha(p, \mathbf{y}) - \ell_\alpha(q, \mathbf{y})) \\ &= D_1.(\ell_\alpha(p, 1) - \ell_\alpha(q, 1)) \\ &\quad + D_2.(\ell_\alpha(p, 2) - \ell_\alpha(q, 2)) \\ &= (D_1 - D_2)(\ell_\alpha(q, 2) - \ell_\alpha(q, 1)) \end{aligned}$$

since $\ell_\alpha(p,1) = \ell_\alpha(q,2)$ and $\ell_\alpha(p,2) = \ell_\alpha(q,1)$ and the other terms cancel by construction. As $D_1 - D_2 > 0$ by assumption, all that is required now is to show that $\ell_\alpha(q,2) - \ell_\alpha(q,1) = \alpha \ln \frac{q_1}{q_2} + (1-\alpha)(\ell_H(q,2) - \ell_H(q,1))$ is strictly positive.

Since $q_1 > q_{\mathbf{y}}$ for $\mathbf{y} \neq 1$ we have $\ln \frac{q_1}{q_2} > 0$, $\ell_H(q,2) = \left[1 - \ln \frac{q_2}{q_1}\right]_+ > 1$, and $\ell_H(q,1) = \left[1 - \ln \frac{q_1}{q_{\mathbf{y}}}\right]_+ < 1$, and so $\ell_H(q,2) - \ell_H(q,1) > 1 - 1 = 0$. Thus, $\ell_\alpha(q,2) - \ell_\alpha(q,1) > 0$ as required.

Now suppose that $p_2 = p_1$ is a maximum. In this case we show a slight perturbation $q = (p_1 + \epsilon, p_2 - \epsilon, p_3, \ldots, p_k)$ yields a lower for $\epsilon > 0$. For $\mathbf{y} \neq 1, 2$ we have $\ell_L(p,\mathbf{y}) - \ell(q,\mathbf{y}) = 0$ and since $p_2 > p_{\mathbf{y}}$ and $q_1 > q_{\mathbf{y}}$ thus $\ell_H(p,\mathbf{y}) - \ell_H(q,\mathbf{y}) = 1 - \ln \frac{p_{\mathbf{y}}}{p_2} + 1 - \ln \frac{q_{\mathbf{y}}}{q_1} = \ln \frac{p_2}{q_1} > 1 - \frac{q_1}{p_2} = -\frac{\epsilon}{p_2}$ since $-\ln x > 1 - x$ for $x \in (0,1)$ and $q_1 = p_1 + \epsilon = p_2 + \epsilon$. Therefore

$$\ell_\alpha(p,\mathbf{y}) - \ell_\alpha(q,\mathbf{y}) > -\epsilon \frac{(1-\alpha)}{p_1} \tag{B.2}$$

When $\mathbf{y} = 1$, $\ell_L(p,1) - \ell_L(q,1) = -\ln \frac{p_1}{q_1} > \frac{q_1 - p_1}{p_1} = \frac{\epsilon}{p_1}$ and $\ell_H(p,1) - \ell_H(q,1) = (1 - \ln \frac{p_1}{p_2}) - (1 - \ln \frac{q_1}{q_2}) = \ln \frac{q_1}{q_2} = \ln \frac{p_1 + \epsilon}{p_1 - \epsilon}$ since $p_1 = p_2$. Thus $\ell_H(p,1) - \ell_H(q,1) > 1 - \frac{p_1 - \epsilon}{p_1 + \epsilon} = \frac{2\epsilon}{p_1 + \epsilon}$. And so

$$\ell_\alpha(p,\mathbf{y}) - \ell_\alpha(q,\mathbf{y}) > \epsilon \left[\frac{\alpha}{p_1} + \frac{2(1-\alpha)}{p_1 + \epsilon}\right] \tag{B.3}$$

Finally, when $\mathbf{y} = 2$ we have $\ell_L(p,2) - \ell_L(q,2) = -\ln \frac{p_2}{q_2} > \frac{q_2 - p_2}{p_2} = \frac{-\epsilon}{p_1}$ and $\ell_H(p,2) - \ell_H(q,2) = (1 - \ln \frac{p_2}{p_1}) - (1 - \ln \frac{q_2}{q_1}) = \ln \frac{q_2}{q_1} > 1 - \frac{q_1}{q_2} = \frac{-2\epsilon}{p_1 + \epsilon}$. Thus,

$$\ell_\alpha(p,2) - \ell_\alpha(q,2) > -\epsilon \left[\frac{\alpha}{p_1} + \frac{2(1-\alpha)}{p_1 + \epsilon}\right]. \tag{B.4}$$

Putting the inequalities (B.2), (B.3) and (B.4) together yields

$$\begin{aligned}
&\lim_{\epsilon \to 0} \frac{L_\alpha(p,D) - L_\alpha(q,D)}{\epsilon} \\
&> \lim_{\epsilon \to 0} (D_1 - D_2) \left[\frac{\alpha}{p_1} + \frac{2(1-\alpha)}{p_1 + \epsilon}\right] - \sum_{\mathbf{y}=3}^{k} D_{\mathbf{y}} \frac{1-\alpha}{p_1} \\
&= \frac{D_1 - D_2}{p_1}(2 - \alpha) - \frac{1 - D_1 - D_2}{p_1}(1 - \alpha) \\
&= \frac{1}{p_1}(D_1 - D_2 + (1-\alpha)(2D_1 - 1)).
\end{aligned}$$

Observing that since $D_1 > D_2$, when $D_1 > \frac{1}{2}$ the final term is positive without any constraint on α and when $D_1 < \frac{1}{2}$ the difference in risks is positive whenever

$$\alpha > 1 - \frac{D_1 - D_2}{1 - 2D_1} \tag{B.5}$$

completes the proof. ■

Theorem 19 [Generalisation Margin Bound] For any data distribution D, for any prior P over $\mathbf{w}$, for any $\mathbf{w}$, any $\delta \in (0, 1]$ and for any $\gamma > 0$, with probability at least $1 - \delta$ over random samples S from D with m instances, we have

$$e_D \leq \Pr_{(\mathbf{x},\mathbf{y})\sim S}(\mathbb{E}_Q(M(\mathbf{w}', \mathbf{y})) \leq \gamma) + O\left(\sqrt{\frac{\gamma^{-2}\frac{\|\mathbf{w}\|^2}{2}\ln(m|\mathcal{Y}|) + \ln m + \ln \delta^{-1}}{m}}\right).$$

Proof By choosing the weight prior $P(\mathbf{w}) = \frac{1}{Z}\exp(-\frac{\|\mathbf{w}\|^2}{2})$ and the posterior $Q(\mathbf{w}') = \frac{1}{Z}\exp(-\frac{\|\mathbf{w}'-\mathbf{w}\|^2}{2})$, one can show $e_D = \Pr_D(\mathbb{E}_Q M(\mathbf{w}', \mathbf{y}) \leq 0)$ by symmetry argument proposed in Langford et al. (2001); McAllester (2007). Applying the PAC-Bayes margin bound Langford et al. (2001); Zhu and Xing (2009) and using the fact that $\mathrm{KL}(Q||P) = \frac{\|\mathbf{w}\|^2}{2}$ yields the theorem. ■

Bibliography

D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci*, 66(4):671–687, 2003.

Y. Altun and A. Smola. Unifying divergence minimization and statistical inference via convex duality. In H. Simon and G. Lugosi, editors, *Proc. Annual Conf. Computational Learning Theory*, LNCS, pages 139–153. Springer, 2006.

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *Proc. Intl. Conf. Machine Learning*, pages 3–10, Menlo Park, California, 2003. AAAI Press.

A. Appleby. Murmurhash, 2008. `http://sites.google.com/site/murmurhash/`.

G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data*. MIT Press, Cambridge, Massachusetts, 2007.

R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde. Model-based compressive sensing. In *arxiv.org*, 2009.

R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin. A simple proof of the restricted isometry principle for random matrices. *Constructive Approximation*, 2007.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis and Machine Intell.*, 24 (4):509–522, 2002.

C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer, New York, 1984.

C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422C426, July 1970.

K. M. Borgwardt, H.-P. Kriegel, S. V. N. Vishwanathan, and N. Schraudolph. Graph kernels for disease outcome prediction from protein-protein interaction networks. In R. B. Altman, A. K. Dunker, L. Hunter, T. Murray, and T. E. Klein, editors, *Proceedings of the Pacific Symposium of Biocomputing 2007*, Maui Hawaii, January 2007. World Scientific.

Y. Bulatov and O. Bousquet. Log loss or hinge loss, 2007. `http://yaroslavvb.blogspot.com/2007/06/log-loss-or-hinge-loss.html`.

C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381, Cambridge, MA, 1997. MIT Press.

E. Candés. The restricted isometry property and its implications for compressed sensing. *C. R. Acad. Sci. Paris, Ser. I*, 346:589–592, 2008.

E. Candés, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Information Theory*, 52(2):489–509, 2006.

E. Candes and T. Tao. Decoding by linear programming. *IEEE Trans. Info Theory*, 51(12):4203–4215, 2005.

V. Cevher, M. F. Duarte, C. Hegde, and R. G. Baraniuk. Sparse signal recovery using markov random fields. In *Twenty-Two Annual Conference on Neural Information Processing Systems*, CanadaA, 2008.

L. Cheng, S. Wang, D. Schuurmans, T. Caelli, and S. Vishwanathan. An online discriminative approach to background subtraction. In *IEEE international conference on advanced video and signal based surveillance (AVSS)*, 2006.

M. Collins. Discriminative training methods for hidden Markov models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002.

G. Cormode and M. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *LATIN: Latin American Symposium on Theoretical Informatics*, 2004.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In N. Cesa-Bianchi and S. Goldman, editors, *Proc. Annual Conf. Computational Learning Theory*, pages 35–46, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, 2001.

A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *J Med Chem*, 34:786–797, 1991.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: A kernel-based perceptron on a fixed budget. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

P. D. Dobson and A. J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *J Mol Biol*, 330(4):771–783, Jul 2003.

P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS workshop*, 2005.

D. L. Donoho. Compressed sensing. *IEEE Trans. Information Theory*, 52(4): 1289–1306, 2006.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *JMLR*, 2:243–264, Dec 2001.

C. Galleguillos, A. Rabinovich, and S. Belongie. Object categorization using co-occurrence, location and appearance. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008*. MIT Press, 2008.

K. Ganchev and M. Dredze. Small statistical models by random feature mixing. In *workshop on Mobile NLP at ACL*, 2008.

D. Genzel. A paragraph boundary detection system. In *Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing'05)*, pages 825–830, 2005.

D. Genzel and E. Charniak. Variation of entropy and parse trees of sentences as a function of the sentence number. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 65–72, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.

P. Germain, A. Lacasse, F. Laviolette, and M. Marchand. Pac-bayesian learning of linear classifiers. In *ICML*, 2008.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1995.

I. Guyon, V. Vapnik, B. Boser, L. Bottou, and S. A. Solla. Structural risk minimization for character recognition. In J. E. Moody, S. J. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 471–479, San Mateo, CA, 1992. Morgan Kaufmann Publishers.

X. He, S. Yan, Y. Hu, and P. Niyogi. Face recognition using Laplacianfaces. *IEEE Trans. Pattern Anal. Mach. Intelli.*, 27(3):328–340, 2005.

D. Hsu, S. M. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *Twenty-Third Annual Conference on Neural Information Processing Systems*, CanadaA, 2009.

P. Indyk and R. Motawani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the* 30^{th} *Symposium on Theory of Computing*, pages 604–613, 1998.

T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 470–476, Cambridge, MA, 2000. MIT Press.

J. Janssen and N. Limnios. *Semi-Markov Models and Applications*. Kluwer Academic, 1999.

H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

M. I. Jordan. *An Introduction to Probabilistic Graphical Models*. MIT Press, 2008. To Appear.

Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *International Conference on Computer Vision*, volume 1, pages 166 – 173, October 2005.

L. Kontorovich. A universal kernel for learning regular languages. In *Machine Learning in Graphs*, 2007.

J. D. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *Proc. Intl. Conf. Machine Learning*, volume 18, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

J. Langford. Tutorial on practical prediction theory for classification. *JMLR*, 6: 273–306, 2005.

J. Langford, L. Li, and A. Strehl. Vowpal wabbit online learning project, 2007. `http://hunch.net/?p=309`.

J. Langford, M. Seeger, and N. Megiddo. An improved predictive accuracy bound for averaging classifiers. In *ICML*, 2001.

J. Langford and J. Shawe-Taylor. Pac-bayes and margin. In *Neural Information Processing Systems*. MIT Press, 2003.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, UK, 1996.

D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5: 361–397, 2004.

D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

P. Li, K. Church, and T. Hastie. Conditional random sampling: A sketch-based sampling technique for sparse data. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 873–880. MIT Press, Cambridge, MA, 2007.

Y. Liu. Fisher consistency of multicategory support vector machines. In *Proc. Intl. Conf. Machine Learning*, 2007.

D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

O. L. Mangasarian. Generalized support vector machines. Technical report, University of Wisconsin, Computer Sciences Department, Madison, 1998.

A. Martinez and R. Benavente. The ar face database. Technical Report 24, CVC Tech. Report, 1998.

D. McAllester. Generalization bounds and consistency for structured labeling. In *Predicting Structured Data*, Cambridge, Massachusetts, 2007. MIT Press.

D. A. McAllester. Some PAC Bayesian theorems. In *Proc. Annual Conf. Computational Learning Theory*, pages 230–234, Madison, Wisconsin, 1998. ACM Press.

D. A. McAllester. PAC-Bayesian model averaging. In *Proc. Annual Conf. Computational Learning Theory*, pages 164–170, Santa Cruz, USA, 1999.

D. A. McAllester. Pac-bayesian stochastic model selection. *ML*, 2001.

B. D. McKay. nauty user's guide. Technical report, Dept. Computer Science, Austral. Nat. Univ., 1984.

A. Nobel and A. Dembo. A note on uniform laws of averages for dependent processes. *Statistics and Probability Letters*, 17:169–172, 1993.

S. Nowozin, G. Bakir, and K. Tsuda. Discriminative subsequence mining for action classification. In *International Conference on Computer Vision*, 2007.

S. Park and H. Kautz. Hierarchical recognition of activities of daily living using multi-scale, multiperspective vision and rfid. the 4th. In *IET International Conference on Intelligent Environments*, page 2008, 2008.

J. Phillips, G. Humphreys, U. Noppeney, and C. Price. The neural substrates of action retrieval: An examination of semantic and visual routes to action. *Visual Cognition*, 9(4-5):662–685, 2002.

N. Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, Jan 2007.

G. Raetsch and S. Sonnenburg. Large scale hidden semi-markov SVMs. In *Advances in Neural Information Processing Systems 19*, 2006.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

X. Ren and J. Malik. Learning a classification model for segmentation. In *Proc. 9th Int'l. Conf. Computer Vision*, volume 1, pages 10–17, 2003.

R.Gross and J. Shi. The CMU motion of body (MoBo) database. Technical Report Tech. Report CMU-RI-TR-01-18, Robotics Institute, Carnegie Mellon University, 2001.

M. Rudelson and R. Veshynin. Geometric approach to error correcting codes and reconstruction of signals. *Int. Math. Res. Notices*, 64:4019–4041, 2005.

S. Sarawagi and W. Cohen. Semi-markov conditional random fields for information extraction. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, 2004.

R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

K. Schindler and L. van Gool. Action snippets: How many frames does human action recognition require? *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2008.

B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, Munich, 1997. Download: http://www.kernel-machines.org.

C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *Proc. Intl. Conf. Pattern Recognition*, pages 32–36, Washington, DC, USA, 2004. IEEE Computer Society.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, pages 213–220, Edmonton, Canada, 2003. Association for Computational Linguistics.

J. Shawe-Taylor, P. Bartlett, R. C. Williamson, and M. Anthony. A framework for structural risk minimization. In *Proc. Annual Conf. Computational Learning Theory*, pages 68–76, New York, 1996. Association for Computing Machinery.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.

Q. Shi, Y. Altun, A. Smola, and S. V. N. Vishwanathan. Semi-markov models for sequence segmentation. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 640–648, 2007.

Q. Shi, H. Li, and C. Shen. Rapid face recognition using hashing. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Francisco, USA, 2010a.

Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, A. Strehl, and S. V. N. Vishwanathan. Hash kernels. In M. Welling and D. van Dyk, editors, *Proc. Intl. Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, 2009a.

Q. Shi, J. Petterson, G. Dror, J. Langford, A. J. Smola, and S. Vishwanathan. Hash kernels for structured data. *Journal of Machine Learning Research - Special Topic on Large Scale Learning*, 10:2615–2637, Nov. 2009b.

Q. Shi, M. Reid, and T. Caetano. Hybrid model of conditional random field and support vector machine. In *Workshop at the 23rd Annual Conference on Neural Information Processing Systems*, Vancouver/Whistler, B.C., Canada, 2009c.

Q. Shi, M. Reid, and T. Caetano. Conditional random fields and support vector machines for structured prediction: A hybrid approach. In *Proc. Intl. Conf. Machine Learning*, Haifa, Israel, 2010b. submitted.

Q. Shi, L. Wang, L. Cheng, and A. J. Smola. Discriminative human action segmentation and recognition using semi-markov model. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Anchorage, AK, 2008.

Q. Shi, L. Wang, L. Cheng, and A. J. Smola. Discriminative human action segmentation and recognition using semi-markov model. *International Journal of Computer Vision*, 2009d. Accepted under revision.

J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *In ECCV*, pages 1–15, 2006.

J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, Oct. 2003.

A. Smola, S. Vishwanathan, and Q. Le. Bundle methods for machine learning. In *NIPS*, 2007.

Y. Song, F. Nie, C. Zhang, and S. Xiang. A unified framework for semi-supervised dimensionality reduction. *Pattern Recognition*, 41(9):2789–2799, 2008.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening lp relaxations for map using message passing. In *Proceedings of the 24th Annual Conference on Uncertainty in Artificial Intelligence (UAI-08)*, pages 503–510. MIT Press, 2008.

C. Sporleder and M. Lapata. Broad coverage paragraph segmentation across languages and domains. *ACM Trans. Speech Lang. Process.*, 3(2):1–35, 2006.

M. Stojnic. Block-length dependent thresholds in block-sparse compressed sensing. In *arxiv.org*, 2009.

B. Taskar. *Learning Structured Prediction Models: A Large Margin Approach.* PhD thesis, Stanford University, 2004.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32, Cambridge, MA, 2004. MIT Press.

C. Teo, Q. Le, A. Smola, and S. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *KDD*, 2007.

C. H. Teo and S. V. N. Vishwanathan. Fast and space efficient string kernels using suffix arrays. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 929–936, New York, NY, USA, 2006. ACM Press.

A. Tewari and P. Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8:1007–1025, 2007.

H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma. Statistical evaluation of the predictive toxicology challenge 2000-2001. *Bioinformatics*, 19 (10):1183–1193, July 2003.

J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Information Theory*, 53(12):4655–4666, 2007.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. Intl. Conf. Machine Learning*, New York, NY, USA, 2004. ACM Press.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6: 1453–1484, 2005.

K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18 (Suppl. 2):S268–S275, 2002.

M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 586–591, Hawaii, June 1991.

V. Vapnik. Structure of statistical learning theory. In A. Gammerman, editor, *Computational and Probabalistic Reasoning*, chapter 1. John Wiley and Sons, Chichester, 1996.

S. V. N. Vishwanathan, K. Borgwardt, and N. N. Schraudolph. Fast computation of graph kernels. In B. Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*, Cambridge MA, 2007a. MIT Press.

S. V. N. Vishwanathan, A. J. Smola, and R. Vidal. Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. *International Journal of Computer Vision*, 73(1):95–119, 2007b.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Department of Statistics, September 2003.

C. Watkins. Dynamic alignment kernels. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50, Cambridge, MA, 2000. MIT Press.

K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford, and A. Smola. Feature hashing for large scale multitask learning. In L. Bottou and M. Littman, editors, *International Conference on Machine Learning*, 2009.

K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10:207–244, 2009.

S. Wong, T. Kim, and R. Cipolla. Learning motion categories using both semantic and structural information. In *IEEE Conf. on CVPR*, pages 1–6, 2007.

J. Wright, A. Y. Yang, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intelli.*, 2008.

A. Y. Yang, J. Wright, Y. Ma, and S. S. Sastry. Feature selection in face recognition: A sparse representation perspective. *Tech. Report*, 2007.

C. Zhu. Ut once more: The sentence as the key functional unit of translation. *Meta*, 44:429–447, 1999.

J. Zhu and E. P. Xing. Maximum entropy discrimination markov networks. *JMLR*, 2009.