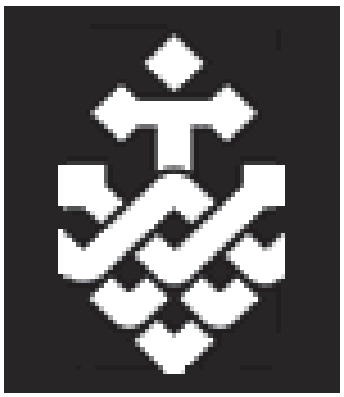
Semi-supervised and Unsupervised Extensions to Maximum-Margin Structured Prediction



Shaukat Abidi

Faculty of Engineering and Information Technology

University of Technology Sydney

A thesis submitted for the degree of

Doctor of Philosophy

2016

Certificate of Original Authorship

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Student's Name : Shaukat Abidi
Signature of Student :
Date : 18/July/2016

Abstract

Structured prediction is the backbone of various computer vision and machine learning applications. Inspired by the success of maximum-margin classifiers in the recent years; in this thesis, we will present novel semi-supervised and unsupervised extensions to structured prediction via maximum-margin classifiers.

For semi-supervised structured prediction, we have tackled the problem of recognizing actions from single images. Action recognition from a single image is an important task for applications such as image annotation, robotic navigation, video surveillance and several others. We propose approaching action recognition by first partitioning the entire image into "superpixels", and then using their latent classes as attributes of the action. The action class is predicted based on a graphical model composed of measurements from each superpixel and a fully-connected graph of superpixel classes. The model is learned using a latent structural SVM approach, and an efficient, greedy algorithm is proposed to provide inference over the graph. Differently from most existing methods, the proposed approach does not require annotation of the actor (usually provided as a bounding box).

For the unsupervised extension of structured prediction, we considered the case of labeling binary sequences. This case is important in a detection scenario, where one is interested in detecting an action or an event. In particular, we address the unsupervised SVM relaxation recently proposed in (Li et al. 2013) and extend it for structured prediction by merging it with structural SVM. The main contribution of the proposed extension (named Well-SSVM) is a re-organization of the feature map and loss function of structural SVM that permits finding the violating labelings required by the relaxation. Experiments on synthetic and real datasets in a

fully unsupervised setting reveal a competitive performance as opposed to other unsupervised algorithms such as k-means and latent structural SVM.

Finally, we approached the problem of unsupervised structured prediction by M^3 Networks. M^3 Networks are an alternative formulation of maximum-margin structured prediction that can satisfy the complete set of constraints for decomposable feature and loss functions; hence, the entire set of constraints is considered during the search for the optimal margin as opposed to Structural SVM. In the thesis, we present the interpretation of M^3 Networks in Well-SSVM, thus allowing us to use in a semi-supervised and unsupervised scenario.

Acknowledgements

I would like to take this opportunity to acknowledge enormous support from my supervisors, very useful suggestions from my group and a friendly environment of our lab especially during coffees, lunches and dinners.

First of all, I would like to thank my principal supervisor Professor Mary-Anne Williams who provided me the opportunity to come to Australia for PhD. Without her continuous support and supervision, I would have never gotten a chance to explore beautiful practical applications of robotics and computer vision.

I would like to convey big thanks to my co-supervisor Professor Massimo Piccardi, who nurtured my technical knowledge for doctoral degree. His stream of ideas kept me occupied due to which I was able to explore amazing research path. His continuous technical support especially regular meetings, sometimes at coffee shops, has played a central role in the formulation of my technical abilities.

I am thankful to all of my friends, colleagues and group members of Magic Lab and Surveillance Lab: Dr. Benjamin Johnston, Dr. Xun Wang, Dr. Saleha Raza, Dr. Rony Novianto, Wei Wang, Pramod Parajulli, Jonathan Vitale, Mahya Mirzae, Ali Raza, Sylvan Rudduck, Nima Ramezani, Robert Lange, Sari Awwad, Fairouz Hussein, and Ava Bargi. I am thankful to Professor Ivor Tsang, who provided valuable insights and critical reviews for my work. I am grateful to the visiting professors of Magic lab who shared their research experience with me: Professor Pavlos Peppas, Professor Peter Gärdenfors and Associate Professor Sajjad Haider. I would like to thanks Benjamin Johnston and Xun Wang again, with whom I developed demos for robots.

In the end, I would like to thanks my parents without whom I would never be a person I am at the moment and will be in the future.

Contents

List of Figures

List of Tables

Dedicated to my Parents

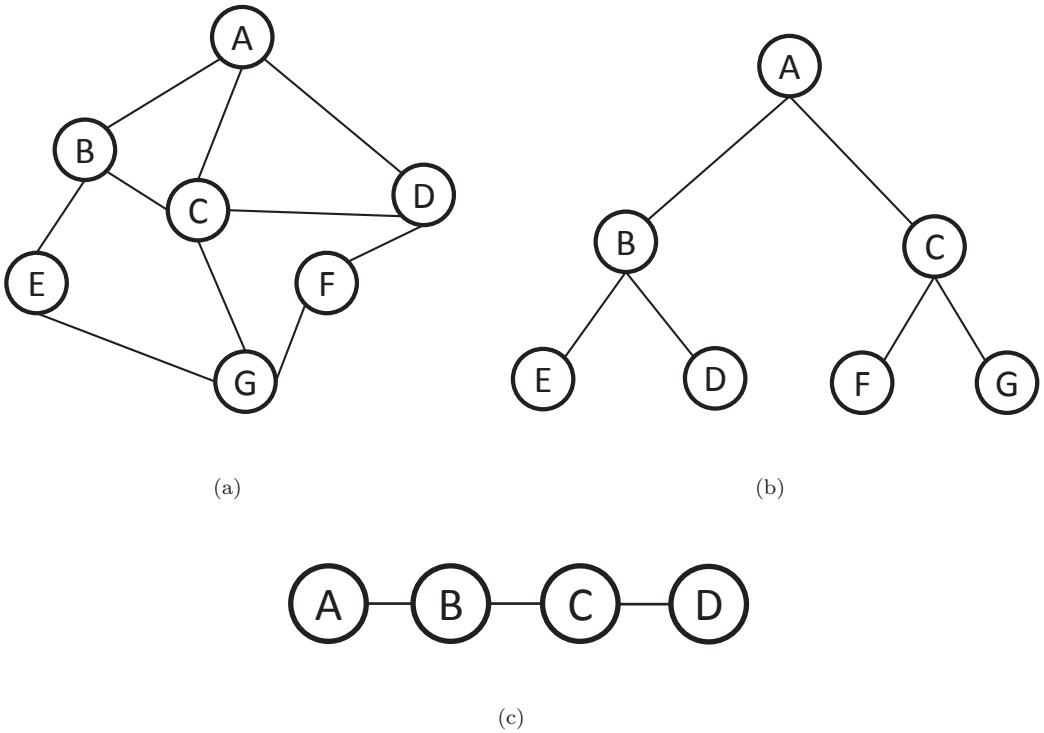
Chapter 1

Introduction

Structured prediction aims to predict complex objects such as graphs, sequences and trees. We are interested in the classification of such objects; elements of complex objects will receive a label from a discrete set Y as opposed to regression where Y is a set of real numbers.

The aforementioned complex objects have been studied thoroughly in the field of Mathematics and have found several applications in Computer Science. We will define graphs, trees and sequences in their simplest form: a graph is a collection of nodes/vertices connected together by lines called edges (Figure 1.1(a)); a tree is an object that originates from a root node and terminates at leaf nodes (Figure 1.1(b)); and a sequence is the simplest structured object in which each node is connected to only one node by an edge forming a chain structure, therefore often referred to as linear chain (please see Figure (1.1(c))). Throughout this thesis, we will focus on undirected complex objects i.e. edges connecting nodes that do not have any associated direction.

In the domain of computer vision and machine learning, several applications make use of such complex objects to label unknown entities. For instance, action recognition in videos can make use of a sequence structure to label each frame. Similarly, classification of a human body in images can make use of a tree structure where the node representing the head will serve as the *root* node and feet will represent *leaf* nodes.



(a) (b)

(c)

FIGURE 1.1: Examples of complex objects: a) graph with 7 nodes; b) tree with 7 nodes where A is the root node and {E,D,F,G} are leaf nodes; c) a sequence with 4 nodes.

1.1 Structured Prediction

Structured prediction deals with the classification of structured objects such as graphs and trees. In this type of prediction, the class label assigned to a node depends on the class labels assigned to the other nodes. The edges encode the structure of dependencies. Consider a sequence as shown in Figure (1.1(c)). Rather than classifying all nodes separately, structured prediction aims to assign label to each node by considering its immediate neighbours.

One of the main challenges faced by structured prediction is how to handle its humongous output space. In most cases, its output space is too large to enumerate. Let's assume that the nodes (A,B,C and D) of the sequence shown in Figure (1.1(c)) can take binary values. With this assumption, 16 distinct labelings are possible; the number of labelings grows exponentially in the size of the sequence. It is evident

that number of labelings would explode if the size of sequence increases. The situation will become worse in the case of graphs and trees with large number of nodes and larger sets of labels. It is not uncommon to face such situation in practical applications. Despite such a challenge, structured prediction has found its use across several disciplines and has reported state-of-the-art performance.

Structured prediction has numerous applications in the fields of computer vision, natural language processing and bioinformatics to name some. In computer vision, recognizing human actions from a single image can be formulated as the task of classifying image portions and the human action jointly. Intuitively, we can model dependencies among actions and objects in the image. For instance, in the classification of "eating" action, the presence of plates and cups in the image will aid the process of action classification. Similarly, while classifying stream of characters forming a word, let's say "CLAPPING", we can take advantage of dependencies among characters using structured prediction. It simply means we will have some means to say that the chances of having character "L" after "C" are much higher than characters such as "S" or "V". Therefore, modeling structural dependencies among data is of vital importance across several disciplines.

To perform structured prediction, we need to "train" a classifier from the available training set. The problem of training "structural classifiers" is usually approached in two ways: (i) Generative and (ii) Discriminative training. In generative training of classifiers, we aim to model joint probability $P(x, y)$ where variable x is the measurement and y is the output variable. On the other hand, discriminative learning of classifiers can be viewed as methods that model conditional probability $P(y|x)$. In this thesis, we have trained structured predictors (or structural classifiers) by utilizing the notion of maximum margin, which falls inside the branch of discriminative training.

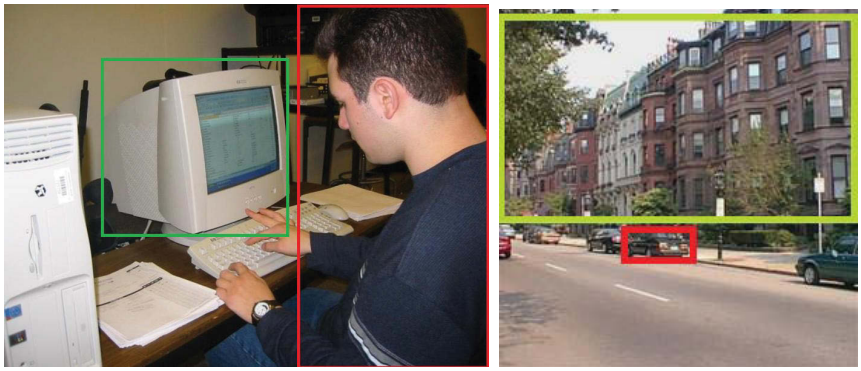
1.2 Semi-supervised Structured Prediction

Semi-supervised algorithms perform learning over a training set that consist of labeled and unlabeled samples. In the case of structured prediction, labelings for each

training instance form a graph or tree whose nodes are partially unobserved during training.

Several applications in the field of computer vision are addressed via semi-supervised learning. Consider the case of action and scene classification as depicted in Figure 1.2(a) and Figure 1.2(b), respectively. Both of these tasks can be solved via structured prediction. In Figure 1.2(a), the goal is to classify the action performed in the image. This can be achieved by jointly labeling the objects inside the image. For instance, the green and red rectangles highlight the presence of a monitor screen and a human. During semi-supervised learning of actions, the action class would be provided while the location and class of objects (for instance monitor and human) would be latent or unobserved. Our learning algorithm will assign values to the unobserved labels by minimizing some notion of error. During prediction, our classifier will predict the action along with the detected objects. We refer to this case as semi-supervised action recognition. We can apply the same learning paradigm for the task of scene classification as well. In Figure 1.2(b), the red and green rectangles highlight the presence of a car and a building in the image. Our learning algorithm can treat the locations and classes of these objects as latent variables while the scene class ("street") would be observed. During prediction, scene recognition will be aided by the detection of objects inside the image. We refer to this scenario as semi-supervised scene classification.

Success in semi-supervised learning can bring huge benefits for labeling enormous amount of data. Since the data size is growing at unprecedented rate, data annotation has become an expensive operation. A huge effort is required to annotate millions of images (for action/scene recognition) and video frames (action recognition from videos). While it is possible to assign a single label to images or videos, it is infeasible to annotate all objects constituting them. In such scenario, semi-supervised learning frames the task of learning classifiers by assigning labels to the unlabeled objects by itself. In this way, one can achieve the goal of annotating objects and at the same time, missing labels are handled efficiently by the algorithm.



(a) (b)

FIGURE 1.2: a) Action class: using a computer (Yao et al. (2011a)) b) Scene class: street (Xiao et al. (2010)).

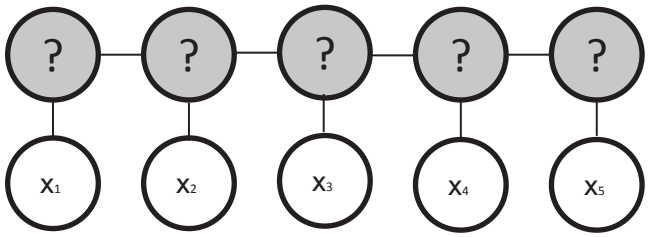


FIGURE 1.3: An example of unsupervised training for sequence prediction with 5 output (shaded) nodes.

1.3 Unsupervised Structured Prediction

Unsupervised algorithms perform the task of learning a classifier over an unlabeled training set. Unsupervised learning is also commonly referred to as "clustering". In the case of structured prediction, the labelings for each training instance form a graph or a tree whose output nodes are unobserved during training.

In Figure 1.3, a synthetic case for sequential labeling with 5 output nodes is depicted as an example of unsupervised learning. During training, only x is observed (x is the input vector or observation) and the task is to find its corresponding labeling. This approach is useful in several applications, for instance gesture segmentation. Given a video of an actor performing "K" gestures (frames are represented by x_i), we can perform clustering over video frames that will assign a label (out of K labels) to each

frame. K-means is an immediate option to achieve such clustering; yet, it will ignore completely the dependencies among output nodes (shaded nodes in Figure 1.3). We will discuss a novel method of performing unsupervised structured prediction (or clustering) for the above mentioned case in Chapter 4.

1.4 Contributions

In this thesis, we have proposed the following contributions to semi-supervised and un-supervised structured prediction:

- Proposed efficient greedy algorithm for the task of predicting human actions from still images (Chapter 3).
- A novel formulation of unsupervised structured prediction based on a minimax relaxation (Chapter 4).
- Proposed alternative solution for the task of unsupervised structured prediction via M^3 Networks (Chapter 5).

1.5 Thesis Organization

The thesis is organized into six chapters that are summarized as follows:

Chapter 2. This chapter presents the background knowledge for the proposed contributions. It discusses the learning of maximum-margin classifiers for "independent and identically distributed (i.i.d)" and sequential datasets. How to move from a simple i.i.d training to the structured prediction is the focus of entire discussion. Finally, a brief introduction of Multiple Kernel Learning (MKL) is provided to lay the foundation of Chapter 4.

Chapter 3. This chapter sets out to offer solution for the task of recognizing human actions in a semi-supervised manner. Particularly, we have proposed a graphical

model that can be used to classify human actions from still images. Later, we will discover that training such a classifier is an expensive process. To mollify this issue, we have proposed efficient greedy inference algorithms that make the learning task tractable.

Chapter 4. In this chapter, we focus on how to perform unsupervised structured prediction via maximum-margin training. We have extended a recently proposed convex relaxation for labeling binary datasets to the case of structural datasets. To this aim, we have proposed a feature map design that is the main element of our new methodology.

Chapter 5. In Chapter 4, the quadratic program (QP) for Well-SSVM is solved using cutting-plane approach. In this chapter, we have proposed an alternative way for solving the same QP with full set of constraints. The proposed formulation is the direct application of M^3 Networks.

Chapter 6. We have concluded our discussion with a summary of the main findings and suggestions for possible future extensions.

Chapter 2

Literature Review

In this chapter, we will review the formulation of support vector machines (SVMs) for *independent and identically distributed* (I.I.D) data (such as binary/multiclass SVM), an overview of Multiple Kernel Learning (MKL) – an essential approach to tackle the case of multiple ground-truth labelings in Chapter 4 and Chapter 5, followed by Structural and Latent Structural SVM for structured data. Although, there is an entire field of graphical models that handles the probabilistics learning of structured dataset (Koller and Friedman (2009)), we have focused on the discriminative way of learning classfiers i.e. maximum-margin approach (Structural and Latent Structural SVM). The literature review will be concluded by the application of structured prediction in the field of computer vision for recognizing human actions from still images.

2.1 Maximum Margin Classifiers

In the following sequel, we aim to learn a linear discriminant function using the maximum margin framework. A function $h(x)$ is called a linear discriminant function if:

$$h(x) = w^\top x + b = \begin{cases} h(x) > 0 & \text{when } x \in A \\ h(x) < 0 & \text{when } x \in B \end{cases}$$

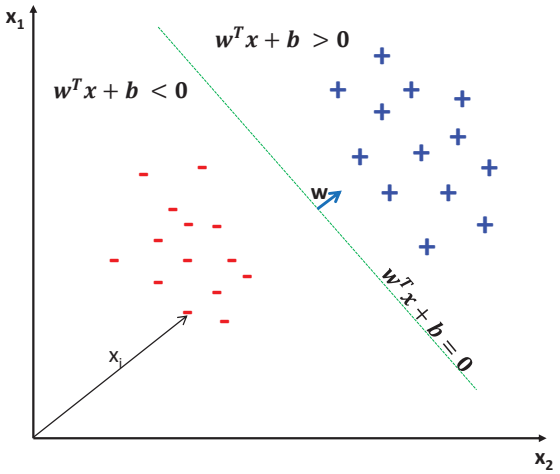


FIGURE 2.1: An example of linear discriminant function separating distinct regions of points in 2D-space

Imagine we wish to separate points as shown in Figure 2.1. Here, $h(x) = 0$ corresponds to the line (hyperplane in high dimensions) that cleanly separates the red and blue points. All blue points will fall inside region "A" where $h(x) > 0$ while the red ones, in region "B" where $h(x) < 0$. "w" is the vector normal to line $h(x) = 0$ and "b" is its offset from the origin.

The parameters, w and b, of this function will be learnt from a training set by utilizing the notion of maximizing the margin between the closest points of opposing class. After parameter learning, any data point x can be classified using $h(x)$.

In this chapter, we will introduce the notion of maximum-margin learning in a supervised and semi-supervised setting. The discussion will be concluded with detailed explanation of different formulations of SVMs that will lay the foundation of our proposed techniques for performing: (i) static action recognition (Chapter 3) and (ii) unsupervised structured prediction (Chapter 4).

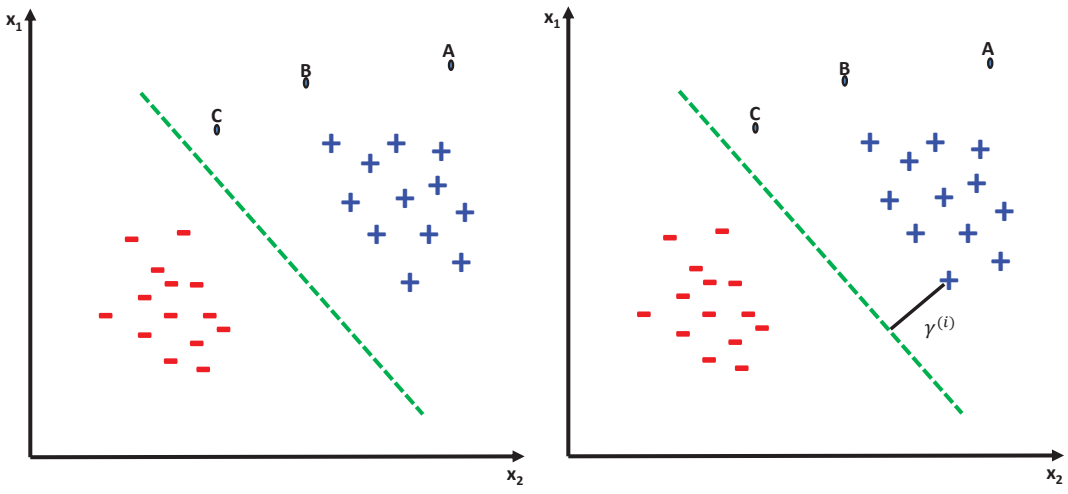


FIGURE 2.2: a) Functional margin of points A, B and C; b) Geometric margin define as a distance between point A and B.

2.2 Support Vector Machines (Binary Case)

Binary SVM learns a classifier that separates training data into two classes by utilizing the notion of maximum margin. We will elucidate the notion of "margin", that will make the task of formulating hard and soft margin SVM easier.

In the following section, X is a collection of N points with D-dimensions i.e. $X = \{x_1, x_2, \ldots, x_N\}$ and $x_i \in R^D$. Each point has an associated label y_i, therefore the corresponding label set is denoted by $Y = \{y_1, y_2, \ldots, y_N\}$ where each label y_i picks a value from the set $\{+1, -1\}$. The overall set for training sbinary SVM is denoted by $S = \{(x_i, y_i)\}_{i=1,\ldots,N}$. We would like to mention that the following discussion on binary SVMs is inspired by Andrew Ng's lecture notes on SVMs (Ng (2003)).

2.2.1 Intuitions for Margin

Consider Figure 2.2 depicting a line that separates the data points of the positive and negative classes. We will define this line by $h_{w,b}(x) = w^\top x + b = 0$. "$h_{w,b}(x)$" can change its position and orientation by varying its parameters (w, b). As we can

see, all points lying on the upper side of $h_{w,b}$ belong to the positive class and the rest are members of the negative class. From now on, we will call this separating line as "decision boundary" or "separating hyperplane" since it acts as a plane that separates different classes. Now, we are in a position to define "functional" and "geometric" margin from the decision boundary.

During classification, our goal it to classify points with high confidence. Let us measure this confidence by means of functional margin, $\tilde{\zeta}$. For instance, in Figure 2.2, $h_{w,b}(x = A) > h_{w,b}(x = B) > h_{w,b}(x = C)$. The reason for such inequality is that the distance of "A" from $h_{w,b}(x) = 0$ is larger than "B" and "C". Therefore, we can say that the functional margin of "A" is larger than the functional margin of "B" and "C". As a consequence, we are more confident in the prediction of "A" rather than "B" or "C". We can scale the functional margin as much as we like by scaling "w" and "b". To introduce the distance of a point from a line in Euclidean space, we will now introduce the notion of geometric margin.

The geometric margin for x_i is defined as the distance of x_i from decision boundary $h_{w,b}$. In Figure 2.2, the geometric margin of arbitrary point x_i is denoted by ζ_i. Such geometric margin is the perpendicular distance of x_i from the decision boundary. Using simple formula for the perpendicular distance between any point and line in Euclidean space, we can easily impute the geometric margin as follows:

$$\zeta_i = \frac{w^\top x_i + b}{||w||}$$

The numerator of ζ_i is our previously defined functional margin. Therefore, the above equation establishes the relation between functional and geometric margins of binary support vector machines.

During the training phase, the geometric margin of $h_{w,b}$ w.r.t given training set S of N examples is set to the smallest geometric margin i.e.

$$\zeta = \min_{i=1,\ldots,N} \zeta_i$$

In simpler words, given a Euclidean space and separating hyperplane, the distance between any point x and the hyperplane is uniquely defined. Instead, the functional margin (also known as "score") is defined up to an arbitrary multiplicative constant, $||w||$.

The goal of SVM learning is to find the parameters of separating hyperplane (w, b) that maximize the geometric margin. Alternatively, we have to find the best hyperplane from a family of linear functions (defined by $h_{w,b}$) that maximizes the geometric margin on both sides of the hyperplane.

2.2.2 Hard-Margin SVM

Consider the hard-margin case as depicted in Figure 2.3. The name "hard-margin" points to the fact that the data points are completely separable by a linear decision boundary. The two dotted lines are at a functional margin of 1 from the decision boundary. The geometric margin can easily be calculated as shown in figure, $\frac{2}{||w||}$. In order to find a hyperplane that maximizes the distance between the closest points of opposing class, it is sufficient to minimize the norm of w i.e. $||w||$. Therefore, we can pose our primal problem for *hard-margin* SVM as follows:

$$\begin{aligned} (w^*, b^*) &= \underset{w,b}{\operatorname{argmin}} \ \frac{1}{2} \|w\|^2 \\ s.t. \ & y_i \left(w^\top x_i + b \right) \geq 1, \forall i = 1, 2, \ldots, N \end{aligned} \tag{2.1}$$

By treating (2.1) as a Lagrangian equation (see Appendix. A), the dual formulation of hard-margin SVM turns out to be:

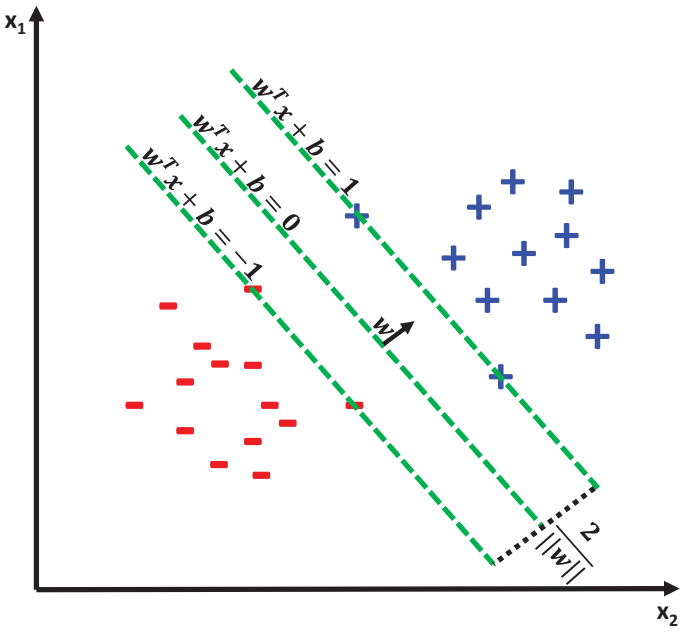


FIGURE 2.3: Hard margin SVM: When data is separable by linear decision boundary.

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^\top x_j \\ &s.t.\ \alpha_i \geq 0, \forall i = 1, \ldots, N \\ &\sum_{i=1}^{N} \alpha_i y_i = 0, \end{aligned} \tag{2.2}$$

Points (or examples) that lie on the dashed lines in Figure 2.3 are known as "support vectors" and their corresponding $\alpha_i > 0$. All other points that do not lie on the dashed line will have their corresponding $\alpha_i = 0$. Therefore, at optimality only few points (or support vectors) will have non-zero alphas. One of the great advantages of this dual formulation is its reliance on a kernel function. If we observe Equation (2.2), we can see that it uses a dot product between pairs of input vectors denoted by x_i and x_j. Therefore, we can replace $x_i^\top x_j$ by our desired kernel function (for instance, a Gaussian kernel). As a consequence, the dual formulation of SVM with

kernels other than the dot product ("linear kernel") leads to non-linear decision boundaries in data space R^D. This is called the kernel trick, using which we can work effectively in very high-dimensional spaces, even infinite dimensions. The notion of using kernels is more general than SVM, therefore it can be applied to any algorithm that uses the dot product between pairs of inputs x_i and x_j.

Once we get α_i's from Objective (2.2), we can classify test example x_t as follows:

$$y_t = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i x_i^\top x_t\right) \tag{2.3}$$

2.2.3 Soft-Margin SVM

In practical scenarios, it is very rare that the given dataset is neatly separable in data space (Figure 2.3). Generally, it contains data points that are not separable by linear decision boundaries. Moreover, the presence of outliers may bias the position of the decision boundary immensely. This case is depicted in Figure 2.4, where the task is to find a seperating hyperplane when some data points overlap in opposing classes. To solve the case of non-separable data, a new set of variables (slack variables), ξ_i, are introduced in the optimization objective. This provides a way to select a trade-off between training error and margin size. With these modifications, our new primal problem for this so called *soft-margin* SVM is posed as follows:

$$\begin{aligned} (w^*, b^*) &= \underset{w,b}{\text{argmin}}\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ s.t.\ & y_i\left(w^\top x_i + b\right) \geq 1 - \xi_i, \forall i = 1, \ldots, N \\ & \xi_i \geq 0, \forall i = 1, \ldots, N \end{aligned} \tag{2.4}$$

The new constraint allows us to have data points with functional margin less than 1 by incorporating the slack variables. These slack variables are quantified as penalties for misclassification. For each misclassification (a case when functional margin is less than 1), our objective function receives penalty of $C\xi_i$. The parameter C determines

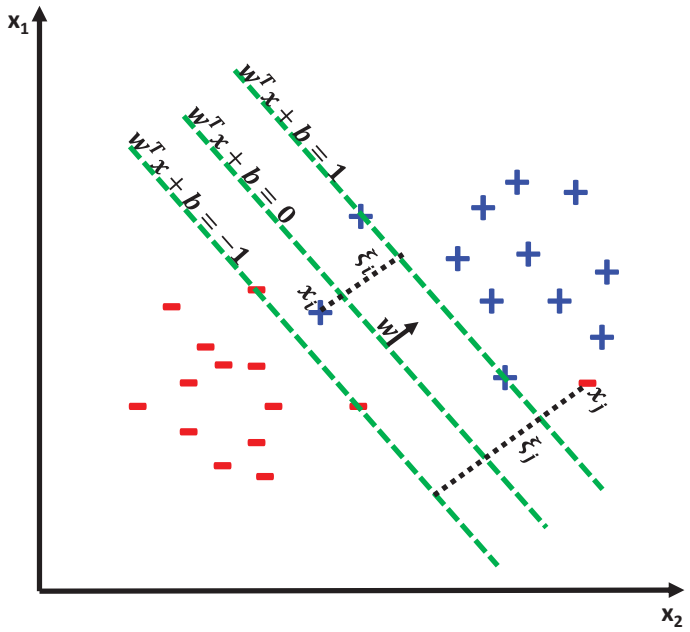


FIGURE 2.4: Soft margin SVM: When data is non-separable by linear decision boundary, the case of overlapping classes.

the trade-off between margin and training error i.e. it is a weighting factor that controls the margin (if $||w||^2$ is small, the margin will increase and vice versa) and number of examples that have margin equal to $1 - \xi_i$. "C" is defined by the user and is normally chosen via cross-validation.

Actually, what we are interested in is to minimize the number of misclassified examples. Please note that if we were to count the number of misclassified examples and use it as the objective, then Objective (2.4) will become non-differentiable and non-convex as ξ would be an L_0 norm. Therefore, instead of this loss, we use a piecewise linear function known as the "hinge loss" (Figure 2.5). It is convex and serves as an upper bound to the ordinary 0-1 loss. For binary classification problem where y_i can take values from the set $\{+1, -1\}$, hinge loss can be written as: $\max(0, 1 - y_i w^\top x_i)$. In Figure 2.5, respective 0-1 and hinge loss are plotted for $y_i = 1$.

The dual of Equation (2.4) can be obtained in the same fashion of hard-margin SVM. After forming its Lagrangian, differentiating in w and b and plugging the results back into the Lagrangian, we can pose the dual problem as follows:

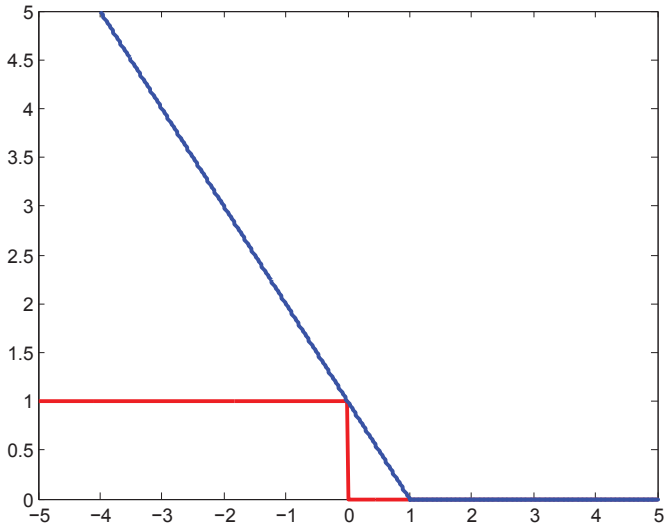


FIGURE 2.5: Hinge loss (blue) as a convex surrogate of zero-one loss (red).

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^\top x_j \\ &s.t.\ 0 \leq \alpha_i \leq C, \forall i = 1, \ldots, N \\ &\sum_{i=1}^{N} \alpha_i y_i = 0, \end{aligned} \tag{2.5}$$

The only difference between the Objective (2.2) and Objective (2.5) is the addition of a "box constraint", $0 \leq \alpha \leq C$. Similar to Objective (2.2), we can apply the kernel trick to learn non-linear decision boundaries (in high, even infinite, dimensional spaces). Objective (2.2) and Objective (2.5) are quadratic programs with linear constraints. They are efficiently solved via off-the-shelf solvers, among which SMO (Platt et al. (1999)) is widely used.

2.3 Multiclass Support Vector Machines

We have seen how a binary SVM learns the decision boundary for a dataset having two classes. What should be done if we have more than two classes? This case is depicted in Figure 2.6 where the task is to separate three classes by linear hyperplanes.

Multiclass classification can be approached in the following two intuitive ways: (i) design a one-vs-all classifier for each class (Rifkin and Klautau (2004)) or (ii) design a one-vs-one classifier for each pair of classes (Hastie et al. (1998)). If the number of classes is denoted by K, then the former case requires the training of K binary classifiers and the latter requires training $\frac{K(K-1)}{2}$ binary classifiers. We will discuss the training of the first case since it leads to the multiclass training of SVM when classes are considered altogether.

As mentioned earlier, let us discuss the case of training a one-vs-all classifier (Rifkin and Klautau (2004)). For the sake of simplicity, let us assume we want to find an unbiased decision boundary i.e. $b = 0$. Thus assigning a label to point x requires the training of K binary classifiers —one for each class. For any class k, we will have its respective weight vector w_k. To achieve high classification accuracy, a careful (manual) tuning of each classifier is needed. For prediction, the class with the highest score ($w_k^\top x$) —among K classes —will be selected as the target class. The most probable label y to x is assigned by evaluating the score of each class as follows:

$$y = \underset{k \in K}{\operatorname{argmax}}\, w_k^\top x \ , \forall k \in K \tag{2.6}$$

Crammer and Singer (2002) cast the objective of multiclass SVM as a single optimization problem for K classes. To obtain K models from a single training, the following (primal) objective was proposed:

$$\begin{aligned} &\min_{w_k \in R^D, \xi \in R^N} \sum_{k=1}^{K} \frac{1}{2} ||w_k||^2 + C \sum_{i=1}^{N} \xi_i \\ &s.t. \quad w_{y_i}^\top x_i - w_k^\top x_i \geq 1 - \xi_i, \ \ \forall i, \forall k \in K \end{aligned} \tag{2.7}$$

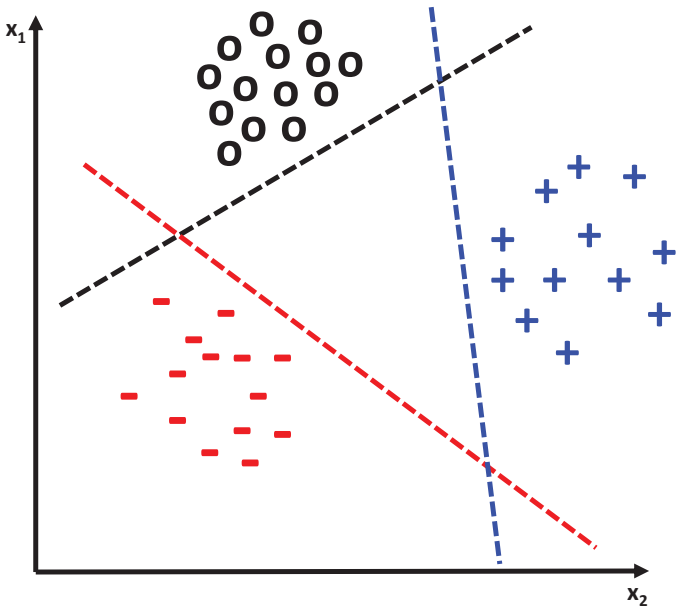


FIGURE 2.6: Multiclass SVM: (+, - and o) represent separate class and the task is to find decision boundaries that classify test point into 3-classes.

The objective posed in 2.7 is the soft-margin formulation of multiclass SVM. For a training datum x_i, y_i is its ground-truth label and ξ_i is its associated slack variable introduced in the objective to permit constraint violation. Each example has a set of $K-1$ constraints and the posed objective tries to maintain the difference of at least $1-\xi_i$ between the scores assigned to x_i by its true model (w_{y_i}) and all other models ($w_{k \in K \setminus y_i}$). In multiclass SVM, $w_k^\top x$ changes its meaning of boundary (as we have seen in binary SVM) and becomes a score relative to class k. The decision boundary among class k and l is given by $w_k^\top x - w_l^\top x = 0$. If $k = 2$ (binary SVM), then it can be proven that $w_1 = -w_2$.

2.4 Multiple Kernel Learning

The dual formulation of SVM (2.5) allows implementing kernels for learning non-linear decision boundaries. If x_i represents the input data point with label y_i, it is known that SVM aims to learn a linear discriminant function $f(x)$ of the following form:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i \left(x_i^\top x\right) + b \tag{2.8}$$

To learn a non-linear decision function, the input features x_i can be transformed into a high-dimensional space by a mapping function, $\phi(x_i)$. Fortunately, thanks to the dual formulation of SVM (2.5), all x_i's appear in a product with another sample. As a consequence, instead of applying an explicit mapping function to each input, kernels are used to implicitly represent such products in some high-dimension space. After introducing the kernel, the linear discriminant function shown in Equation (2.8) will take the following form:

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b \tag{2.9}$$

where $K(x_i, x) = \phi(x_i)^\top \phi(x)$ is the kernel function over x_i and x. Now, Objective (2.5) can be re-written as follows:

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\ &s.t.\ 0 \leq \alpha_i \leq C, \forall i = 1, \ldots, N \\ &\quad \sum_{i=1}^{N} \alpha_i y_i = 0, \end{aligned} \tag{2.10}$$

Up till here, we have utilized a single kernel K. Utilizing multiple kernels brings several advantages for various applications and has been studied extensively by (Bach et al. (2004); Rakotomamonjy et al. (2008); Gönen and Alpaydin (2008); Xu et al. (2013)), to name some. Therefore, instead of using just a single kernel, one can combine different kernels as follows:

$$K(x, x') = \sum_{l=1}^{L} \mu_l K_l(x, x') \quad \text{such that } \mu_l \geq 0, \ \sum_{l=1}^{L} \mu_l = 1 \tag{2.11}$$

In Equation (2.11), kernel K is the convex combination of L basis kernels weighted by their corresponding μ's. Other combinations are also possible such as conic combination. Such basis kernels can be totally different kernels such as RBF and Gaussian kernel, same kernel with different parameters such as Gaussian kernel with different sigmas or they may take the full set or subset of input variables thus allowing input from different sources.

Replacing the value of $K(x, x')$ from Equation (2.11) into Objective (2.10) will give rise to two unknowns, α_i and μ_l. Learning such unknowns (α and μ) in a single optimization problem is known as the *multiple kernel learning (MKL) problem* (Rakotomamonjy et al. (2008)).

Rakotomamonjy et al. (2008) proposed the following MKL primal problem whose simplest solution is obtained via alternate optimization algorithm:

$$\begin{aligned} \min_{\{w_l\}, b, \xi, \mu} \ & \frac{1}{2} \sum_{l=1}^{L} \frac{1}{\mu_l} ||w_l||^2 + C \sum_{i=1}^{N} \xi_i \\ \text{s.t} \quad & y_i \left(\sum_{l=1}^{L} w_l^\top x_i + b \right) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \qquad \forall i \\ & \sum_{l=1}^{L} \mu_l = 1, \ \mu_l \geq 0 \ \forall l \end{aligned} \tag{2.12}$$

with

$$w_l = \sum_{i=1}^{N} \mu_l \alpha_i y_i x_i$$

Objective (2.12) is the MKL SVM problem, which is referred to as the primal MKL problem by Rakotomamonjy et al. (2008). Please note the L1 norm constraint on weighting coefficient μ due to which the final kernel matrix is the convex combination of basis kernels (can be seen by deriving its dual). As mentioned before, the solution of Objective (2.12) can be obtained into two steps:

- Optimize Objective (2.12) by keeping μ fixed. It will be an instance of SVM, which can be solved by off-the-shelf solvers.
- Update the weight factor of basis kernel μ_l in a closed form to decrease the objective value of (2.12) while keeping b,ξ and w fixed.

The above method to solve MKL primal problem might have some convergence issues especially when few components of μ approach zero (Rakotomamonjy et al. (2008)). However, two great advantages offered by the aforementioned technique are its straightforward implementation and the cheap computation to estimate the weighting parameters.

2.5 Structured Prediction

Structured prediction is the task of predicting structured objects like graphs, sequences or trees. Each structured object consist of T nodes with E edges. Figure 2.7 is an illustration of a simple structured object, a sequence of T nodes having E (=T-1) edges. In our discussion of structured prediction, we will denote an input by $x = \{x^1, \ldots, x^T\}$ while $y = \{y^1, \ldots, y^T\}$ will be its corresponding labeling. Throughout our discussion on Structural SVM and Latent Structural SVM, elements of y will be discrete valued.

Structured prediction can be seen as a multi-class prediction problem with huge numbers of classes. The number of classes are exponential in the number of nodes

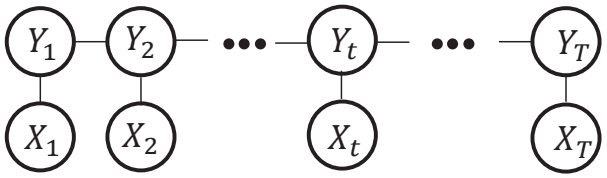


FIGURE 2.7: Structural SVM: An example of sequential labeling.

of the structured object. For instance, a linear sequence (see Figure 2.7) of length T where each output node have cardinality K, will have K^T unique sequences or classes. As an example, a sequence x of (T=)3 binary nodes (i.e. $|y| \in \{0, 1\}$) will have (2^T) 8 unique labelings (See Figure 2.8). We can refer to such 8 distinct labelings as separate classes (as mentioned in the start of this paragraph). For the rest of our discussion, we will refer to such number of classes as "distinct labelings".

The problem of structured prediction is solved under the framework of probabilistic graphical models. These graphical models are categorized into two types: i) Directed Graphs (Bayesian Network) ii) Undirected Graphs (Markov Random Fields). Once we have chosen the type and form of the graphical model, we need to learn its parameters. These parameters can be learnt using generative or discriminative approaches. In this thesis, we have taken into account the discriminative, maximum-margin way of learning parameters. Therefore, we will review Structural SVM (Tsochantaridis et al. (2005a)) for the task of learning such parameters. These learnt parameters will help us achieve the aim of structured prediction.

Structural SVM

Structural SVM (SSVM) is the discriminative method of training maximum-margin classifiers that predict structured objects like graphs, sequences and trees. Structured prediction is gaining popularity thanks to its promising performance across several fiels. Action recognition in videos and images, sequence alignment in bioinformatics, tracking multiple targets in videos are some of its popular applications, to name a few.

Let $x \in X$ denote an input vector with associated output variable $y \in Y$. The training set would then be represented as input-output pairs:

$\{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_N, y_N)\} \in X \times Y$. Let T_i denote the length of x_i, hence x_i^t points to the t^{th} element of i^{th} example. Each x_i will have its corresponding labeling y_i, whereas each element of y_i can have a discrete value from the set of K labels i.e. $|Y| = \{1, \ldots, K\}$. In the following, reference to an arbitrary example is made by x_i whereas reference to its arbitrary element t is made by x^t.

The overall goal of SSVM is to learn a discriminant function $f : X \times Y \longrightarrow R$ linear in the parameter set, w, having the following form:

$$\begin{aligned} f(x; w) &= \underset{y \in Y}{\operatorname{argmax}} F(x, y; w) \\ &= w^\top \Phi(x, y) \end{aligned} \tag{2.13}$$

$\Phi(x, y)$ is called joint feature map. Intuitively, it encodes structural information of input-output pair (x_i, y_i). Its design is application-dependent and varies with the nature of problem to be solved. The important thing to note is the linearity of $f(x; w)$ in weight vector w.

As an example, consider a case of tagging a sequence with parts-of-speech. For the sake of simplicity, let's restrict the length of sequence (T) to 3 words (i.e. T= 3) while each word can only be tagged as either "Noun (N)" or "Verb (V)" (i.e. cardinality of output node y_i is set to 2). As shown in Figure 2.8, the task is to tag input $x = \{I, like, fruits\}$ with the most likely sequence of tags. In Figure 2.8, the correct assignment of tags is shown on top of output nodes y_i. With the simplifications we made on output, we can see that there are only 8 possible sequences of tags. They are shown in Figure 2.8 on an axis labeled by Equation 2.13. The goal of SSVM learning is to find a weight vector w that assigns the highest score to the correct sequence. In SSVM learning, ideally all "wrong assignments" shown in the figure would serve as constraints. The "correct assignment", shown in figure, is the ground truth labeling for our simple example. The joint feature map, $\Phi(x, y)$, for the case

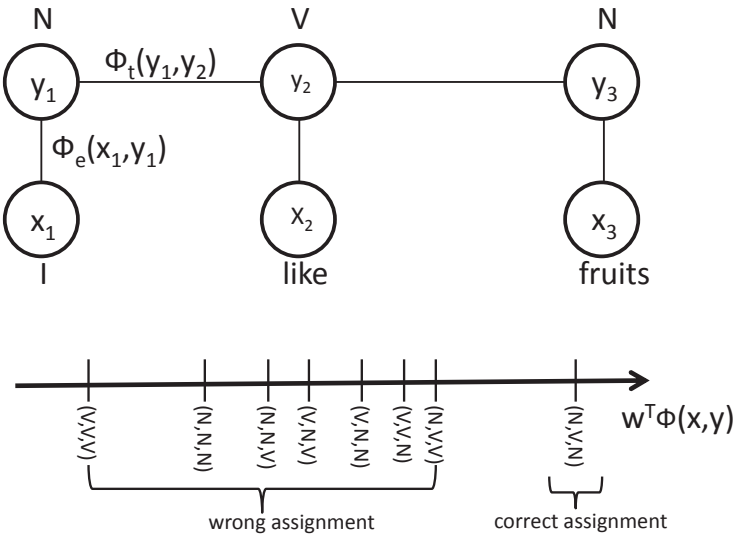


FIGURE 2.8: Scores for all possible combinations of a given sequence. There are few assumptions made for the sake of simplicity.

depicted in Figure 2.8 can be decomposed into "emission" and "transition" factors (Φ_e and Φ_t) as follows (Yu (2011)):

$$\Phi(x,y) = \sum_{t=1}^{T} \Phi_e(x_t, y_t) + \sum_{t=2}^{T} \Phi_t(y_t, y_{t-1}) \tag{2.14}$$

For linearly separable datasets, we can pose a convex quadratic program for Structural SVM (Tsochantaridis et al. (2005a)) with linear constraints as follows:

$$\begin{aligned} &\min_{w \in R^M} \ \frac{1}{2} \|w\|^2 \\ &s.t. \ \forall y \in Y \backslash y_i : w^\top \Phi(x_i, y_i) - w^\top \Phi(x_i, y) \geq 1, \forall i = 1, \ldots, N \end{aligned} \tag{2.15}$$

In structured prediction it is very rare to have linearly separable datasets, therefore we should incorporate slack variables in the objective of structural SVM. These

slack variables will permit margin violation while the objective function will receive a penalty quantified by ξ_i:

$$\begin{aligned} &\min_{w \in R^M, \xi \in R^N} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ &s.t.\ \forall y \in Y \backslash y_i : w^\top \Phi(x_i, y_i) - w^\top \Phi(x_i, y) \geq 1 - \xi_i, \forall i = 1, \ldots, N \\ &\qquad \xi_i \geq 0, \qquad\qquad \forall i = 1, \ldots, N \end{aligned} \tag{2.16}$$

Taskar et al. (2004) and Tsochantaridis et al. (2005a) introduced another way to incorporate loss by rescaling the margin. With these changes, the constraints in (2.16) take the following form:

$$\begin{aligned} &\min_{w \in R^M, \xi \in R^N} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ &s.t.\ \forall y \in Y \backslash y_i : w^\top \Phi(x_i, y_i) - w^\top \Phi(x_i, y) \geq \triangle(y_i, y) - \xi_i, \forall i = 1, \ldots, N \\ &\qquad \xi_i \geq 0, \qquad\qquad \forall i = 1, \ldots, N \end{aligned} \tag{2.17}$$

The above objective leads to the weight vector w that minimizes the sum of hinge losses $\sum_{i=1}^{N}\xi_i$. The user defined parameter "C" will choose an appropriate balance between the training error (ξ) and the regularizer($||w||^2$). The constraints in (2.17) say that the difference between the scores of groundtruth (y_i) and all other assignment ($y \neq y_i$) must be at least $\triangle(y_i, y) - \xi_i$. The number of constraints is exponential in the length of sequence x (shown in Figure 2.8 for the simplest case of sequential tagging). Therefore, it is not feasible to solve (2.17) with the full set of constraints since it makes learning intractable. To mollify the issue of intractable learning, Joachims et al. (2009a) proposed a cutting plane algorithm for solving (2.17) and its variants. The idea is to find the most violated constraint for each example; these constraints are added to a working set of constraints and the resulting quadratic program (QP) is solved. This process is repeated until convergence. The detailed algorithm is described in Joachims et al. (2009a).

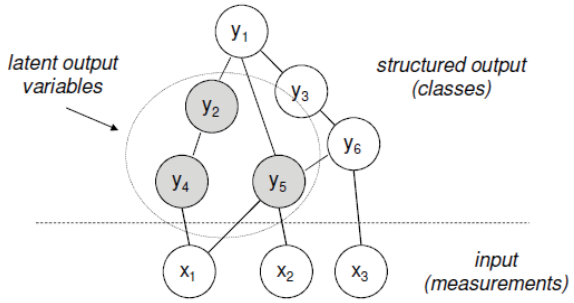


FIGURE 2.9: Graphical model with latent variables (Piccardi (2013))

Latent Structural SVM

Until now, we have dealt with fully supervised learning in which labels y are known prior to learning. In the recent years, the typical size of datasets has grown immensely which has made the task of annotating data more expensive than ever before. Therefore, latent variable models provide promising solutions to the case of unsupervised or semi-supervised learning where data labels are unknown or partially known. Moreover, latent variables can extract useful information from data that might be unobserved by the user.

In Latent SSVM (Latent-SSVM), some of the output variables are unknown during learning (semi-supervised learning). Let h be a set of latent variables and y the supervised labels, leading to a label set of supervised and unsupervised variables (y, h) respectively. This is depicted in Figure 2.9, where the set of hidden variables are shown in color. Yu and Joachims (2009) proposed the training of Structural SVM with latent variables h by incorporating h inside the linear discriminant function, $w^\top \Phi(x, y)$; thus, it becomes $w^\top \Phi(x, y, h)$. The authors posed the constraints for Latent-SSVM as follows:

$$\begin{aligned} &\forall \hat{y} \in Y \backslash y_i : \xi_i \geq \max_{\hat{h} \in H} \left[w^\top \Phi(x_i, \hat{y}, \hat{h}) + \triangle(y_i, \hat{y}, \hat{h}) \right] - \max_{\hat{h} \in H} w^\top \Phi(x_i, \hat{y}, \hat{h}) \\ &\xi_i \geq 0 \end{aligned} \tag{2.18}$$

Here, $\hat{h}$ is the predicted latent states from the weight vector of previous iteration. The inclusion of h inside our prediction function can either be marginalized or maximized. Yu and Joachims (2009) proposed to get rid of h by the operation of maximization instead of marginalization, because marginalizing h is a more expensive operation. With these changes, a constraint for example i (or value of ξ_i) can be written as follows:

$$\xi_i = \max_{(\hat{y},\hat{h})\in Y\times H} \Big[w^\top \Phi(x_i, \hat{y}, \hat{h}) + \triangle(y_i, \hat{y}, \hat{h})\Big] - \max_{h\in H} w^\top \Phi(x_i, y_i, h) \tag{2.19}$$

As a consequence, ξ becomes the difference of two convex terms (each term is actually a maximum of linear functions which is in turn convex). Therefore, the primal problem for Latent-SSVM becomes:

$$\begin{aligned} &\min_{w\in R^{M'},\xi\in R^N} \ \frac{1}{2}\left\|w\right\|^2 + C\sum_{i=1}^{N}\xi_i \\ &\forall \hat{y} \in Y\backslash y_i : \max_{h\in H} w^\top \Phi(x_i, y_i, h) - \max_{\hat{h}\in H} w^\top \Phi(x_i, \hat{y}, \hat{h}) \geq \triangle(y_i, \hat{y}, \hat{h}) - \xi_i \\ &\xi_i \geq 0 \qquad\qquad\qquad\qquad\qquad\qquad \forall i = 1, \ldots, N \end{aligned} \tag{2.20}$$

Since the Equation (2.19) involves convex and concave terms, the Objective (2.20) becomes non-convex. To alleviate this issue, Yu and Joachims (2009)) computes h_i^* in the following fashion:

$$h_i^* = \max_{h\in H} w^\top \Phi(x_i, y_i, h) \tag{2.21}$$

This is named as "latent variable completion problem" and imputes an upper bound on the concave part in Equation (2.19). After solving Equation (2.21), the objective for Latent-SSVM becomes convex because the second term turns linear in w. The

convexified version of Latent-SSVM is presented as follows:

$$\begin{aligned}&\min_{w\in R^{M'},\xi\in R^N} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\max_{(\hat{y},\hat{h})\in Y\times H}\left[w^\top\Phi(x_i,\hat{y},\hat{h}) + \triangle(y_i,\hat{y},\hat{h})\right] \\ &- C\sum_{i=1}^{N} w^\top\Phi(x_i,y_i,h_i^*)\end{aligned} \tag{2.22}$$

The above objective is solved using Structural SVM since the concave part of Equation (2.19) is replaced by its linear upper bound. The process of iterating between Equation (2.21) and Objective (2.20) is repeated until convergence which is guaranteed, and equivalent to the Convex-Concave Procedure (Yuille and Rangarajan (2003)).

2.6 Still Image Action Recognition

In the field of computer vision, "Action Recognition" is a well-known problem that has received great interest from researchers. Extensive research for the solution of this problem has given birth to several methods that can recognize human actions from videos (Poppe (2010)) and images (Guo and Lai (2014)). In the following discussion, we will restrict our focus to action recognition in images, that aims to recognize human actions from a single image. This problem is also known as "Still Action Recognition" or "Static Action Recognition".

Research in still action recognition gained popularity due to the fact that some actions can be recognized from a single image instead of videos. Figure 2.10(a) depicts a person playing a musical instrument and Figure 2.10(b) portrays a human riding a bike. It can be seen qualitatively that recognizing these two actions does not need temporal information over videos since these two images contain enough information for recognizing the respective actions. Which actions are suitable for still action recognition is a separate discussion outside the scope of this thesis. We will accept the fact that certain actions (Yao et al. (2011a); Gupta et al. (2009a); Sener et al. (2012)) are simple enough to be identified from just a single image. Guo and Lai (2014) present a survey of still action recognition that briefly highlights a significant sample of the literature and datasets to date.

(a) (b)

FIGURE 2.10: a) Action class: Playing a guitar (Yao et al. (2011a)) b) Action class: Riding a bike (Yao et al. (2011a)).

The following subsections contain a brief overview of learning action classifiers from a training set of images. We will conclude our discussion by highlighting some advantages of still image action recognition.

2.6.1 Learning an Action Recognition Classifier

Still action recognition in images can generally be approached into two distinct ways. Several other methods found in the literature can be seen as a combination of these two approaches: (i) by fitting articulated pose models to humans that estimate poses in images (this method is more suitable for pose estimation problem) and (ii) by computing a bag-of-feature representation of the image in which various features are calculated over the entire image (global features) or its local patches like superpixels or grids of particular pixel size (local features for instance HoG, Dense SIFT etc). The latter approach is more likely to capture the contextual information of the action being performed in the image.

To learn an action classifier, we need to represent the image content in terms of global or local features (action representation). Several learning algorithms can be applied over such representation of actions to obtain the desired classifier.

2.6.1.1 Action Representation

To learn actions from images, we need to represent the image by an appropriate set of features. Each image will be represented in terms of features which will be handed over to the learning algorithm. Such sets of features are generally divided into two categories: (i) Global and (ii) local features.

2.6.1.2 Global Features

Global features capture different aspects of actions from images; for instance, they can take into account human pose, arrangement of body parts, background of image and interaction of human with objects. They are computed over the regions of

interest. For instance, Wang et al. (2006) imputed feature set over human body for still action recognition. Again for the purpose of recognizing human actions, pose information from human body was extracted by Thurau and Hlaváč (2008) and Ikizler et al. (2008) since pose contains crucial information about undergoing actions. There are several other global features that serve different purposes for computer vision application.

2.6.1.3 Local Features

Local (or low-level) features are calculated over a fixed portions of the image. They are imputed over grids of fixed sizes of image pixels. Examples include the scale invariant feature transform (SIFT) (Lowe (2004)), histogram of gradients (HOG) (Dalal and Triggs (2005)) and shape context (Belongie et al. (2000)) to name a few.

SIFT (Lowe (2004)) is the most frequently used low-level feature in several computer vision applications. It has been widely used for object recognition, action recognition from videos and images, tracking objects, and many others. SIFT captures a descriptor vector (128D) at a location called keypoint. In SIFT, these keypoints are detected by the algorithm itself. In DSIFT (dense sampling of scale invariant feature points), keypoints are selected by the dense sampling of image pixels. After the selection of keypoint, a regular grid is placed around it to perform operations that achieve robustness against rotation, small illumination changes and scaling of the image.

2.6.1.4 Learning a Classifier

After representing the image and its content by the set of features, action classifiers can be learnt within a generative or discriminative learning framework. Structured prediction has become a popular approach for various applications in the field of computer vision. In this thesis, we have applied latent structural SVM (Yu and Joachims (2009)) for the task of learning human actions in still images.

2.6.2 Advantages of Still Image Action Recognition

Success in static action recognition will bring major benefits across several useful applications:

- Image annotation i.e. tagging images with corresponding actions has become an expensive task due to the extensive growth in the number of images on social media and internet. We can use reliable algorithms for still action recognition to reduce the cost of manual image annotation.
- Robots are entering our society and they need to interact with their surroundings. Since processing human actions from a single image is much cheaper than action recognition from videos, algorithms for still action recognition will be in high demand among robotic communities.
- Given the massive numbers of images present on the internet, search engines can greatly benefit from action labels.
- Still action recognition will aid solving other problems; for instance, "scene recognition" and "object recognition", to name a few.

These are quite a few useful applications that rely on the success in static action recognition. New applications that can benefit from static action recognition may emerge as the time moves on.

Chapter 3

Semi-Supervised Structured Prediction SVM and its Application for Static Action Recognition

3.1 Introduction

Automated recognition of actions in still images can play an important role for annotation of image catalogues, including the large collections of images which are increasingly made available by social networks. Actions which can be plausibly recognized from still imagery are those inferrable from the actors' poses and the presence of relevant objects: examples range from "waving hands" and "jumping" to "throwing a javelin" and "playing guitar". Moreover, recognition from single frames can also provide a means to recognize actions in video. For instance, in video surveillance it is not uncommon to clearly sight an actor for only a few frames due to repeated occlusions. In such cases, it is not easy to recognize the action in dynamical terms, i.e., as the temporal evolution of a measurement vector. Rather, inference must be obtained as the cumulative evidence from a (possibly small) set of individual frames. In robotics as well, the varying camera viewpoint may make

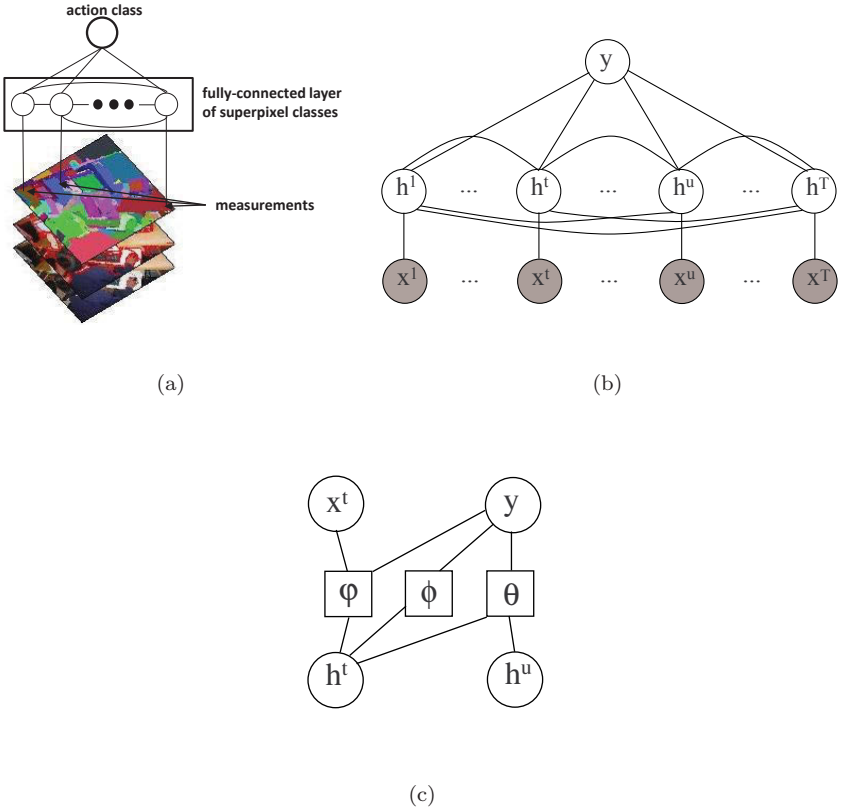


FIGURE 3.1: a) The proposed action recognition approach: bottom layer: superpixel segmentation and feature extraction; intermediate layer: superpixel classification; top variable: action class. b) The graphical model: x: superpixel measurements; h: superpixel classes, or states; y: action class. c) Factor graph representation: φ,ϕ and θ are the feature functions in (3.5).

it easier to recognize actions from isolated frames than from sequences. Estimation from still images is therefore a foundational technology for all these cases.

The most straightforward solution to recognize actions from still images is to compute a bag-of-feature representation of the image and use it for classifying it into a relevant set of action classes Delaitre et al. (2010); Ikizler et al. (2008); Laptev (2005). Useful features include local texture descriptors such as the histogram of oriented gradients (HOG), dense SIFT, GIST and several others Dalal and Triggs (2005); Lowe (2004); Oliva and Torralba (2001). Bag-of-features analysis usually discards the spatial coordinates at which descriptors are collected since it focuses on textural rather than spatial or structural information. These approaches have

reported very interesting results on challenging still image action datasets such as those described in Ikizler-Cinbis et al. (2009); Gupta et al. (2009a); Yao and Fei-Fei (2010). At the opposite end of the spectrum are approaches based on the explicit recovery of body parts and the incorporation of structural information in the recognition process Yang et al. (2010); Yao et al. (2011b). The baseline model is a latent part-based model akin to Pictorial Structure which can be estimated as a joint, conditional or large-margin model Fischler and Elschlager (1973); Felzenszwalb et al. (2010). It is important to note that the bag-of-features and pose estimation approaches tackle two different overarching problems: a) a bag-of-features approach recognizes the action by describing the image's content, potentially including relevant objects and the context of the action. It can also be used to recognize events with no actors or where actors are too small to allow for pose estimation such as in crowd analysis; b) on the other hand, approaches based on pose estimation are likely to prove more accurate for actions strongly characterized by posture such as "standing", "stretching" or the like. Delaitre *et al.* have reported a comparison between a bag-of-features and a structural approach, showing that hybridization of the two can be a way to capture the benefits of both models Delaitre et al. (2010). A recent survey from Guo and Lai offers a comprehensive outline of the research in this area Guo and Lai (2014).

It appears that the existing approaches have not explicitly explored the underlying relationship between the action class and the segments of the containing image. Such a relationship promises to contain useful information about the joint presence of the actor and revealing objects. For this reason, in this paper we propose to approach action recognition in still images by leveraging the latent classes of the image's "superpixels" (homogenous regions obtained from over-segmentation of the image Felzenszwalb and Huttenlocher (2004)). To this aim, we have designed a graphical model with the action as its root node and a fully-connected layer of superpixel classes to capture the relationship with the image segments (see Fig. 3.1(b)). A rich measurement vector is extracted from each superpixel, and model training is provided by a latent structural SVM approach. The layer of superpixels is fully connected for two reasons: a) the superpixels have irregular shape and the model cannot be a conventional lattice with fixed neighborhood; and b) any segment of the image is thus allowed to directly influence the classification of any other segment,

enforcing stronger correlation between parts. However, inference becomes NP-hard and therefore a main contribution of our paper is an efficient, greedy algorithm that provides approximate inference over the graph.

The rest of this chapter is organized as follows: Section (3.2) describes the graphical model for action recognition and the detectors for the superpixels' classes. Section (3.3) briefly reviews the latent structural SVM framework and describes the feature functions, the initialization heuristic and the greedy inference algorithm. Section (3.4) describes the experiments and discusses the results. Section 5 highlights conclusions and future work.

3.2 Action recognition by superpixel classification

The first step in our approach is the decomposition of an image into small and coherent patches commonly referred to as superpixels. Our underlying assumption is that certain actions can be recognized effectively by utilizing useful information about the superpixels. The superpixel segmentation of the image can be obtained with graph-based algorithms such as Ren and Malik (2003); Felzenszwalb and Huttenlocher (2004); Mori (2005) and many others. Superpixels are becoming increasingly popular in computer vision as an intermediate representation in-between image pixels and semantic-level segments.

Given their homogeneous nature, superpixels of natural images are likely to be assignable with single class labels describing the object they belong to. Fig. 2 shows an example of superpixels with objects such as body parts, books, computers and desks. The assignment of a superpixel to a class can be obtained by extracting a feature vector from the superpixel's region and applying any supervised classifier. However, a recent paper from Pei et al. (2013) has shown that superpixel classification proves more accurate if all the superpixels in the image are classified jointly. This stems from the fact that the relationship between superpixels is intuitively semantic rather than only spatial: a superpixel of class "sky" can be correlated to

a superpixel of class "road" even if they are not neighboring. To exploit this relationship, Pei et al. (2013) has proposed to approach superpixel segmentation by a fully-connected graph structure.

In this work, we speculate that the action depicted in an image should also be related to the superpixels' classes. To this aim, we extend the graphical model of Pei et al. (2013) with a variable representing the action class, and with edges between the action and all the superpixel classes and measurements. As measurements, we apply a set of pre-trained object detectors at each superpixel and we use their scores as the feature vector. Using sets of diverse object detectors as feature vectors has become a widespread approach in recent years and has led to remarkable accuracy improvements in action and scene recognition (e.g., Han et al. (2009); Yao et al. (2011a); Ullah et al. (2010)). In the rest of this section, we describe the graphical model and the object detectors in greater detail.

3.2.1 The graphical model

The proposed graphical model comprises three sets of variables, namely measurements (x), hidden nodes, or states, (h) and an output node (y). The measurements are a vector of detector scores for each superpixel, the hidden nodes are the superpixels' classes, and the output node is the action class. The nodes are connected by three different types of edges: a) edges connecting measurements and states, b) edges over state pairs, and c) edges between states and the action class. Noting as T_i the number of superpixel in the i-th image in a training set, we thus have the following variables: $x_i = [x_i^1, \ldots, x_i^t, \ldots, x_i^{T_i}]$, with x_i^t a multi-dimensional vector of detector scores; $h_i = [h_i^1, \ldots, h_i^t, \ldots, h_i^{T_i}]$, with $h_i^t \in \{1, \ldots, K\}$ a superpixel class; and $y_i \in \{0, 1\}$ a binary variable for a given action class. We build one such model for each class. At training time, action classes are supervised while states are hidden.

Generally, it is observed in semi-supervised learning for I.I.D datasets, an addition of unlabeled data result in the improvement of classification accuracy. In our case, we have a fully connected graph of latent (unlabeled) variables (h). Since the nature of latent variables across I.I.D dataset and our graphical model is inherently different,

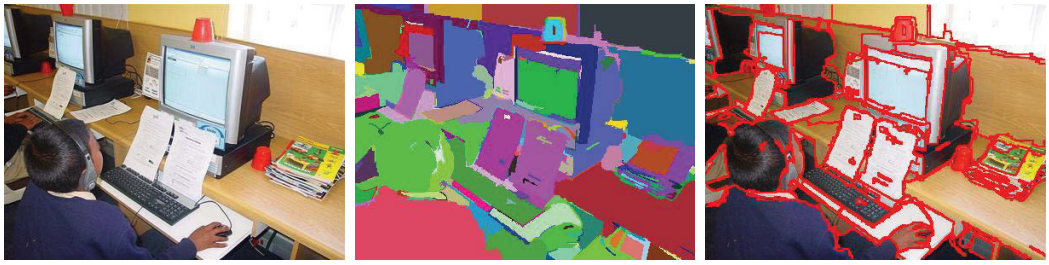
Symbol	Explanation
x	measurement vector of a generic image
x_i	measurement vector of the i-th image in the training set
x_i^t	measurement for the t-th superpixel of the i-th image
T_i	total number of superpixels in the i-th image
y	action class of a generic image
y_i	ground-truth class of the i-th image
h	vector with the classes of all superpixels of a generic image
h_i	vector with the classes of all superpixels for the i-th image
h_i^t	class of the t-th superpixel in the i-th image
K	number of possible classes for each superpixel
s_i^t	initial feature set for the t-th superpixel of the i-th image
w_k	weight vector for superpixel class k in the initial SVM multiclass classifier
w	weight vector of latent structural SVM
w_φ	section of w that scores the measurement features
w_θ	section of w that scores the state features
w_ϕ	section of w that scores the class features

TABLE 3.1: Main notations (notations valid for this chapter only).

therefore we can not expect trade-offs in the classification accuracy by augmenting unlabeled portion of data.

3.2.2 Object detectors

The first step of our processing pipeline is the decomposition of the image into superpixels. For this task, we use the efficient graph-based algorithm proposed by Felzenszwalb and Huttenlocher (2004) that was also used in Pei et al. (2013). This step achieves good over-segmentation of the image into regions of predominantly



(a) (b) (c)

FIGURE 3.2: Example of superpixel segmentation: a) original image; b) superpixel segmentation; c) superpixel boundaries highlighted.

homogeneous nature, and errors in this process can be tolerated by the ensuing soft-assignment stage.

As class set for the superpixels, we have decided to adopt the class set of the MSRC 21-Class dataset (MSRC-21), version 2, consisting of 23 diverse classes from typical background and foreground objects (Criminisi (2004)). The main benefit of this dataset is that all its images are accurately object-annotated at pixel level. As feature vector, we use the concatenation of an appearance-based and a bag-of-features descriptors. The appearance-based descriptor contains 51 features, consisting of: 1) 40 color features measuring mean, standard deviation, skewness and kurtosis of RGB, LAB, YCrCb color space channels and the gray-level image; and 2) 11 texture features obtained from the application of an average filter and five different responses from Gaussian and Laplacian-of-Gaussian filters. The bag-of-features descriptor is obtained by first computing dense SIFT descriptors (Vedaldi and Fulkerson (2008)) in the superpixel region at three different scales and then encoding the descriptors into a dictionary of 400 visual words learned by k-means clustering. We concatenate the appearance-based and the bag-of-features descriptors into an overall 451-D vector, noted as s_i^t for the t-th superpixel of the i-th image.

Once the feature vectors are extracted for all superpixels, they are used to compute class scores from trained object detectors. Rather than training separate object detectors, we have built a single, multi-class classifier by using multiclass SVM (Crammer and Singer (2002)) over MSRC-21. An advantage of a joint detector is that its

scores are naturally normalized across classes. In addition, we convert the scores to posterior probabilities and collect them as measurement x_i^t:

$$\left[x_i^t = p(k|s_i^t) \propto \exp w_k^\top s_i^t\right],\ k = 1 \dots K \tag{3.1}$$

where x_i^t are the posterior probabilities for the t-th superpixel of the i-th image and w_k denotes the k-th class' parameter vector of the SVM multiclass classifier. Using the output of this initial classifier as measurements exploits the semantic of the object classes and reduces the measurement dimensionality from 451-D to 23-D. In addition, in the graphical model the states are assumed to be in correspondence with the latent object classes and their number, K, is correspondingly set to 23. Please note that the dataset used for training the object classifier is an altogether separate dataset from the action dataset and that the object detectors will not be re-trained on the action dataset in any form.

3.3 Semi-supervised Latent structural SVM

For action inference, we use the following linear prediction function:

$$(\bar{y}, \bar{h}) = f_w(x) = \underset{y,h}{\operatorname{argmax}}[w^\top \psi(x, h, y)] \tag{3.2}$$

where $\psi(x, h, y)$ is a generalized feature function computing a combined map over measurements x, states h and action class y, and w is a corresponding parameter vector. The action class and the superpixels' classes are predicted jointly and typically only the predicted class, $\bar{y}$, is retained. For parameter estimation, we adopt the well-established latent structural SVM framework (Yu and Joachims (2009)). This is a regularized minimum-risk framework guaranteed to provide a local optimum for structural models with latent variables. Its learning objective:

$$
\begin{aligned}
w^* = \underset{w,\xi_{1:N}}{\operatorname{argmin}} \; & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\
s.t. \; & w^\top \psi(x_i, h_i^*, y_i) - w^\top \psi(x_i, h, y) \geq 1 - \xi_i \\
& \forall \{y, h\} \neq \{y_i, h_i^*\}
\end{aligned}
\tag{3.3}
$$

$$
h_i^* = \underset{h}{\operatorname{argmax}}\, w^{*\top} \psi(x_i, h, y_i) \tag{3.4}
$$

is an iterative objective that alternates between the constrained optimization in (3.3), performed using the current values for latent variables h_i^*, and a new assignment for h_i^* (3.4) from updated model w^*. Notations in (3.3) are as follows: w is the desired weight vector needed for classification; N is the total number of images in the training set; ξ_i is the slack variable associated with image i; y_i is the ground-truth class of image i; eventually, h_i^* is the current assignment for the superpixels' labels of image i. Implementation of (3.3) requires a number of design choices including the definition of a suitable feature function, $\psi(x, h, y)$, the initialization of latent variables h, and efficient algorithms for inference and augmented inference which are described in the following sub-sections.

3.3.1 Feature and score functions

The features in feature function $\psi(x, h, y)$ reflect the topology of the graphical model that includes an edge between each superpixel's measurement and its state variable, a fully-connected graph amongst states, and an edge between each state and the action class. In detail, $\psi(x, h, y)$ breaks into:

- *measurement features*, $\varphi(x^t, h^t = j, y = b)$: these features account for the compatibility between the measurement vector of the t-th superpixel, x^t (that is a K-dimensional vector as shown by (3.1)), its state h^t (possible values: $\{1 \dots K\}$) and class $y \in \{0, 1\}$. The size of this feature vector is $2K^2$. Given

$x^t, h^t = j$ and $y = b$, it consists of the values of x^t starting at index $K(j-1)+1$ (if $b = 1$) or $K^2 + K(j-1)+1$ (if $b = 0$); all other entries are filled with zeros;

- *state features*, $\theta(h^t = j, h^u = k, y = b)$: these features report the co-occurrence of states $h^t = j$ and $h^u = k$ for class $y = b$. The size of this feature vector is again $2K^2$, with a value 1 at index $K(j-1)+k$ (if $b = 1$) or $K^2 + K(j-1)+k$ (if $b = 0$); all other entries are filled with zeros;

- *class features*, $\phi(h^t = j, y = b)$: these features report the co-occurrence of action class $y = b$ and state $h^t = j$. The size of this feature vector is $2K$, with a value 1 at index j (when $b = 1$) or $K + j$ (when $b = 0$), and zeros elsewhere.

We refer the reader to Tsochantaridis et al. (2005b) for further details on feature maps. Given such feature vectors, the score function in (3.2) is computed as:

$$\begin{aligned} w^\top \psi(x, h, y) = &\sum_{t=1}^{T} w_\varphi^\top \varphi(x^t, h^t, y) \\ &+ \sum_{t=1}^{T} \sum_{\substack{u=1, \\ u \neq t}}^{T} w_\theta^\top \theta(h^t, h^u, y) + \sum_{t=1}^{T} w_\phi^\top \phi(h^t, y) \end{aligned} \tag{3.5}$$

where $w^\top = \left[w_\varphi^\top \, w_\theta^\top \, w_\phi^\top \right]$ is the concatenation of the parameter vectors for the corresponding features.

3.3.2 Loss-augmented inference

Following the structured learning approach of Yu and Joachims (2009), a fundamental step in the learning procedure is the computation of a loss-augmented version of the inference. As loss function, we simply use the 0-1 loss function:

$$\triangle(y^{gt}, y) = \begin{cases} 1 & \text{if } y \neq y^{gt} \\ 0 & \text{if } y = y^{gt} \end{cases}$$

where y^{gt} represents the ground-truth label. With this choice, it can be easily seen that the loss-augmented inference:

$$(\bar{y}, \bar{h}) = \underset{y,h}{\operatorname{argmax}} \left[w^\top \psi(x, h, y) + \triangle(y^{gt}, y) \right] \tag{3.6}$$

is equivalent to the standard inference in (3.2) with the addition of a unit score over the incorrect class.

3.3.3 Latent variables' initialization

The learning procedure in (3.3-3.4) can be initialized by either an arbitrary vector w^* in (3.4) or an arbitrary assignment for the h_i^* in (3.3). Given that we have trained a multiclass superpixel classifier to provide the measurements, the most natural choice is to initialize the states with the prediction from this classifier:

$$h_{i_{init}}^{*t} = \underset{k}{\operatorname{argmax}}[p(k|s_i^t)] \tag{3.7}$$

The above is equivalent to initializing the states with individual predictions, reserving their joint inference to the following structural inference steps.

3.3.4 Inference by efficient greedy algorithms

The inference in (3.2) is over a fully-connected graph and cannot be computed exactly (Cooper (1990)). Similarly to Pei et al. (2013), we resort to a greedy algorithm that infers every state in turn from the marginal scores:

$$\bar{h}^t = \underset{h^t}{\operatorname{argmax}}[w^\top \psi(x, h^t, h^{\setminus t}, y)] \tag{3.8}$$

The above inference is computed for each superpixel in the image for a number of iterations (set to four in our experiments) to obtain an acceptable approximation of the joint inference. The inference over $y \in \{0, 1\}$ is instead simply computed explicitly. The pseudocode of the greedy inference is shown in Algorithm 1.

Algorithm 1 Greedy inference

MaxIter=4
Input: Measurement variables x
Output: Inferred action class $\bar{y}$
for $y = 0, 1$ **do**
    Initialize $\bar{h}$ using (3.7)
    repeat
        for $t = 1, \ldots, T$ **do**
            $\bar{h}^t = \operatorname{argmax}_{h^t} w^\top \psi(x, h^t, \bar{h}^{\setminus t}, y)$ (3.8)
        end for
    until MaxIter
    $h^y \leftarrow \bar{h}$
end for
$\bar{y} = \operatorname{argmax}_y w^\top \psi(x, h^y, y)$

At a first attempt, we tried to implement (3.8) directly, but learning times proved immediately impractical (exceeding weeks on a high-performance workstation). Naïve computation of (3.8) requires $K(2T + T^2)$ addenda, with $K = 23$ and T typically in the order of a few hundreds. This has to be repeated for every state, for every iteration of the greedy inference and for every internal iteration of the SVM solver. We thus introduced a first simplification by only computing the terms required by the maximization:

$$\begin{aligned}\bar{h}^t &= \underset{h^t=1:K}{\operatorname{argmax}}[w^\top \psi(x, h^t, h^{\setminus t}, y)] \\ &= \underset{h^t=1:K}{\operatorname{argmax}}[w_\varphi^\top \varphi(x^t, h^t, y) \\ &+ \sum_{u=1, u\neq t}^{T} w_\theta^\top(\theta(h^t, h^u, y) + \theta(h^u, h^t, y)) + w_\phi^\top \phi(h^t, y)]\end{aligned} \tag{3.9}$$

This simplification requires only $K(2+2T)$ addenda per state, abating the complexity from quadratic to linear in the number of superpixels. A further major reduction in computational cost is achieved by caching the state scores as follows:

Algorithm	Time (s)
Naïve ($O(T^2)$)	0.542
First simplification ($O(T)$)	0.480
Proposed (caching) ($O(1)$)	**0.002**

TABLE 3.2: Time in seconds for the various greedy inference algorithms (loop at lines 7-9 in Algorithm 1).

$$\begin{aligned}&\sum_{u=1, u\neq t}^{T} w_\theta^\top \theta(h^t = k, h^u, y) \\ &= \underbrace{\sum_{u=1, u\neq t-1}^{T} w_\theta^\top \theta(h^{t-1} = k, h^u, y)}_{\text{cached during the previous maximization}} \\ &- w_\theta^\top \theta(h^{t-1} = k, h^t, y) + w_\theta^\top \theta(h^t = k, h^{t-1}, y)\end{aligned} \tag{3.10}$$

since the scores in (3.10) only depends on the values of the states, not on their ordinal indices. Thanks to this second modification, the number of addenda required to evaluate (3.9) becomes constant (i.e., $O(1)$) in the number of superpixels, permitting a drastic decrease in computational time.

Table 2 provides a comparison of actual execution times for the inference algorithms (loop at lines 7-9 in Algorithm 1) for an image with 178 superpixels and a PC with

a 3.4 GHz CPU and 32 GB of memory. The naïve algorithm ($O(T^2)$ complexity) takes 0.542 s, while the first simplification ($O(T)$ complexity) mildly reduces the execution time to 0.480 s. The proposed algorithm ($O(1)$) instead drastically abates the execution time to only 0.002 s.

3.4 Experimental results

We have evaluated the proposed approach on the most challenging static action recognition dataset released to date, Stanford 40 Actions (Stanford-40) (Yao et al. (2011a)). This dataset contains images of humans performing 40 different classes of actions, including visually-challenging cases such as "fixing a bike" versus "riding a bike" or "phoning" versus "texting message"; the full class list is provided in the annotation of Fig. 3.5. The number of samples per class varies between 180 and 300, for a total of $9,532$ images. A standard training/test split is made available by the authors on their website, selecting 100 images from each class for training and leaving the remaining for testing.

To recognize the actions of Stanford-40, we have trained a model (3.3, 3.4) for each action class. However, training by using all the available training samples (that is, for each model, 100 positive and $3,900$ negative samples, respectively) is still heavily time-consuming. Therefore, we have decided to sub-sample the negative training samples by choosing the first 15 training images from each of the 39 negative classes. Parameter C in (3.3) was set to 1. As software, we have used Joachims' latent structural SVM solver [1] with convergence threshold $\epsilon = 0.1$. For the multiclass SVM classifier we have used a linear kernel, $C = 10$ and convergence threshold $\epsilon = 0.1$, scaling the 451-D feature vectors in the training set between -1 and 1. Eventually, for the superpixel segmentation we have used smoothing factor $\sigma = 0.5$, threshold $K = 500$ and a minimum superpixel size of 20 pixels. All our software has been uploaded as Supplementary Material of this submission.

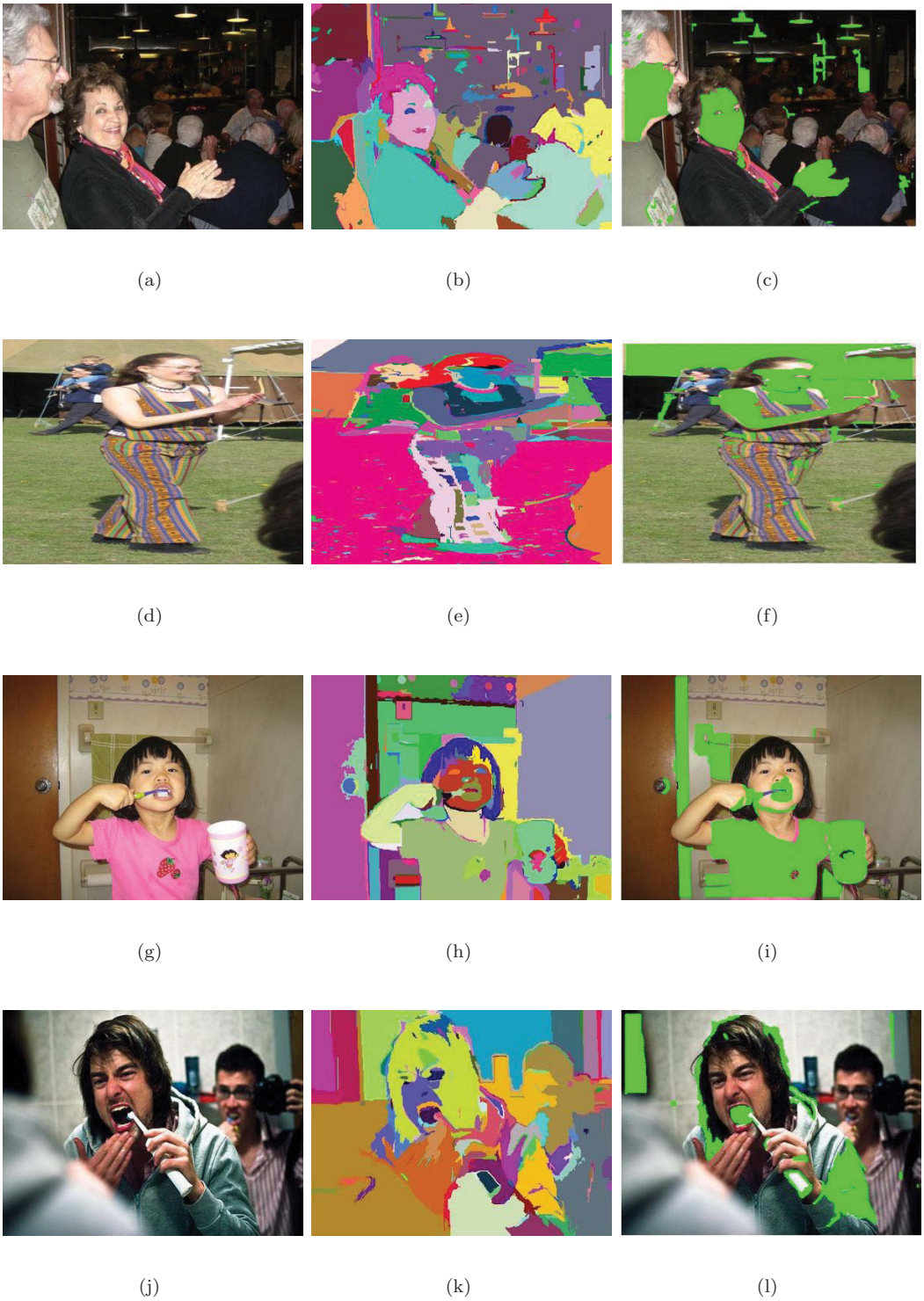
As term of comparison, we consider the most recent papers to have used this dataset (Yao et al. (2011a); Sharma et al. (2013); Khan et al. (2013); Sener et al.

[1] Downloadable from: http://svmlight.joachims.org/svm_struct.html

Method	mAP
EPM (Sharma et al. (2013))	42.2%
Sparse Bases (Yao et al. (2011a))	45.7%
Color Action Recognition (Khan et al. (2013))	51.9%
Multiple Instance Learning (Sener et al. (2012))	55.6%
Ours	**72.3%**

TABLE 3.3: Comparison of mean average precision on Stanford-40.

(2012)). The authors of Yao et al. (2011a) represented actions in still images by sparse bases of attributes, objects and poselets. Such bases are obtained through pre-trained detectors for 81 objects, 45 attributes and 150 poselets. A part-based approach was proposed by Sharma et al. (2013) that mined objects' parts and their locations in training images. Reference Khan et al. (2013) approached action recognition by leveraging color information from the actor and nearby objects. Reference Sener et al. (2012) used multiple instance learning to learn objects relevant to an action from a noisy set of candidate objects. As customary for this dataset, Table 3 reports the accuracy for the compared approaches in terms of mean average precision (mAP). The proposed approach reports an mAP of 72.3%, higher than the second best (55.6%, Sener et al. (2012)) by over 16 percentage points. For a break-down, Fig. 3.5 shows the average precision by class. The achieved accuracy can be regarded as remarkable also considering that the 23 object detectors were trained on a completely separate dataset and from classes mostly unrelated with the objects portrayed in Stanford-40. The evaluation protocol of Stanford-40 does not specify whether training can avail of cross-training or should instead be restricted to the training set alone (unlike, for instance, the PASCAL Visual Object Classes Challenge which separates the two cases). However, pre-training of detectors has become a common practice in computer vision (see, for instance, Han et al. (2009); Yao et al. (2011a); Ullah et al. (2010)). In addition, the annotation of Stanford 40 Actions also provides the actors' bounding boxes which we did not need to use with the proposed approach. In fact, our approach provides action recognition based on the superpixels' classes of the entire image, and therefore suits images with context. Conversely, it does not suit primarily postural datasets where the actor is depicted against a non-informative background such as HumanEva (Sigal et al. (2010)).



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

FIGURE 3.3: Examples of the top 20% superpixels contributing to the action score for classes *applauding*, *brushing teeth*, *gardening* and *waving hands* of Stanford 40 Actions. The figure shows triplets of {original image, superpixel decomposition, top-20 pixels highlighted} as a continuous sequence. This figure should be viewed in color.

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

FIGURE 3.4: (continued)

To further illustrate these results, we visualize the correspondence between the action and its "main" superpixels in Fig. 3. Given that the score for the action is provided by a summation over all the superpixels (3.5), they can be sorted in ascending order based on their individual contribution. Fig. 3 shows various examples of the top 20% superpixels displayed onto their original image, showing a remarkable correspondence with the most telling objects of the action.

As a validation experiment, we have tested our model with another static action dataset from Gupta et al. (2009b). This dataset contains 300 images from 6 sport action classes: batting (cricket), bowling (cricket), serve (tennis), forehand (tennis), serve (volleyball), and shot (croquet). We have adopted its usual training/test split with the first 30 images from each class for training and the remaining 20 for testing. Without re-training the object detectors, we have obtained an interesting mean average precision of 79.2% with a minimum average precision of 41.0% for class bowling and a maximum of 96.7% for class forehand. This shows that the trained object detectors are sufficiently generic to achieve good accuracy on other datasets. In all cases, more universal or more specialized object detectors can prove a useful alternative to further improve accuracy for given datasets.

3.5 Conclusion

We have proposed an approach to static action recognition that leverages the relationships between the classes of the image's superpixels. The approach consists of two stages: in the first stage, the image is segmented into a set of superpixels and a set of trained object detectors is applied to each superpixel to extract a vector of detector scores. In a second stage, the score vectors are used as measurements in a graphical model that jointly predicts the superpixels' classes together with the action class. A main contribution of our approach is an efficient greedy algorithm that provides approximate inference over the fully-connected graph of the superpixels' classes, decreasing the computational complexity per superpixel from the $O(T^2)$ of a naïve implementation down to $O(1)$. Experiments conducted over the highly challenging Stanford 40 Actions dataset have achieved an mAP of 72.3%, the highest by far to date. This result gives evidence to the existence of a useful relationship

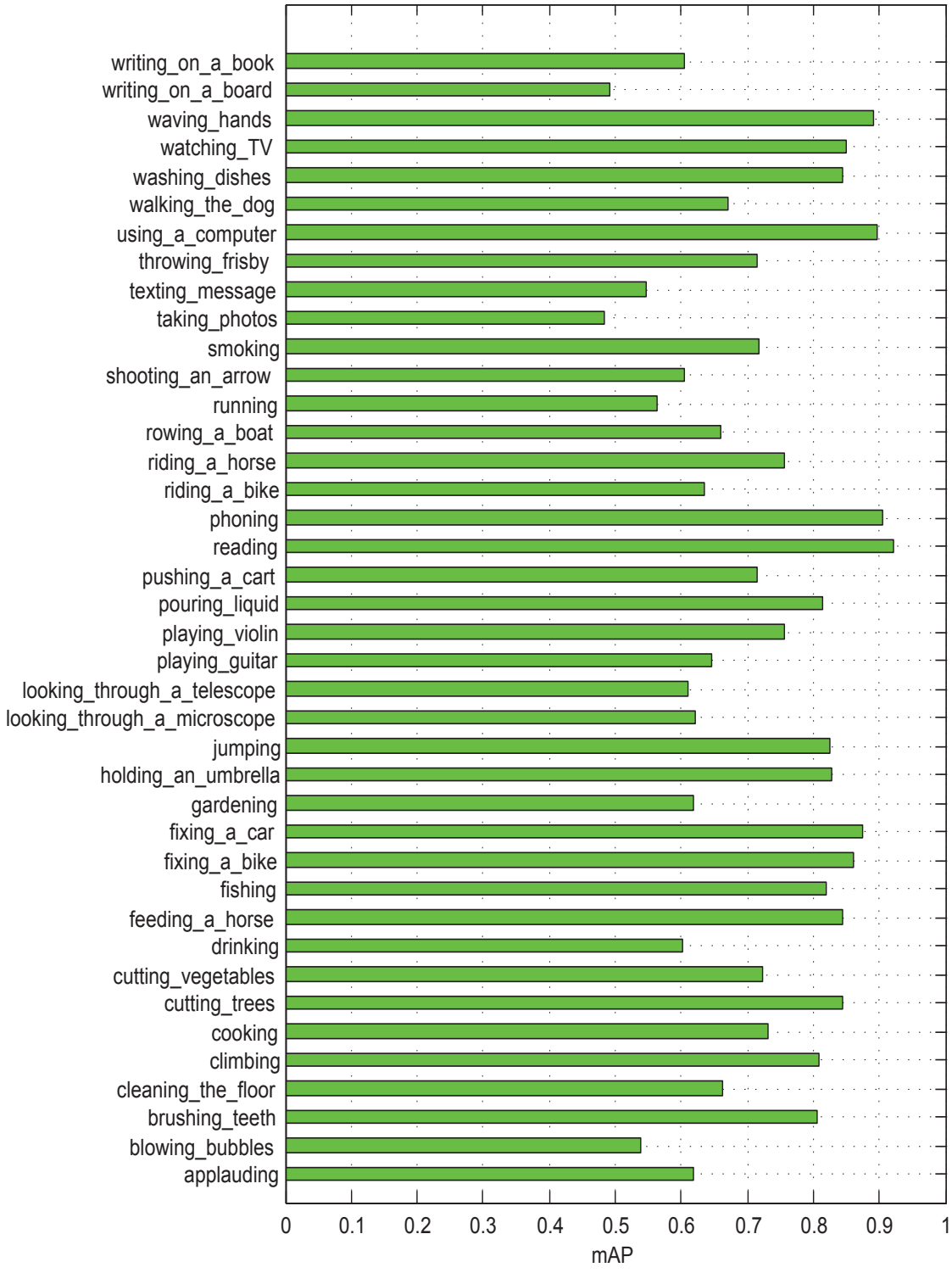


FIGURE 3.5: Average precision achieved by the proposed method in each class of Stanford 40 Actions.

between the classes of the superpixels and that of the main action. An obvious way to further improve the performance of the proposed model would be to adopt a larger set of object detectors or a set of detectors specifically tuned in the object classes of Stanford-40. Another advantage of the proposed approach is that its graphical model is very general and could therefore be used for other classification tasks that rely on latent variables including, among others, scene recognition and complex event detection.

Chapter 4

Unsupervised Structured Prediction SVM

4.1 Introduction

Clustering algorithms group unlabeled data according to chosen measures of similarity. For independent data, k-means is perhaps the most well-known and common approach. However, in recent years, maximum-margin approaches to clustering have attracted increasing attention. The central premise is to pursue the maximization of a margin objective in an unsupervised manner (Vapnik (1998)). The challenge with this task is that the number of possible labelings is exponential in the number of samples and the problem is notoriously NP-hard. Therefore, solutions must rely on either local optima or relaxations of the original problem. The basic local approach consists of alternating optimizations where a step of SVM learning gives turns to a step of inference on the latent variables (Vapnik (1998); Felzenszwalb et al. (2008)). While local solutions are computationally efficient, they tend to show early convergence and have been reported as highly sensitive to the initialization (Schwing et al. (2012)). A different support vector approach was proposed in (Ben-Hur et al. (2001)) in terms of minimum-enclosing hyperspheres. Determining the minimum-enclosing hypersphere of a set of points requires solving a quadratic objective with quadratic, positive-semidefinite constraints and it is therefore a convex problem.

However, assigning the points to clusters remains a combinatorial problem and the overall solution is still only locally optimal.

Max-margin approaches have also attracted increasing attention for unsupervised structured prediction. Latent structural SVM (Latent SSVM) (Yu and Joachims (2009)) was proposed as an extension of structural SVM (Tsochantaridis et al. (2005a)) for structured problems with latent variables. It has been extensively adopted in computer vision for problems of object detection, pose estimation and activity recognition, and also in natural language processing, bioinformatics and various other fields (Felzenszwalb et al. (2008); Zhu et al. (2010); Wang and Mori (2011); Duan et al. (2012); Yu and Joachims (2009)). Paralleling other local methods, it is based on alternating a step of inference with a step of training under the inferred values. Therefore, the objective of Latent SSVM is also naturally non-convex.

Given that combinatorial approaches are inherently local, exploring convex relaxations offers an appealing alternative. Xu et al. (2005) have proposed a convex relaxation of unsupervised SVM based on semidefinite programming (SDP), and have later extended it to the multi-class and structured cases (Xu and Schuurmans (2005); Xu et al. (2006)). However, the computational complexity of SDP is much higher than that of quadratic programming and seems prohibitive even for datasets of limited size. More recently, Li et al. (2009, 2013) have proposed an alternative relaxation based on the minimax theorem. The relaxation, called Weakly Labeled SVM or Well-SVM for short, has proven to be tighter than the SDP relaxation of Xu et al. (2005). In addition, it can be implemented in terms of multi-kernel SVM and is efficient and scalable. For this reason, we hereby design and evaluate its extension for the purposes of structured prediction. Like the original relaxation, its structured extension can effectively and efficiently avail of existing solvers. The rest of this chapter is organized as follows: the remainder of this section summarizes Well-SVM and structural SVM to set the ground for the proposed extension. The extension is presented in Section 2, while experimental results are described in Section 3. The Conclusion summarizes the main findings.

4.1.1 Well-SVM

Let us note the constrained objective of a conventional, soft-margin SVM as $SVM(w, y)$, with w the parameter vector and y the labeling for the sample set. The problem entailed by unsupervised SVM is:

$$\min_{y} \min_{w} SVM(w, y) \tag{4.1}$$

that is, finding the labeling returning the minimum of all SVM primal problems $\min_w SVM(w, y)$. This equates to finding the two clusters with the maximum soft-margin between them. However, this formulation may degenerate in assigning all the samples to only one cluster with a hypothetically infinite margin. It is then necessary to limit the search over y to a set of feasible, "balanced" labelings which we will note as β hereafter, i.e., $y \in \beta$. An equivalent solution to (4.1) can be obtained by replacing the internal minimization with its dual:

$$\min_{y} \max_{\alpha} G(\alpha, y) \tag{4.2}$$

where α is the vector of the dual variables. The idea behind the Well-SVM relaxation Li et al. (2009, 2013) is to exchange the order of $\max_\alpha$ and $\min_y$ to instead solve:

$$\max_{\alpha} \min_{y} G(\alpha, y) \tag{4.3}$$

Li et al. (2009) have proved that this relaxation is tighter than an earlier relaxation based on semidefinite programming proposed by Xu et al. (2005). On this ground, Well-SVM can be regarded as the tightest known relaxation of unsupervised SVM.

The inner combinatorial minimization in (4.3) can be posed in terms of a continuous, constrained maximization:

$$\begin{aligned} &\max_{\theta} \ \theta \\ &s.t. \ \theta \leq G(\alpha, y^l) \quad \forall y^l \in \beta \end{aligned} \tag{4.4}$$

with corresponding Lagrangian:

$$\theta + \sum_{l:y^l \in \beta} \mu^l \left(G(\alpha, y^l) - \theta \right) \tag{4.5}$$

where we have chosen to explicitly enumerate the labelings to be able to refer to them and their corresponding multipliers by a common apex. Differentiating the Lagrangian in θ, equating to zero and replacing the result in (4.3) lead to:

$$\max_{\alpha} \min_{\mu} \sum_{l:y^l \in \beta} \mu^l G(\alpha, y^l) \tag{4.6}$$

subject to constraints $\sum_l \mu^l = 1, \mu^l \geq 0 \ \forall l$, and with μ the vector of all μ^l's. Since (4.6) meets the KKT conditions, the max-min order can be swapped to obtain:

$$\min_{\mu} \max_{\alpha} \sum_{l:y^l \in \beta} \mu^l G(\alpha, y^l) \tag{4.7}$$

Li et al. (2013) have shown that (4.7) can be solved as an instance of multi-kernel learning (MKL) and by using conventional SVM solvers. The solution is obtained by iterating the following three steps until convergence:

1. Find a labeling, y^l, violating constraint (4.4) for the current α. Add y^l to a working set of labelings, C_w. The procedure for finding a violating labeling is presented in Section 2. For the first iteration, an arbitrary labeling y^l can be chosen as the initial working set.

2. Using the current working set, solve for α. This solution can be framed as an instance of multi-kernel learning SVM.

3. Solve for μ^l in closed form. Details are also provided in Section 2.

4.1.2 Structural SVM

Structural, or structured-output, SVM extends the conventional support vector machine to the classification of multiple, dependent classes. Given one or multiple measurements in input, noted as x hereafter, a prediction or labeling consists of a graph of class labels, y. Structural SVM uses a generalized feature function, $\psi(x, y)$, to capture the structure in the data, and imposes a given margin between the score assigned to the ground-truth labeling and the score assigned to any other labeling. Assuming a training set $\{x_i, y_i\}, i = 1 \dots N$, the primal problem is expressed as:

$$\begin{aligned} &\min_{w,\xi} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ &s.t. \ w^T\psi(x_i, y_i) - w^T\psi(x_i, y) \geq \Delta(y_i, y) - \xi_i \\ &\qquad \forall i, \forall y \in \mathcal{Y} \end{aligned} \tag{4.8}$$

The use of an arbitrary loss function, $\Delta(y_i, y)$, in the constraints - known as margin re-scaling - allows structural SVM to potentially minimize any empirical loss. The most common choices are the Hamming loss or other losses that decompose over the single variables of the graph. Moreover, by assuming that $\Delta(y_i, y_i) = 0$, the required non-negativity constraint on ξ_i is included. However, the number of constraints per sample, $|\mathcal{Y}|$, is exponential in the number of classes and exact solution of (4.8) is generally impossible. Therefore, structural SVM relaxes this problem by only

considering a sub-set of the constraints. At each iteration of the solver and for each sample, a search is conducted for the labeling that sets the value of ξ_i (the most violating labeling):

$$\xi_i = \max_y \left(-w^T\psi(x_i, y_i) + w^T\psi(x_i, y) + \Delta(y_i, y)\right) \tag{4.9}$$

$$\rightarrow \quad y_i^* = \operatorname*{argmax}_y \left(w^T\psi(x_i, y) + \Delta(y_i, y)\right) \quad \forall i \tag{4.10}$$

If the value of ξ_i is larger than its previous value by a chosen ϵ, labeling y_i^* is added to a working set of constraints for the sample, W_i. The inference in (4.9) is commonly referred to as loss-augmented inference and is resolved efficiently by leveraging the structure in the output. Tsochantaridis et al. (2005a) have proved that the solution is ϵ-close to that of the original problem and that the size of the working set is only polynomial. The relaxed dual problem can be obtained with a standard derivation (Tsochantaridis et al. (2005a)). By posing $\delta\psi_i(u) \equiv \psi(x_i, y_i) - \psi(x_i, u)$, the dual problem can be expressed as:

$$\begin{aligned} &\max_\alpha G(\alpha, y, W) = \max_\alpha -\frac{1}{2}\sum_{i,u\in W_i}\sum_{j,v\in W_j}\alpha_{i,u}\alpha_{j,v}\delta\psi_i(u)^T\delta\psi_j(v) + \sum_{i,u\in W_i}\alpha_{i,u}\Delta(y_i, u) \\ &s.t.\ 0 \leq \sum_{u\in W_i}\alpha_{i,u} \leq C, \quad i = 1, \ldots, N \end{aligned} \tag{4.11}$$

4.2 Weakly Labeled Structural SVM

The merging of structural SVM with Well-SVM is obtained by replacing the dual equation in (4.7) with the structural dual:

$$\min_\mu \max_\alpha \sum_{l:y^l\in\beta} \mu^l G(\alpha, y^l, W) \tag{4.12}$$

This problem is an instance of multi-kernel learning (MKL) through the multiple ground-truth labelings, $y_i^l, l = 1 \dots L$, and can be solved by any MKL algorithm. Li et al. (2013) have proposed using the *multiple kernel learning by group lasso* (MKLGL) which alternates solving for w, ξ at fixed μ, with solving for μ at fixed w, ξ. The above formulation also offers an intuitive view of Well-SVM compared to methods such as k-means and latent SVM: in the latter, at every iteration the newly inferred latent variables replace the previous. In Well-SVM, instead, all the labelings inferred up to the current iteration, $y^l, l = 1 \dots L$, are used jointly in the optimization by way of an individual weight, μ^l.

By merging (4.8) into (4.12), we obtain:

$$\begin{aligned} &\min_{\mu,w,\xi} \ \frac{1}{2}\sum_{l=1}^{L}\frac{\|w^l\|^2}{\mu^l} + C\sum_{i=1}^{N}\xi_i \\ &s.t. \ \sum_{l=1}^{L}\Big({w^l}^T\psi(x_i, y_i^l) - {w^l}^T\psi(x_i, y_i^*)\Big) \geq \sum_{l=1}^{L}\mu^l\Delta(y_i^l, y_i^*) - \xi_i \quad \forall i, \forall y_i^* \in W_i \end{aligned} \tag{4.13}$$

Every sample now has multiple ground-truth labelings and the score is obtained as the sum over the labelings. It can also be easily proven that the margin is set by the weighted sum of the loss of each labeling. The minimum in w, ξ in (4.13) can be found by conventional structural SVM solvers through the following positions:

- $\tilde{w} = [\frac{w^1}{\sqrt{\mu^1}}, \dots, \frac{w^L}{\sqrt{\mu^L}}]^T$: the concatenation of L, scaled parameter vectors;
- $\tilde{\psi}(x, y^{1:L}) = [\sqrt{\mu^1}\psi(x, y^1), \dots, \sqrt{\mu^L}\psi(x, y^L)]^T$, with $y^{1:L}$ the concatenation of L labelings;
- $\tilde{\psi}(x, y^{(\times L)}) = [\sqrt{\mu^1}\psi(x, y), \dots, \sqrt{\mu^L}\psi(x, y)]^T$, where $y^{(\times L)}$ denotes the concatenation of L copies of labeling y;
- $\tilde{\Delta}(y^{1:L}, y) = \sum_{l=1}^{L}\mu^l\Delta(y^l, y)$: the loss of prediction y versus labelings $y^{1:L}$.

With the above positions, the primal objective (4.13) can be re-written as follows:

$$\begin{aligned} &\min_{\tilde{w},\xi} \ \frac{1}{2}\|\tilde{w}\|^2 + C\sum_{i=1}^{N}\xi_i \ \ s.t. \\ &\tilde{w}^T\tilde{\psi}(x_i, y_i^{1:L}) - \tilde{w}^T\tilde{\psi}(x_i, y_i^{*(\times L)}) \geq \tilde{\Delta}(y_i^{1:L}, y_i^*) - \xi_i \qquad \forall i, \forall y_i^* \in W_i \end{aligned} \tag{4.14}$$

which is formally identical to a single structural SVM problem and can therefore be solved with any structural SVM solver. Please note that if loss $\Delta(y^l, y)$ is decomposable over the structure, so is loss $\tilde{\Delta}(y_i^{1:L}, y)$.

Using similar notations to (4.11), it can be shown that the dual of (4.14) is:

$$\begin{aligned} &\max_{\alpha} \sum_{l=1}^{L} \mu^l G(\alpha, y^l, W) = \\ &\max_{\alpha} \ -\frac{1}{2}\sum_{i,u\in W_i}\sum_{j,v\in W_j}\alpha_{i,u}\alpha_{j,v}\tilde{\delta\psi}_i(u^{(\times L)})^T\tilde{\delta\psi}_j(v^{(\times L)}) + \sum_{i,u\in W_i}\alpha_{i,u}\tilde{\Delta}(y_i^{1:L}, u) \\ &s.t. \ 0 \leq \sum_{u\in W_i}\alpha_{i,u} \leq C \quad \forall i \end{aligned} \tag{4.15}$$

where the scalar products $\tilde{\delta\psi}^T\tilde{\delta\psi}$ and loss $\tilde{\Delta}(y_i^{1:L}, u)$ contain the μ factor. The proof is provided in Appendix (B).

Once the solution for (4.14) is obtained, the vector of multipliers, μ, can be computed in closed form as follows:

$$\mu^l = \frac{\|w^l\|}{\sum_{l=1}^{L}\|w^l\|} = \frac{\sqrt{\mu^l}\|\tilde{w}^l\|}{\sum_{l=1}^{L}\sqrt{\mu^l}\|\tilde{w}^l\|}, \ \ l = 1 \ldots L \tag{4.16}$$

4.2.1 Feature Maps

The first challenge in approaching (4.15) is that the dependence of the kernel matrix on the ground-truth labels must be made explicit. To this aim, we re-code labels y

in terms of 1-out-of-N encoding. For the binary variable of a node, we use encoding [01] for the negative class and [10] for the positive class. For an edge, we use encoding [0001] if the edge connects a negative node with another negative node; encoding [0010] for an edge connecting a negative and a positive node; and so forth. Such an encoding could also easily accommodate multi-class nodes at the cost of a significant increase in size. However, the increase is mainly notational and an optimized implementation can efficiently support a multi-class extension.

For a graph with T binary nodes and E pairwise edges, there will be $2T$ binary variables to encode the nodes (*unary* variables) and $4E$ binary variables for the edges (*pairwise* variables), for a total of $2T+4E$ binary variables that we concatenate into y. Index $t = 1 \dots T$ will be used to index the nodes and index $e = 1 \dots E$ for the edges. Without the limitation of generality, we also assume that there is one measurement per node, x_t, and that the feature function for a node-measurement pair is of the type $x_t y_t$. Our first aim is to rewrite the feature function, $\psi(x, y)$, as follows:

$$\psi(x, y) = h^T y \tag{4.17}$$

Assuming measurement x_t to be D-dimensional, we have that $\psi(x, y)$ is overall a $(2D+4)$-dimensional vector organized as two D-dimensional parameter vectors, one per class, and four pairwise parameters, one per edge type. Hence, matrix h is $(2T + 4E) \times (2D + 4)$. Fig. 4.1 visualizes $\psi(x, y)$, h and y .

Given that the SSVM dual contains products of feature vectors, we need to express such products based on (4.17). Let us have two graphs with labelings y_i and y_j, of size $(2T_i + 4E_i)$ and $(2T_j + 4E_j)$, respectively. From (4.17), it follows that:

$$\psi(x_i, y_i)^T \psi(x_j, y_j) = y_i^T (h_i h_j^T) y_j = y_i^T \, H_{ij} \, y_j \tag{4.18}$$

Matrix H_{ij} is thus $(2T_i + 4E_i) \times (2T_j + 4E_j)$ (it is not square since the two graphs are generally not equal in size). Obviously, $H_{ij} = H_{ji}^T$.

The next requirement is to express the loss function, $\Delta(y_i, u)$, as an explicit function of y. Hereafter we assume the loss to be the common Hamming loss which is conveniently decomposable over the graph. It can be expressed as a function of y as:

$$\Delta(y_i, u) = y_i^T(\mathbf{1} - 2u) + u^T\mathbf{1} \tag{4.19}$$

where $\mathbf{1}$ is a vector of all 1's.

4.2.2 Finding a Violating Labeling

By now posing $y = [y_1^T, y_2^T \dots y_N^T]^T$ as the concatenation of the ground-truth labelings of the entire training set, we are ready to express $G(\alpha, y, W)$ as a quadratic function of the labelings:

$$\begin{aligned} &G(\alpha, y, W) = \\ &\sum_i \sum_j y_i^T \left(-\frac{1}{2} \sum_{u \in W_i} \sum_{v \in W_j} \alpha_{i,u}\alpha_{j,v}H_{ij} \right) y_j + \sum_i y_i^T \left(\sum_{u \in W_i} \sum_{j, v \in W_j} \alpha_{i,u}\alpha_{j,v}H_{ij}v \right) \\ &+ \sum_i y_i^T \left(\sum_{u \in W_i} \alpha_{i,u}(\mathbf{1} - 2u) \right) + constants \end{aligned} \tag{4.20}$$

Now, ignoring constant terms and making the following positions:

- $\alpha_i = \sum_{u \in W_i} \alpha_{i,u}$;
- $\boldsymbol{H} = \frac{1}{2}[\alpha_1\alpha_1 H_{11}, \alpha_1\alpha_2 H_{12} \dots \alpha_1\alpha_N H_{1N};$
  $\dots;$

$$\alpha_N\alpha_1 H_{N1}, \alpha_N\alpha_2 H_{N2} \dots \alpha_N\alpha_N H_{NN}]$$

(please note that matrix $\boldsymbol{H}$ is positive semidefinite by construction. Its row dimension, M, is given by the sum of the labeling size over the entire training set: $M = \sum_i 2T_i + 4E_i$)

- $\tau = -\sum_j [\alpha_1 H_{1j} (\sum_{v \in W_j} \alpha_{j,v} v)^T \dots \alpha_N H_{Nj} (\sum_{v \in W_j} \alpha_{j,v} v)^T]^T$;
- $\Delta = -[\sum_{u \in W_1} \alpha_{1,u} (\mathbf{1} - 2u)^T, \dots, \sum_{u \in W_N} \alpha_{N,u} (\mathbf{1} - 2u)^T]^T$.

we can eventually show that:

$$-G(\alpha, y, y^*) = y^T \boldsymbol{H} y + y^T (\tau + \Delta) \tag{4.21}$$

For the solution of (4.12), we follow Li et al. (2013): at the generic iteration, a working set C_w of labelings satisfying the constraints in (4.4) has already been determined. We now search if another labeling $\tilde{y}$ exists such that function (4.21) takes a greater value than for every labeling in the working set. If such a labeling exists, it violates (4.4) and a re-computation of the optimum in μ and α in (4.12) is required.

First, we determine the argmax of the current working set, C_w:

$$\bar{y} = \underset{y \in C_w}{\operatorname{argmax}}\, y^T \boldsymbol{H} y + y^T (\tau + \Delta) \tag{4.22}$$

If C_w contains L ground-truth labelings, this step requires L evaluations of (4.21). Then, we search if a labeling $\tilde{y}$ exists such that:

$$\tilde{y}^T \boldsymbol{H} \tilde{y} + \tilde{y}^T (\tau + \Delta) > \bar{y}^T \boldsymbol{H} \bar{y} + \bar{y}^T (\tau + \Delta), \tag{4.23}$$

Given the positive semidefiniteness of $\boldsymbol{H}$, the above search can be performed as a linear program. To this aim, we pose $r = \boldsymbol{H}\bar{y} + \tau + \Delta$ and determine:

$$\tilde{y} = \underset{y \in \beta}{\operatorname{argmax}}\, y^T r \tag{4.24}$$

followed by a post-hoc verification of (4.23). If (4.23) holds, labeling $\tilde{y}$ is added to objective (4.12) and a re-computation is triggered. Otherwise, training concludes.

It should be noted that we are able to determine $\tilde{y}$ so long as we can find the maximum of (4.24) while maintaining the consistency of the unary and pairwise variables and satisfying the balance constraint. At its turn, it is not obvious what a balance constraint should be in the structured case, given that the number of classes, $|\mathcal{Y}|$, is exponential even if the nodes consist of only binary variables. Our choice is simply to balance the number of positive and negative nodes in each sample.

Matrix $\boldsymbol{H}$ is of large size $(M \times M)$ and product $\boldsymbol{H}y$ appears challenging both in terms of storage and the number of flops. However, this is only apparent since an optimized implementation drastically reduces the computational complexity. By calling x_i^t and y_i^t the measurement and label of sample i and node t, the generic element e_i^t of product $\boldsymbol{H}y$ can be computed as:

$$e_i^t = x_i^{t\top} \left(\sum_{j=1:N, u=1:T_j} x_j^u y_j^u \right) \tag{4.25}$$

where the term in parentheses is common to all elements. Similar common terms appear in the pairwise elements and abate the computational complexity from quadratic to linear in M. Please refer to the following Subsection (4.2.3) for more details.

To avoid adding labelings caused by numerical inaccuracies or over-fitting, we impose a minimum threshold, γ, to validate a violating labeling. The violation criterion is therefore given as:

$$\tilde{y}^\top \boldsymbol{H}\tilde{y} + \tilde{y}^\top(\tau + \Delta) - \bar{y}^\top r > \gamma \tag{4.26}$$

4.2.3 Optimized Matrix-Vector Multiplication ($\boldsymbol{H}$y)

Any attempt to store matrix $\boldsymbol{H} \in R^{M\times M}$, even with a small number of samples, will lead to the scarcity of storage capacity. On the other hand, naive multiplication of $\mathcal{H}y$ makes learning an expensive operation. Therefore, we leverage sparsity and the known structure of $\boldsymbol{H}$ to compute $\boldsymbol{H}y$. Instead of formulating the entire $\boldsymbol{H}$ matrix, product $\boldsymbol{H}y$ can easily be calculated by exploiting the known structure of matrix $\boldsymbol{H}$.

To ease the task of imputing product $\boldsymbol{H}y$, we have provided a list of variables used in this section as follows: N = total number of sequences; T_i = number of nodes constituting the i^{th} sequence; x_i^t = D-dimensional measurement vector for the t^{th} node of the i^{th} sequence; $y \in R^M$ is the label vector of the whole dataset (please refer to Section (4.2.1) for the encoding of the label vector); y_i is the i^{th} element of y; $O \in R^N$ where O_i = is the offset parameter for sequence i. Please note that $M = \sum_{i=1}^{N} 2T_i + 4(T_i - 1)$.

By setting $O_1 = 0$, the other entries constituting O are calculated as follows:

$$O_i = 2T_i + 4(T_i - 1) + O_{i-1} \quad i = 2, \ldots, N \tag{4.27}$$

Let $B = \boldsymbol{H}y$. For the i^{th} sequence, B contains $2T_i + 4(T_i - 1)$ corresponding entries (since binary sequences are modeled by a graph having unary and pairwise interaction between variables: the first $2T_i$ entries correspond to unary interaction and the last $4(T_i - 1)$ entries correspond to pairwise interactions). Given any sequence, we

can easily recover its corresponding entries from vector B using our offset vector O. Algorithm (2) summarizes the efficient imputation of $B = \boldsymbol{H}y$.

Algorithm 2 Efficient imputation of unary and pairwise terms constituting B

Input: $x = \{x_1, \ldots, x_N\}$, $O \in R^N$ and $y \in R^M$
Initialize: $B \in R^M$
for i=1:N **do**
  for t=1:T_i **do**
    $t_1 = O_i + 2t - 1$
    $t_2 = O_i + 2t$
    $B_{t_1} = x_i^t \left(\sum_{j=1}^{N} \sum_{u=1}^{T_j} x_j^u y_{2u+O_j-1} \right)$ **(Unary term)**
    $B_{t_2} = x_i^t \left(\sum_{j=1}^{N} \sum_{u=1}^{T_j} x_j^u y_{2u+O_j} \right)$ **(Unary term)**
  end for
  for trans=1:$\mathrm{T}_i - 1$ **do**
    ind $= O_i + 2T_i + 4(\mathrm{trans} - 1) + 1$
    $t_a =$ind
    $t_b =$ind + 1
    $t_c =$ind + 2
    $t_d =$ind + 3
    $B_{t_a} = \sum_{j=1}^{N} \sum_{u=1}^{T_j - 1} y_{(O_j+2T_j+1)+4(u-1)}$ **(Pairwise term)**
    $B_{t_b} = \sum_{j=1}^{N} \sum_{u=1}^{T_j - 1} y_{(O_j+2T_j+2)+4(u-1)}$ **(Pairwise term)**
    $B_{t_c} = \sum_{j=1}^{N} \sum_{u=1}^{T_j - 1} y_{(O_j+2T_j+3)+4(u-1)}$ **(Pairwise term)**
    $B_{t_d} = \sum_{j=1}^{N} \sum_{u=1}^{T_j - 1} y_{(O_j+2T_j+4)+4(u-1)}$ **(Pairwise term)**
  end for
end for
return B

As a consequence of Algorithm (2), we obtain a major reduction in the flop counts of $\boldsymbol{H}y$. With a naive multiplication of $\boldsymbol{H}y$, the flop counts for B_t (t^{th} entry of vector B) would be $2M-1$. By leveraging sparsity, it is significantly reduced to $2M_Z-1$ where

M_Z are the number of non-zeros inside $\boldsymbol{H}$ whose locations are known beforehand, thanks to the design of the feature map. The multiplication of $\boldsymbol{H}y$ is accelerated even further by caching unary and pairwise terms for each sequence (Line 7−Line 8 and Line 16−Line 19 of Algorithm (2)). Computation of blocks of $\boldsymbol{H}$ i.e. $h_i h_j^T$ is avoided as well, which is another advantage of Algorithm (2).

4.2.4 Balanced Sequential Labeling

As a case study, we have implemented a task of sequential labeling with a hidden Markov model (HMM). In an HMM, the output variables form a Markov chain of T nodes with a number of edges, E, equal to $T-1$. Given a sequence of measurements in input, inference of the sequence of states is provided by the classic Viterbi algorithm. Equation (4.24) requires the evaluation of Viterbi for every training sample using the coefficients specified by vector r as emission and transition weights. This is equivalent to inference in an HMM with frame-varying parameters.

To extend the conventional Viterbi algorithm to incorporate a balance constraint over the nodes, it is sufficient to augment the "state" at frame t with the count of nodes assigned with positive labels up to that frame. A sub-sequence at frame t can contain P positive labels only if:

1. the sequence at frame $t-1$ contained $P-1$ positive labels and y_t = positive;
2. the sequence at frame $t-1$ contained P positive labels and y_t = negative.

The above state configuration allows Viterbi to return every best sequence with a given number of positive labels. The number of such best sequences is $T+1$ (from the case of no positive labels, to all positive), for an $O(T^2)$ computational complexity. A final step evaluates the best of such sequences that also meets the balance constraint. This simple algorithm could be further optimized by dropping the evaluation of sequences as soon as they breach the desired balance.

We have summarized the implementation of balanced viterbi in Algorithm (3): $\beta_{\min}$ and $\beta_{\max}$ are the parameters that maintains the desired balance between the number of positive and negative labels; P is the generic number of positive labels; P_t contains the number of positive labels till the t^{th} frame of a given sequence; [] is the string concatenation operator i.e. it will form a string of labels as we move along the sequence; $\Psi(P_t, y_t)$ represents a sequence of t frames with P_t positive labels ending at y_t (Please note that $y_t \in \{0, 1\}$, $P_t \in \{0, t\}$); w_0 and w_1 are the respective models for negative and positive emissions; score(sequence) will return the score of the argument sequence using famous equations of Viterbi algorithm and an invalid sequence can easily be found by the following four rules:

- A given sequence will be invalid if $P < 0$
- A given sequence will be invalid if $P = 0$ and $y^t = 1$
- A given sequence will be invalid if $P = t$ and $y^t = -1$
- A given sequence will be invalid if $P > t$

With these notations and rules for finding valid sequences, we can easily implement the balanced version of Viterbi using the following relations:

- $\text{score}(\Psi(P, y_t = 1)) = \text{score}(\Psi(P - 1, y_{t-1})) + w_1^\top x_t$
- $\text{score}(\Psi(P, y_t = 0)) = \text{score}(\Psi(P, y_{t-1})) + w_0^\top x_t$
- $\Psi(\text{invalid argument}) = \text{NULL}$ and $\text{score}(\text{NULL}) = -\infty$

Algorithm (3) presents the step by step implementation details for the loss augmented inference using the Balanced Viterbi algorithm.

4.3 Experimental Results

In this section, we report experiments comparing the proposed Well-SSVM with the closest structural model, latent structural SVM (Latent SSVM) (Yu and Joachims

Algorithm 3 Balanced Loss Augmented Inference

Input: model w derived from Hy as usual, measurement $x = \left(x^1, \ldots, x^T\right)$
Output: predicted class $\bar{y}$, $\bar{y} = \left(\bar{y}^1, \ldots, \bar{y}^T\right)$
Initialize: $\Psi(P_1 = 0, y^1 = 0) = 0$ and $\Psi(P_1 = 1, y^1 = 1) = 1$
Main loop:
for t=2:T **do**
    for P=0:t **do**
        $\Psi(P, y^t = 0)$ = argmax (score([$\Psi(P, y^{t-1} = 0)$,0]) , score([$\Psi(P, y^{t-1} = 1)$,0]))
        $\Psi(P, y^t = 1)$ = argmax (score([$\Psi(P-1, y^{t-1} = 0)$,1]) , score([$\Psi(P-1, y^{t-1} = 1)$,1]))
    end for
end for
Final loop:
best $= -\inf$, $\bar{y}$ =NULL
for $\beta_{\min}$:$\beta_{\max}$ **do**
    temp = max(score($\Psi(P, y^T = 0)$), score($\Psi(P, y^T = 1)$))
    if temp>best **then**
        $\bar{y}$ = argmax(score($\Psi(P, y^T = 0)$), score($\Psi(P, y^T = 1)$))
        best = temp
    end if
end for
return $\bar{y}$

(2009)), and using k-means as the baseline. The comparison reports the F_1 score for predicting the withheld ground-truth labels of the training set.

4.3.1 Dataset Description

We have conducted experiments using two datasets: a synthetic dataset for controlled experiments and the Gesture Phase Segmentation dataset from Kinect data(Madeo et al. (2013)). We briefly describe both datasets below.

4.3.1.1 Synthetic Dataset

We have first created a small 1D synthetic dataset with 10 sequences, each 12 frames in length. For the generative model, we have used an HMM with two states and

TABLE 4.1: Comparison of clustering accuracy over the synthetic and Gesture Phase Segmentation datasets. Accuracy is reported as F_1 score (± standard deviation) over 10 runs of each technique.

DATASET	k-MEANS	LATENT SSVM	WELL-SSVM
SYNTHETIC (1D)	0.74±0.00	0.76±0.00	**0.83±0.00**
SYNTHETIC (2D)	0.87±0.00	**0.94±0.00**	**0.94±0.00**
GESTURE-A1 (32D)	0.60±0.00	0.59±0.01	**0.66±0.01**
GESTURE-A2 (32D)	0.58±0.00	0.59±0.01	**0.66±0.01**
GESTURE-A3 (32D)	0.72±0.03	0.72±0.04	**0.76±0.01**
GESTURE-B1 (32D)	**0.88±0.01**	**0.88±0.01**	0.79±0.03
GESTURE-B3 (32D)	**0.83±0.01**	**0.83±0.01**	0.82±0.02
GESTURE-C1 (32D)	0.74±0.02	0.72±0.06	**0.79±0.07**
GESTURE-C3 (32D)	0.78±0.03	0.76±0.05	**0.81±0.02**
Gesture average	0.73±0.01	0.73±0.03	**0.76±0.02**

a probability to remain in the same state of 0.95. As emission densities, we have used two Gaussian distributions with parameters $\mu_1 = -1$, $\mu_2 = 1$ and $\sigma = 2$. We have also created a 2D dataset consisting of only one long sequence of 1200 frames using the same generative model. The measurements have been sampled from two Gaussian distributions with the following parameters: $\mu_1 = [-1, -1]$, $\mu_2 = [1, 1]$ and $\Sigma = [1, 0.2; 0.2, 1]$.

4.3.1.2 Gesture Phase Segmentation Dataset

The Gesture Phase Segmentation dataset consists of Kinect joint information for a narrating actor. Temporal annotation is provided in terms of gesture phases such as rest, preparation, stroke, hold and retraction. Given that we only assign binary labels, for our experiment we have equated 'rest' to the negative class and all the other phases to the positive. We have also scaled the measurements up by a factor of 100 to work in an approximately unitary range. In this dataset, a name such as 'Gesture-A1' refers to the video of actor 'A' performing task '1'. The length of the videos varies from a minimum of $1,073$ to a maximum of $1,747$ frames. The dataset can be downloaded from the UCI Machine Learning repository (Lichman (2013)).

4.3.2 Initialization

Initialization plays an important role in the performance of all tested algorithms. k-means is used both as the baseline approach and to initialize the two structural algorithms. In turn, the initial centroids of k-means are assigned randomly from the data.

4.3.3 Performance Comparison

Table 4.1 reports the accuracy in terms of average F_1 score over 10 runs. On the synthetic dataset, both Latent SSVM and Well-SSVM report notable improvement over k-means. However, Latent SSVM achieves on average the same accuracy as k-means on the Gesture Phase Segmentation dataset. Conversely, Well-SSVM achieves an average improvement of 3 percentage points on this dataset and the highest accuracy for 5 videos out of 7. This result is remarkable since the models of Latent SSVM and Well-SSVM are directly comparable and share the same initialization procedure. k-means only achieves the highest accuracy on two videos of the Gesture Phase Segmentation dataset: by plotting their measurements on a principal component space, we can observe that they are more neatly separated in feature space. However, when sequentiality is more pronounced, Well-SSVM always reports the highest accuracy.

It is useful to compare the computational complexity of Latent SSVM and Well-SSVM. Both rely on structural SVM, whose computational complexity was analysed in Tsochantaridis et al. (2005a); Joachims et al. (2009b). In its most efficient implementation (i.e., 1-slack), the complexity is substantially independent of the number of samples; we refer to it simply as $O(\mathrm{SSVM})$. Latent SSVM iterates structural SVM until the latent variables stabilize: noting the number of iterations as I_{L}, its complexity is then $O(I_{\mathrm{L}}\,\mathrm{SSVM})$. In our experiments, the value of I_{L} proved typically to be between 2 and 3. Conversely, Well-SSVM iterates structural SVM over the number of ground-truth labelings, L, which is typically in the order of 2 to 4. In addition, at every iteration it solves a multi-kernel problem that is iterative at its

turn. Noting the number of MKL iterations as I_{MKL}, the overall complexity is therefore $O(L\, I_{\text{MKL}}\, \text{SSVM})$. De facto, the training time of Well-SSVM is approximately $5 - 6\times$ that of Latent SSVM in an average case.

4.4 Conclusion

We have presented a novel approach for performing unsupervised maximum-margin learning of structured data. The proposed approach, called weakly labeled structural SVM or Well-SSVM for short, extends the minimax relaxation of Li et al. (2013) to the structured case. The main contribution of our work is the re-organization of the structured feature vectors and the Hamming loss in a form that allows for finding violating labelings for the relaxation. Experimental results on sequential labeling with synthetic and real datasets show that the proposed approach outperforms the popular latent structural SVM and a k-means baseline in the majority of cases. While we have only addressed structures of binary labels, an extension to multi-class labels is also possible through binary coding of the class variables.

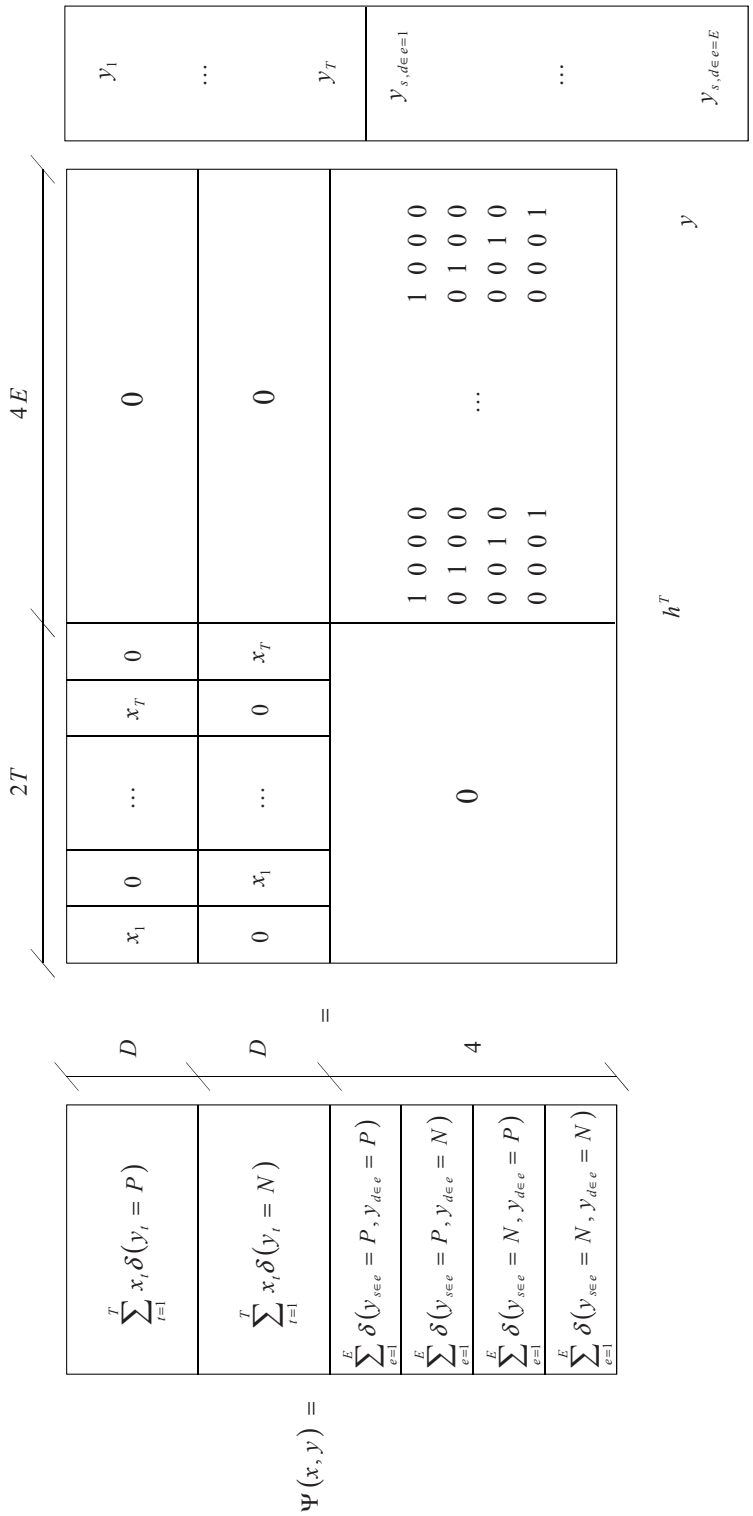


FIGURE 4.1: Feature map h. The two binary variables for node t are noted jointly as y_t; the four binary variables for edge e are noted jointly with a double index as $y_{s,d=e}$.

Chapter 5

Unsupervised Structured Prediction Maximum Margin Markov Networks

5.1 Introduction

The Maximum Margin Markov Networks (M^3N or M^3 Networks hereafter) (Taskar et al. (2004)) provide an alternative way to train structured predictors (Section (2.5)) in a discriminative fashion. For the sake of simplicity, we will consider the case of a linear sequence (comprising of T nodes) throughout this chapter. In Section (2.5), we have already seen how Structural SVM (Tsochantaridis et al. (2005a)) handles the training of structured predictors. The main difference between Structural SVM (Tsochantaridis et al. (2005a)) and M^3 networks lies in the formulation of the quadratic program (QP) that is eventually solved to obtain the desired parameters for the classifier. Since the number of constraints is exponential in the length of sequence, Structural SVM (Tsochantaridis et al. (2005a)) attempts to solve the resulting QP using the cutting-plane method. In this way, the constraint size becomes manageable for solution via decomposition method or SMO. Conversely, the M^3 networks propose to solve the resulting QP with the full set of constraints. The

main trick is to obtain a factored dual, that reformulates the original QP into a factored QP whose number of parameters is linear in T.

In this chapter, we propose an alternative solution of Well-SSVM dual via the M^3 networks. How to extend M^3 networks to its multiple-kernel learning (MKL) equivalent (Section (5.5.1)) and how to perform a step of violating labeling (Section (5.5.2)) is the focus of the entire chapter. With the suggested modifications, we can avail of the current implementation of Well-SSVM to solve the factored QP with the full set of constraints (Taskar et al. (2004)).

It is worth mentioning that the Structural SVM is applicable to a broader class of QP, where feature functions and losses over graphs (for instance F1-loss) are non-decomposable as well. Conversely, decomposability of feature function and loss (for instance Hamming loss) over graph is the fundamental requirement of M^3 networks.

In the upcoming sections, we will present the solution of Well-SSVM via M^3 networks formulation. Section (5.2) contains the summary of notations that is consistent with Chapter 4. Section (5.3) takes the dual formulation of N-slack Multiclass SVM and shows its conversion into the proposed M^3 networks dual formulation (Taskar et al. (2004)). Section (5.4) presents the decomposition of our previously proposed feature function of Chapter 4; its decomposition is on par with the decomposed feature function of M^3N. Since we are dealing with the multiple ground-truth labelings, Section (5.5) presents the extension of M^3 networks QP to the case of the multiple ground-truth labelings. In Section (5.6), experimental results are presented where the proposed algorithm is compared against k-Means, Unsupervised-SSVM (Yu and Joachims (2009)) and the EM algorithm (Dempster et al. (1977)). Finally, Section (5.7) will conclude the discussion with the possible future work.

5.2 Notations

We will maintain the consistency of notations used throughout our thesis. For the sake of simplicity, we have considered a case of linear sequence with T nodes. We have made explicit notations to address nodes and pairs of a sequence as follows:

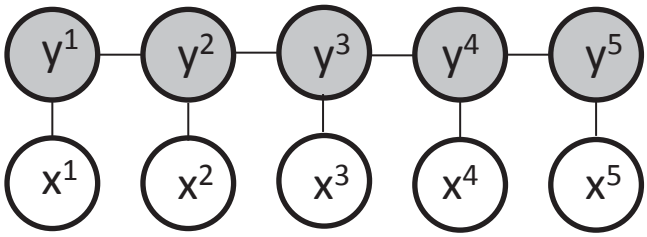


FIGURE 5.1: An example of a sequence with 5 output (shaded) nodes.

Accessing Output Nodes

To work at node levels, consider the case of sequence with (T =5) nodes shown in Figure 5.2. Given any sequence, y^a refers to its a^{th} output node. For instance, the value of the 5^{th} node is symbolized by y^5. In our discussion, y_i^a will be used to denote the ground-truth value of the a^{th} output node from i^{th} sequence. Similarly, u^a will represent its predicted value. Furthermore, (throughout this chapter) the cardinality of each output node is set to 2 i.e. $y^a = \{0, 1\}$. However, our work can easily be extended to the case where cardinality of each output node is greater than 2 (multiclass case).

Accessing Pair of Output Nodes

Given any labeling y, we will refer to an edge between node 'a' and 'b' using y^{ab}. For instance, in Figure 5.2; y^{45} represents an edge between y^4 and y^5. For binary nodes, each edge can have four unique assignments i.e. $y^{ab} = \{(0,0),(0,1),(1,0),(1,1)\}$. Similar to the case of nodes, y_i^{ab} and u^{ab} will represent the ground-truth and predicted assignment of node pair (a, b).

5.3 Factorized Dual

We will start from the N-slack formulation of Multiclass SVM (margin rescaling case). The dual of N-slack formulation is given as follows (an extra dual variable α_{iy_i} is added for each example as we always include $u = y_i$ for the case of margin

rescaling):

$$\begin{aligned} &\underset{\alpha}{\operatorname{argmax}} \sum_{i,u} \alpha_{iu} \Delta(y_i, u) - \frac{1}{2} || \sum_{i,u} \alpha_{iu} \delta\psi_i(u) ||^2 \\ &s.t. \ \sum_{u} \alpha_{iu} = C, \forall i; \quad \alpha_{iu} \geq 0, i = 1, \ldots, N, \forall u. \end{aligned} \tag{5.1}$$

where α_{iu} are the Lagrange multipliers for the constraints of sample i; $\Delta(y_i, u)$ is the loss over prediction u; and $\delta\psi_i(u) = \psi(x_i, y_i) - \psi(x_i, u)$ is the difference of feature function. Please see Section (2.5) for the details of these parameters. Taskar et al. (2004) observed that since the sum of all Lagrangian coefficients associated with sample constraints is constant i.e. $\sum_u \alpha_{iu} = C$ and all of them are positive i.e. $\alpha_{iu} \geq 0$ therefore, the dual QP can actually be seen as a density function over u given sample i. Hence, the dual QP is the function of expectation of loss term $(\Delta(y_i, u))$ and the difference of feature functions $(\delta\psi_i(u))$. The two vital conditions for the proposed factorization of dual are: (i) the loss is decomposable over a labeling i.e $\Delta(y_i, u) = \sum_{t=1}^{T} I(y_i^t \neq u^t) = \sum_{t=1}^{T} \Delta t_i(u^t)$ and (ii) the feature function for a labeling y is decomposable over its edges i.e. $\delta\psi_i(y) = \sum_{(a,b)} \delta\psi_i(y^a, y^b)$. With such conditions, Taskar et al. (2004) worked out the marginal dual variables over nodes and edges as follows:

$$\begin{aligned} \lambda_i(u^a) &= \sum_{u \sim [u^a]} \alpha_{iu} \qquad \forall a, \forall u^a, \ \forall i; \\ \lambda_i(u^a, u^b) &= \sum_{u \sim [u^a, u^b]} \alpha_{iu} \qquad \forall (a,b) \in E, \forall u^a, u^b \ \forall i; \end{aligned} \tag{5.2}$$

where $u \sim [u^a]$ is the labeling with the fixed assignment of its a^{th} node i.e. $[\ldots, u^a, \ldots]$. Similarly, $u \sim [u^a, u^b]$ is the labeling with the fixed assignment of its node pair (a,b) i.e. $[\ldots, u^a, u^b, \ldots]$. From Equation (5.2), it follows that $\lambda_i(u^a) = \sum_{u^b} \lambda_i(u^a, u^b)$. For a sequence of $(T = 5)$ binary nodes, we will have $(2 \times 5 = 10)$ unary marginal dual variables (2 for each node) and $(4 \times 4 = 16)$ pairwise marginal dual variables (4 for each edge). Using Equation (5.2), we can easily impute the linear and quadratic

terms of the proposed dual QP (Taskar et al. (2004)). Consider the linear term of QP in Objective (5.1):

$$\begin{aligned}
&\sum_{u} \alpha_{iu} \Delta(y_i, u) \ \forall i \\
&= \sum_{u} \alpha_{iu} \left(\sum_{t=1}^{T} \Delta t_i(u^t) \right) \\
&= \sum_{t,u^t} \Delta t_i(u^t) \sum_{u \sim [u^t]} \alpha_{iu} \\
&= \sum_{t,u^t} \Delta t_i(u^t) \lambda_i(u^t)
\end{aligned} \tag{5.3}$$

It is the linear term of M^3N dual formulation (Taskar et al. (2004)). Similarly, consider the quadratic term of QP in Objective (5.1):

$$\begin{aligned}
&\sum_{u} \alpha_{iu} \delta\psi_i(u) \ \forall i \\
&= \sum_{u} \alpha_{iu} \sum_{(a,b)} \delta\psi_i(u^a, u^b) \\
&= \sum_{(a,b)} \sum_{u} \alpha_{iu} \delta\psi_i(u^a, u^b) \\
&= \sum_{(a,b)} \sum_{u^a, u^b} \sum_{u \sim [u^a, u^b]} \alpha_{iu} \delta\psi_i(u^a, u^b) \\
&= \sum_{(a,b)} \sum_{u^a, u^b} \delta\psi_i(u^a, u^b) \sum_{u \sim [u^a, u^b]} \alpha_{iu} \\
&= \sum_{(a,b)} \sum_{u^a, u^b} \delta\psi_i(u^a, u^b) \lambda_i(u^a, u^b)
\end{aligned} \tag{5.4}$$

It is the quadratic term of the M^3N dual formulation of Taskar et al. (2004). Plugging results from Equation (5.3) and Equation (5.4) into the N-slack formulation of multiclass, we can write the factored QP (Taskar et al. (2004)) as follows:

$$\max G(\lambda) =$$
$$\max \sum_{i} \sum_{a,u^a} \lambda_i(u^a) \Delta t_i(u^a) - \frac{1}{2} \sum_{i,j} \sum_{\substack{(a,b) \\ u^a,u^b}} \sum_{\substack{(c,d) \\ v^c,v^d}} \lambda_i(u^a,u^b) \lambda_j(v^c,v^d) \delta\psi_i(u^a,u^b)^\top \delta\psi_j(v^c,v^d)$$
$$s.t. \ \sum_{u^a} \lambda_i(u^a,u^b) = \lambda_i(u^b); \ \sum_{u^a} \lambda_i(u^a) = C; \ \lambda_i(u^a,u^b) \geq 0. \tag{5.5}$$

To show how such factorization has reduced the number of parameters in the QP of Objective (5.1), we will again go back to the case of only one sequence with 5 binary nodes. We can write the linear term of the original and factored dual as follows (please note that $N = 1$ for the following linear and quadratic terms):

1. $\sum_u \alpha_{iu} \Delta(y_i, u)$ (Original dual): It will have $(2^5 =)32$ addenda.
2. $\sum_{a,u^a} \lambda_i(u^a) \Delta t_i(u^a)$ (Factored dual): It will have $(2 * 5 =)10$ addenda.

Similarly, the quadratic term of the original and factored dual will have the following number of addenda:

1. $||\sum_u \alpha_{iu} \delta\psi_i(u)||^2$ (Original dual): It will have $(2^5 \times 2^5 =)1024$ addenda.
2. $\sum_{\substack{(a,b) \\ u^a,u^b}} \sum_{\substack{(c,d) \\ v^c,v^d}} \lambda_i(u^a,u^b) \lambda_j(v^c,v^d) \delta\psi_i(u^a,u^b))^\top \delta\psi_j(v^c,v^d)$ (Factored dual): It will have $(4 \times 4 \times 4 \times 4 =)256$ addenda.

It can be seen clearly that solving a dual QP with full set of constraints using M^3 networks formulation (Objective 5.5) entails a polynomial number of terms as opposed to the unfactored dual which entails an exponential number (Objective 5.1).

5.4 Decomposition of Feature Function over Edges

The proposed M^{3}N formulation (Taskar et al. (2004)) depends on the factorization of feature function $\psi(x, y)$. Feature functions can be decomposed as a sum of feature functions over edges:

$$\psi(x, y) = \sum_{(a,b)\in E} \psi(x, y^a, y^b) \tag{5.6}$$

In Chapter 4, we proposed to represent $\psi(x, y)$ as:

$$\psi(x, y) = h^\top y$$

where, $x^a \in R^D$ and $\psi(x, y) \in R^{2D+4}$. For a linear sequence x with T nodes, h was a matrix of dimension $(2T + 4E) \times (2D + 4)$ where $E = T - 1$. $h^\top y$ actually models the emission and transition of all nodes. Now, we will decompose h according to nodes and edges. From now on, h^{ab} would model the transition between nodes 'a' and 'b' whereas for emission, it will consider node 'b' only. Therefore, h^{ab} will be a matrix of dimension $(2 + 4) \times (2D + 4)$.

Now we are set to decompose h. Our proposed representation of feature function (in Chapter 4) will be factorized in the similar way of Equation (5.6) as follows:

$$\psi(x, y) = \sum_{(a,b)\in E} {h^{ab}}^\top y^{ab} \tag{5.7}$$

The difference of feature function for prediction u corresponding to sequence i will be denoted by $\delta\psi_i(u)$, which can be evaluated as follows:

$$\begin{aligned} \delta\psi_i(u) &= \psi_i(x_i, y_i) - \psi_i(x_i, u) \\ &= \sum_{(a,b)\in E} \left[\psi_i(y_i^a, y_i^b) - \psi_i(u^a, u^b)\right] \\ &= \sum_{(a,b)\in E} \left[{h_i^{ab}}^\top y_i^{ab} - {h_i^{ab}}^\top u^{ab}\right] \end{aligned} \tag{5.8}$$

5.5 Solution of WellSSVM via M^{3}N

In the previous sections, we presented M^{3}N notations, factorized dual and the decomposition of feature function as a sum over the edges of sequence $\psi(x, y) = \sum_{(a,b)\in E} h^{ab\top} y^{ab}$. In this section, we will present the complete algorithm for unsupervised structured prediction (Chapter 4) with the proposed formulation of factorized dual (Taskar et al. (2004)).

Algorithm (4) presents the brief overview to perform unsupervised structured prediction via M^{3}N formulation. To avoid clashing with other indices, we note the labeling index l as ${}^l y$.

Algorithm 4 Well-SSVM via M^{3}N

1: **Initialization:** ${}^1 y; \mu^1 = 1.0$; $\gamma = 0.1$ (Equation (4.26)) and $L = 1$
2: Solve Objective (5.5) and store λ.
3: Find violating labeling (Section) (5.5.2)
4: **repeat**
5: $L = L + 1$
6: Initialize $\mu^{1:L} = \left[\mu^1, \ldots, \mu^L\right]^T$
7: Concatenate ground-truth labelings from Step 3/Step 9: ${}^{1:L} y = \left[{}^1 y, \ldots, {}^L y\right]$
8: **repeat**
9: Solve Objective (5.10) and store λ
10: update $\mu^{1:L}$ (Section (5.5.3))
11: **until** $\mu^{1:L}$ converges
12: Find violating labeling (Section (5.5.2))
13: **until** Equation (4.26)
14: Impute w (Section (5.5.3))

5.5.1 Learning as an Instance of MKL

We have seen in Section (4.2) that the objective of Well-SSVM with multiple ground-truth ($l = 1 : L$) labelings is given by:

$$\min_{\mu} \max_{\lambda} \sum_{l: {}^l y \in \beta} \mu^l G(\lambda, {}^l y) \tag{5.9}$$

Every sample has multiple ground-truth labelings (denoted by ${}^l y$) and the score is obtained as the sum over such labelings. Such labelings are constrained to belong to set β, which is a set of balanced labelings. We will introduce another notation to address labeling with multiple groudtruths: ${}^l y_i^a$ is the l^{th} ground-truth value of a^{th} from i^{th} sequence.

The maximum in λ (5.9) can be found by the factorized dual (Taskar et al. (2004)) after making the following position:

- $\tilde{\psi}_i(y^a, y^b) = [\sqrt{\mu^1}\psi_i(y^a, y^b), \ldots, \sqrt{\mu^L}\psi_i(y^a, y^b)]^T$: L weighted concatenation of $\psi_i(y^a, y^b)$;
- $\delta\tilde{\psi}_i(y^a, y^b) = \left[\sqrt{\mu^1}\left(\psi_i({}^1y_i^a, {}^1y_i^b) - \psi_i(y^a, y^b)\right), \ldots, \sqrt{\mu^L}\left(\psi_i({}^Ly_i^a, {}^Ly_i^b) - \psi_i(y^a, y^b)\right)\right]$
- $\tilde{\Delta}t_i(y^a) = \sum_{l=1}^{L} \mu^l \Delta t_i({}^l y_i^a, y^a)$: the loss of node y^a versus a^{th} node of all 'L' ground-truth labelings. Please note that $\Delta t_i({}^l y_i^a, y^a) = I({}^l y_i^a \neq y^a)$.

With the following position, we can pose the inner maximization of Objective (5.9) as follows (it is actually an instance of M^{3}N; more precisely, we have replaced Structural SVM by M^{3}N from here) :

$$\begin{aligned}
&\max_{\lambda} \sum_{l:y^l \in \beta} \mu^l G(\lambda, {}^l y) = \\
&\max \sum_i \sum_{a,u^a} \lambda_i(u^a)\tilde{\Delta}t_i(u^a) - \frac{1}{2}\sum_{i,j}\sum_{\substack{(a,b)\\u^a,u^b}}\sum_{\substack{(c,d)\\v^c,v^d}} \lambda_i(u^a,u^b)\lambda_j(v^c,v^d)\delta\tilde{\psi}_i(u^a,u^b)^\top\delta\tilde{\psi}_j(v^c,v^d) \\
&s.t.\ \sum_{u^a}\lambda_i(u^a,u^b) = \lambda_i(u^b);\ \sum_{u^a}\lambda_i(u^a) = C;\ \lambda_i(u^a,u^b) \geq 0.
\end{aligned} \tag{5.10}$$

where the scalar products $\delta\tilde{\psi}^T\delta\tilde{\psi}$ and loss $\tilde{\Delta}t_i(u^a)$ contain the μ factor.

5.5.2 Finding a Violating Labeling

The main trick for finding a violating labeling is to express $G(\lambda, y)$ in the following form:

$$-G(\lambda, y) = y^T \mathcal{H} y + y^T(\tau + \Delta) \tag{5.11}$$

where y will be a concatenation of L ground-truth labelings. In order to express $G(\lambda, y)$ as a quadratic function of y, we will present the necessary positions to obtain the desired H,τ and Δ. For more details, please refer to Appendix C.

Let us assume the following positions:

- **Implementation of $\mathbf{h_i}$**: As we have discussed earlier, h_i will be decomposed into h_i^{ab}. Inside h_i^{ab}, entries that belong to nodes of a sequence will be pre-multiplied by C whereas for transition, we only need to insert C in the respective positions. After making suggested changes, we can easily formulate $\mathcal{H}$ as follows:

  $$\begin{aligned}\mathcal{H} = \tfrac{1}{2}[&H_{11}, H_{12} \dots H_{1N};\\ &\dots;\\ &H_{N1}, H_{N2} \dots H_{NN}]\end{aligned}$$

  (please note that matrix $\mathcal{H}$ is positive semidefinite by construction. Its row dimension, M, is given by the sum of the labeling size over the entire training set: $M = \sum_i 2T_i + 4E_i$)

- $\tau = \left[-h_1\left(\sum_j \sum_{(c,d)} {h_j^{cd}}^\top \kappa_j^{cd}\right), -h_2^\top\left(\sum_j \sum_{(c,d)} {h_j^{cd}}^\top \kappa_j^{cd}\right), \dots, -h_N^\top\left(\sum_j \sum_{(c,d)} {h_j^{cd}}^\top \kappa_j^{cd}\right)\right]$
  where $\kappa_j^{cd} = \sum_{v^c, v^d} \lambda_j(v^c, v^d) v_j^{cd}$

- $\Delta^\top = \left[\underbrace{\{\lambda_i(u^a = 0), \lambda_i(u^a = 1)\}}_{\text{for } a^{th} \text{ frame}} \quad \underbrace{\mathbf{0} \in R^{4T_i - 1}}_{\text{All zeros for transitions between frames}}\right]$

Once we implement the aforementioned positions, we can eventually transform dual of Well-SSVM (Objective (5.10)) as a quadratic function of labeling y as shown in Equation (5.11). For the detailed description of $\mathcal{H}$, τ and Δ, please refer to Appendix (C.2).

From here, we can easily find the desired violating labeling as discussed in Section (4.2.2) (starting from Equation (4.22)). Please see Section (4.2.4) for balanced sequential labeling or balanced Viterbi implementation.

5.5.3 Update μ

The solution to factored dual (Objective (5.5)) can be used to impute w. Taskar et al. (2004) evaluated w as follows:

$$w = \sum_{i} \sum_{(a,b)} \sum_{y^a, y^b} \lambda_i(y^a, y^b) \delta\psi_i(y^a, y^b) \tag{5.12}$$

For the L ground-truth labelings, we can form the concatenated weight vector as $w^{1:L}$:

$$w^{1:L} = \left[\sum_{i} \sum_{(a,b)} \sum_{y^a, y^b} \lambda_i(y^a, y^b) \delta\psi_i^1(y^a, y^b), \ldots, \sum_{i} \sum_{(a,b)} \sum_{y^a, y^b} \lambda_i(y^a, y^b) \delta\psi_i^L(y^a, y^b) \right]$$

where $\delta\psi_i^l(y^a, y^b) = \sqrt{\mu^l} \left(\psi_i({}^l y_i^a, {}^l y_i^b) - \psi_i(y^a, y^b) \right)$

Now we can update μ in closed form (Li et al. (2013)) as follows:

$$\mu^l = \frac{\|w^l\|}{\sum_{l=1}^{L} \|w^l\|}, \; l = 1 \ldots L \tag{5.13}$$

5.6 Experiments

In this section, we report experiments comparing the proposed Well-M^{3}N with popular unsupervised algorithms such as k-means, Expectation-Maximisation (EM) and

Table 5.1: Comparison of clustering accuracy over the Synthetic and Gesture Phase Segmentation datasets. Accuracy is reported as F_1 score (± standard deviation).

Dataset	k-means	EM	Unsupervised-SSVM	Well-M^{3}N
Synthetic (2D)	0.89±0.00	**0.98±0.00**	0.87±0.00	0.91±0.00
Synthetic (3D)	0.91±0.00	**0.98±0.00**	0.95±0.00	0.74±0.00
Synthetic (5D)	0.94±0.00	**0.99±0.00**	0.93±0.00	0.95±0.01
Gesture-A1 (32D)	0.60±0.00	**0.73±0.00**	0.59±0.01	0.66±0.00
Gesture-A2 (32D)	0.58±0.00	**0.75±0.00**	0.59±0.00	0.66±0.00
Gesture-A3 (32D)	0.72±0.03	0.68±0.00	0.72±0.04	**0.76±0.01**
Gesture-B1 (32D)	0.88±0.01	0.86±0.01	0.88±0.00	**0.94±0.00**
Gesture-B3 (32D)	0.88±0.01	0.75±0.02	0.81±0.00	**0.91±0.00**
Gesture-C1 (32D)	0.74±0.02	**0.81±0.00**	0.63±0.00	**0.81±0.00**
Gesture-C3 (32D)	0.78±0.03	0.70±0.02	0.78±0.00	**0.85±0.00**
Gesture average	0.74±0.00	0.75±0.00	0.71±0.00	**0.80±0.00**

Unsupervised-SSVM (Dempster et al. (1977); Yu and Joachims (2009)). k-means is a baseline algorithm that treats the data as unstructured, EM is a generative approach where latent variables are marginalised during training, and Unsupervised-SSVM is an example of local SVM algorithm. The comparison reports the F_1 score of classifiers at predicting the withheld ground-truth labels of the training set.

5.6.1 Datasets Description

We have conducted experiments over the following datasets: three variations of synthetic dataset, and the Gesture Phase Segmentation dataset (Madeo et al. (2013)). The Gesture Phase Segmentation datasets are available from the UCI Machine Learning repository (Lichman (2013)).

5.6.1.1 Synthetic Dataset

The synthetic dataset is a small dataset containing one-dimensional observations sampled from an HMM with two states, Gaussian emissions and arbitrary parameters. It consists of a single sequence having 1200 frames.

5.6.1.2 Gesture Phase Segmentation Dataset

Please refer to Section 4.3.1.2 for the details of this dataset.

5.6.2 Initialisation

Initialization plays an important role in the performance of all tested algorithms. k-means is used both as the baseline approach and to initialize the three structural algorithms: EM, Unsupervised-SSVM and Well-M^{3}N. In turn, the initial centroids of k-means are assigned randomly from the data.

5.6.3 Performance Comparison

Table 5.1 reports the accuracy in terms of the F_1 score over the initial labelings from k-means, showing best results in bold face. Well-M^{3}N and EM achieve the best results in most cases, with EM reporting a clearly better performance on Synthetic datasets while Well-M^{3}N delivers strong results over Gesture dataset.

EM appears to outperform all approaches in Synthetic datasets. Since the synthetic datasets were generated by HMM with arbitrary parameters, models found by EM are more likely to match the initialised model. The reason for similarities in models lies in the fact that both models are the result of identical generative process. When it comes to the real world scenario (Gesture Segmentation dataset), Well-M^{3}N outperforms k-means and EM by at least 5 percentage points.

Eventually, k-means reports the worst results with a trend to polarise the predictions over only one class (Gesture Segmentation) or divide the data into two clusters of comparable size (Synthetic). This is evidence that structured approaches are beneficial for this type of data and that dismissing sequentiality generally results in poorer predictions.

5.7 Conclusion

We have presented an alternative way to solve Well-SSVM dual using M^3 networks. The main ingredient of our proposed alternative solution lies in the representation of Well-SSVM dual as a quadratic function of labelings that permits the search of violating labeling through our balanced Viterbi. In order to implement Well-SSVM via M^3N, we utilize our previous implementation from Chapter 4 for finding the violating labeling (Section (5.5.2)). The main advantage of Well-M^3N lies in the fact that Structural SVM is solved fully and not via relaxation. Its competitive experimental results suggest that it is more suitable for the integration with the minimax relaxation of Li et al. (2009). Finally, the formulation M^3N helps maintaining the overall convexity of Well-M^3N.

Chapter 6

Conclusion

Structured prediction aims to predict complex objects such as graphs and trees. Classification of such complex objects serves as the main building block for various applications in robotics, computer vision and machine learning. In this thesis, we have proposed extensions for semi-supervised and unsupervised structured prediction by utilizing the notion of maximum-margin classifiers.

In the first part, we have addressed the problem of recognizing actions from single images via semi-supervised structured prediction. To this aim, we have constructed a graphical model that takes all superpixels of the target image as an input and outputs the corresponding action. In order to capture the relation between action and all superpixels, the correlation between every pair of superpixels is also taken into account while classifying actions. This particular feature of our proposed model slows down the step of inference significantly. To speed up the inference, we have proposed an efficient greedy algorithm that reduces the time complexity significantly allowing practical computability of the proposed graphical model.

For the unsupervised extension of structured prediction, we considered the recently proposed convex relaxation for unsupervised SVM (WellSVM) and extended it to the case of structured datasets, hence naming the technique as Well-SSVM. Our contribution has been the design of a joint feature map that permits finding violated labelings using "balanced" Viterbi algorithm. The proposed technique produced competitive experimental results over synthetic and real world datasets.

Lastly, we have presented the solution of Well-SSVM via M^3 Networks. In Well-SSVM, the quadratic program (QP) for Structural SVM was solved by a cutting-plane method. In this section, we have proposed to replace Structural SVM by M^3 Networks. In this way, we solve the QP with the full set of constraints. This seems more suitable for the proposed minimax relaxation that should ensure overall convexity. The important thing to note with this approach is the underlying limitation on the choice of loss functions. With M^3 Networks, we can only work with loss functions that are decomposable over graphs or trees; for instance, the Hamming loss. This was not the case with Structural SVM that is entirely compatible with more general, non-decomposable loss functions such as the popular F1-score. Therefore, one technique (Well-SSVM via M^3 Networks) is promising of greater accuracy versus the greater flexibility of the other (Well-SSVM). The choice of either technique rests on the requirement of specific application.

Appendix A

Lagrange Duality

We will present a very brief introduction to Langrange duality. For detailed discussion, please refer to Boyd and Vandenberghe (2004).

Lets consider the following minimization problem with inequality constraints only:

$$\begin{aligned} &\text{minimize} \quad f_0(x) \\ &\text{subject to} \quad f_i(x) \leq 0, \qquad i = 1, \ldots, m \end{aligned} \tag{A.1}$$

with $x \in R^n$. Here $f_0(x)$ is the objective function that needs to be minimized under "m" inequality constraint denoted by $f_i(x)$. Let p^* be its solution, which we will call the solution of primal problem. The lagrangian of A.1 will tranform this problem from constrained to unconstrained optimization. Thus Lagrangian $L : R^n \times R^m \longrightarrow R$ is defined as follows:

$$L(x, \alpha) = f_0(x) + \sum_{i=1}^{m} \alpha_i f_i(x) \tag{A.2}$$

While formulating lagrangian, each constraint in A.1 has an associated paramater α also called "dual variable" or "lagrange multiplier". We will define $\theta_p(x)$ as following:

$$\begin{aligned}\theta_p(x) &= \max_{\alpha:\alpha_i \geq 0} L(x, \alpha) \\ &= \max_{\alpha:\alpha_i \geq 0} f_0(x) + \sum_{i=1}^{m} \alpha_i f_i(x)\end{aligned} \tag{A.3}$$

Thus, we can obtain solution to (A.1) by minimizing (A.3):

$$\min_x \theta_p(x) = \min_x \max_{\alpha:\alpha_i \geq 0} L(x, \alpha) \tag{A.4}$$

Now, let us define $\theta_D(\alpha)$ as follows:

$$\theta_D(\alpha) = \min_x L(x, \alpha) \tag{A.5}$$

The subscript "D" stands for dual. Our dual optimization problem can be posed as follows:

$$d^* = \max_{\alpha:\alpha_i \geq 0} \theta_D(\alpha) = \max_{\alpha:\alpha_i \geq 0} \min_x L(x, \alpha) \tag{A.6}$$

If we compare (A.4) and (A.6), the only difference observed lies in the order of min and max. Using minimax theorem, it is known that:

$$d^* \leq p^*$$

Under certain conditions(constraint qualification / Slaters condition), this inequality holds strictly ("strong duality"). "$p^* - d^*$" is called duality gap, which becomes zero in the case of strong duality. In that case, we can solve dual instead of primal to get the solution of our problem define in (A.1).

Lets assume that $f_0(x)$ is convex and $f_i(x)$ are affine. This implies there must exist x^* and α^* that satisfy $d^* = p^*$. Please note that x^* is the solution to primal and α^* is the solution to dual formulation. Under these assumptions, KKT conditions must

hold which are as follows:

$$\frac{\partial}{\partial x_i} L(x^*, \alpha^*) = 0, i = 1, \ldots, n \tag{A.7}$$

$$\alpha_i^* f_i(x^*) = 0, i = 1, \ldots, m \tag{A.8}$$

$$\alpha_i^* \geq 0, i = 1, \ldots, m \tag{A.9}$$

$$f_i(x^*) \leq 0, i = 1, \ldots, m \tag{A.10}$$

Appendix B

Well-SSVM: from primal (4.13) to dual (4.15)

Given the Well-SSVM primal (Eq. 14), we can write its Lagrangian as:

$$\begin{aligned} L(w,\xi,\alpha,\beta) &= \frac{1}{2}\sum_{l=1}^{L}\frac{\|w^l\|^2}{\mu^l} + C\sum_{i=1}^{N}\xi_i + \\ - &\sum_{i=1}^{N}\sum_{u\in W_i}\alpha_{i,u}\left(\sum_{l=1}^{L}\left(w^{lT}\psi(x_i,y_i^l) - w^{lT}\psi(x_i,u) - \mu^l\Delta(y_i^l,u)\right) + \xi_i\right) \\ - &\sum_{i=1}^{N}\beta_i\xi_i \qquad\qquad s.t.\quad 0\leq\alpha_{i,u},\beta_i \quad \forall i,\forall u\in W_i \end{aligned} \tag{B.1}$$

To compact notations, we pose $\delta\psi_i^l(u) \equiv \psi(x_i,y_i^l) - \psi(x_i,u)$. Differentiating in w^l and equating to zero, we obtain:

$$\begin{aligned} &\frac{\partial L}{\partial w^l} = \frac{w^l}{\mu^l} - \sum_{i=1}^{N}\sum_{u\in W_i}\alpha_{i,u}\delta\psi_i^l(u) = 0 \Rightarrow w^l = \mu^l\sum_{i=1}^{N}\sum_{u\in W_i}\alpha_{i,u}\delta\psi_i^l(u) \\ &\frac{\partial L}{\partial \xi_i} = C - \sum_{u\in W_i}\alpha_{i,u} - \beta_i = 0 \end{aligned} \tag{B.2}$$

Using these results to eliminate w and ξ (and, indirectly, β) from $L(w, \xi, \alpha, \beta)$, we obtain the dual problem as:

$$\begin{aligned}
&\max_{\alpha} \sum_{l=1}^{L} \mu^l \left(-\frac{1}{2} \sum_{i,u \in W_i} \sum_{j,v \in W_j} \alpha_{i,u} \alpha_{j,v} \delta\psi_i^l(u)^T \delta\psi_j^l(v) + \sum_{i,u \in W_i} \alpha_{i,u} \Delta(y_i^l, u)) \right) = \\
&= \max_{\alpha} -\frac{1}{2} \sum_{i,u \in W_i} \sum_{j,v \in W_j} \alpha_{i,u} \alpha_{j,v} \tilde{\delta\psi}_i(u^{(\times L)})^T \tilde{\delta\psi}_j(v^{(\times L)}) + \sum_{i,u \in W_i} \alpha_{i,u} \tilde{\Delta}(y_i^{1:L}, u) \\
&s.t.\ 0 \leq \alpha_{i,u} \leq C \quad \forall i, \forall u \in W_i \\
&= \max_{\alpha} \sum_{l=1}^{L} \mu^l G(\alpha, y^l, W) \quad \text{i.e., Well-SSVM (4.15)}
\end{aligned} \tag{B.3}$$

where we have used position $\tilde{\delta\psi}_i(u^{(\times L)})^T \tilde{\delta\psi}_j(v^{(\times L)}) = \sum_{l=1}^{L} \mu_l \delta\psi_i^l(u)^T \delta\psi_j^l(v)$ and the positions for (14). $\square$

Appendix C

WellSSVM via M^3N

In Appendix (C), we provide a comprehensive table of notations that is helpful while translating notations from Taskar et al. (2004) to our case. It is followed by a simple example of factorized dual. Finally, we present the derivation of $\mathcal{H}$, τ and Δ previously mentioned in Section (5.5.2).

Our Notations	M^3N Notations (Taskar et al. (2004))	Explanation
i	x	iterating variable over sequence
$\bar{y}$	y	arbitrary (or predicted) labeling
T	l	number of nodes in a sequence
t	i	iterating variable over nodes
N	m	size of training set
x_i	$x^{(i)}$	i^{th} sequence
y_i	$t(x^{(i)}) = y^{(i)}$	groundtruth labeling for a sequence
y^t	y_i	value of particular node
y_i^t	$(t(x))_i$	groundtruth value of a particular node
$\Delta(y_i, y)$	$\sum_{i=1}^{l} \Delta t_x(y_i)$	loss over a sequence
$\psi(x, y)$	$f_x(y)$	feature function for sequence x with labeling y
$\delta(\psi_i(y)) = \psi(x, y_i) - \psi(x, y)$	$\Delta f_x(y)$	difference of feature function ($\Delta f_x(y) = f(x, t(x)) - f(x, y)$ (Taskar et al. (2004)))

Table C.1: Summary of notations.

C.1 Factorized Dual

We will start from the N-slack formulation of Multiclass SVM with margin rescaling. The dual of N-slack formulation in Taskar et al. (2004) is given as follows:

$$\begin{aligned} &\text{maximize} \sum_{x,y} \alpha_x(y)\Delta t_x(y) - \frac{1}{2}||\sum_{x,y} \alpha_x(y)\Delta f_x(y)||^2 \\ &s.t. \ \sum_{y} \alpha_x(y) = C, \forall x; \quad \alpha_x(y) \geq 0, \forall x, y. \end{aligned}$$

Please note that in the above formulation, an extra dual variable $\alpha_x(t(x))$ is added. To maintain consistency in notations with the rest of this thesis, we express the above QP as follows (an extra dual variable $\alpha_{i\bar{y}_i}$ is added for each example):

$$\begin{aligned} &\text{maximize} \sum_{i,\bar{y}} \alpha_{(i\bar{y})}\Delta(y_i, \bar{y}) - \frac{1}{2}||\sum_{i,\bar{y}} \alpha_{(i\bar{y})}\delta(\psi_i(\bar{y}))||^2 \\ &s.t. \ \sum_{\bar{y}} \alpha_{(i\bar{y})} = C, \forall i; \quad \alpha_{(i\bar{y})} \geq 0, \forall i, \bar{y}. \end{aligned} \tag{C.1}$$

The only purpose of presenting such equivalence is to clarify notations used throughout the thesis. For more details, please refer to Table (C.1).

C.2 Implementation of $\mathcal{H}$, τ and Δ

In the following text, a detailed explanation is given for the definition of $\mathcal{H}$, τ and Δ previously mentioned in Section (5.5.2). The main goal is to represent the dual of M^{3}N (Objective (C.1)) as a quadratic function of ground-truth labeling vector y i.e. $y^\top(\mathcal{H}y + \tau + \Delta)$.

For the sake of simplicity, let us revisit the encoding of a labeling vector y. Consider a sequence of three frames, $x_i = \{x_i^1, x_i^2, x_i^3\}$, with labels $\{1, 1, 1\}$; then the encoded label vector would be:

$$y_i = \{ \underbrace{10}_{\text{emission for frame-1}} \underbrace{10}_{\text{emission for frame-2}} \underbrace{10}_{\text{emission for frame-3}} \underbrace{1000}_{\text{frame-1} \rightarrow \text{frame-2}} \underbrace{1000}_{\text{frame-2} \rightarrow \text{frame-3}} \}.$$

While deriving the parameter $\mathcal{H}$, τ and Δ, we will assume the aforementioned encoding of labeling vector although it is not a necessary condition. Such an assumption will make the task of deriving the desired parameters simpler.

Let us start from the dual function of M^{3}N (Objective (5.5)):

$$G(\lambda) = \underbrace{\sum_i \sum_{a,u^a} \lambda_i(u^a)\Delta t_i(u^a)}_{\text{Linear in } \lambda} - \underbrace{\frac{1}{2}\sum_{i,j} \sum_{\substack{(a,b)\\u^a,u^b}} \sum_{\substack{(c,d)\\v^c,v^d}} \lambda_i(u^a,u^b)\lambda_j(v^c,v^d)\delta\psi_i(u^a,u^b)^\top \delta\psi_j(v^c,v^d)}_{\text{Quadratic in } \lambda} \tag{C.2}$$

First of all, consider the linear part of G(λ):

$$\sum_i \sum_{a,u^a} \lambda_i(u^a)\Delta t_i(u^a)$$
$$= \sum_i \sum_a [\lambda_i(u^a = 0)\Delta t_i(u^a = 0) + \lambda_i(u^a = 1)\Delta t_i(u^a = 1)]$$

Terms consistent with the ground-truth labeling would contribute to G(λ), therefore:

$$= y^\top \Delta \tag{C.3}$$

where, for every sequence i we will form Δ as follows:

$$\Delta^{\top} = \Bigg[\underbrace{\{\lambda_i(u^a = 0), \lambda_i(u^a = 1)\}}_{\text{for } a^{th} \text{ frame}} \quad \underbrace{\mathbf{0} \in R^{4T_i - 1}}_{\text{All zeros for transitions between frames}} \Bigg]$$

For each sequence i, terms enclosed inside $\{\}$ will be expanded using the operation of concatenation while a vector of zeros, i.e. $\mathbf{0} \in R^{4T_i-1}$ will follow them. This operation has to be repeated for all sequences i.e. $i = 1, \ldots, N$.

To clarify further, let us consider a case of a single sequence ($N = 1$) with the ground-truth labeling $y = \{1, 2, 1\}$. After encoding, we can rewrite y as: $y = [10011001000010]^{\top}$. If we formulate Δ as per our proposal above, we will eventually see that the quantity $y^{\top}\Delta$ does equate to $\sum_{a,u^a} \lambda_i(u^a)\Delta t_i(u^a)$. We will emphasize again that this technique is only suitable for Hamming loss and more specifically, our encoding of labeling vector is central to the correct computation of $\sum_i \sum_{a,u^a} \lambda_i(u^a)\Delta t_i(u^a)$.

Now, consider the quadratic part of $\mathrm{G}(\lambda)$:

$$\frac{1}{2}\sum_{i,j}\sum_{\substack{(a,b)\\u^a,u^b}}\sum_{\substack{(c,d)\\v^c,v^d}}\lambda_i(u^a,u^b)\lambda_j(v^c,v^d)\delta\psi_i(u^a,u^b)^\top\delta\psi_j(v^c,v^d)$$

$$=\frac{1}{2}\sum_{i,j}\sum_{\substack{(a,b)\\u^a,u^b}}\sum_{\substack{(c,d)\\v^c,v^d}}\lambda_i(u^a,u^b)\lambda_j(v^c,v^d)\left[\psi_i(x_i,y_i^a,y_i^b)-\psi_i(x_i,u_i^a,u_i^b)\right]^\top\left[\psi_j(x_j,y_j^c,y_j^d)-\psi_j(x_j,v_j^c,v_j^b\right.$$

$$=\frac{1}{2}\sum_{i,j}\sum_{\substack{(a,b)\\u^a,u^b}}\sum_{\substack{(c,d)\\v^c,v^d}}\lambda_i(u^a,u^b)\lambda_j(v^c,v^d)\left[h_i^{ab\top}y_i^{ab}-h_i^{ab\top}u_i^{ab}\right]^\top\left[h_j^{cd\top}y_j^{cd}-h_j^{cd\top}v_j^{cd}\right]$$

$$=\frac{1}{2}\sum_{i,j}\sum_{\substack{(a,b)\\u^a,u^b}}\sum_{\substack{(c,d)\\v^c,v^d}}\lambda_i(u^a,u^b)\lambda_j(v^c,v^d)\left[\left(h_i^{ab\top}y_i^{ab}\right)^\top\left(h_j^{cd\top}y_j^{cd}\right)-\left(h_i^{ab\top}y_i^{ab}\right)^\top\left(h_j^{cd\top}v_j^{cd}\right)-\right.$$

$$\left.\left(h_i^{ab\top}u_i^{ab}\right)^\top\left(h_j^{cd\top}y_j^{cd}\right)+\left(h_i^{ab\top}u_i^{ab}\right)^\top\left(h_j^{cd\top}v_j^{cd}\right)\right]$$

Neglecting the last term as it is independent of y during the maximization of $\mathrm{G}(\lambda)$

$$=\frac{1}{2}\sum_{i,j}\sum_{\substack{(a,b)\\u^a,u^b}}\sum_{\substack{(c,d)\\v^c,v^d}}\lambda_i(u^a,u^b)\lambda_j(v^c,v^d)\left[\left(h_i^{ab\top}y_i^{ab}\right)^\top\left(h_j^{cd\top}y_j^{cd}\right)-\left(h_i^{ab\top}y_i^{ab}\right)^\top\left(h_j^{cd\top}v_j^{cd}\right)-\right.$$

$$\left.\left(h_i^{ab\top}u_i^{ab}\right)^\top\left(h_j^{cd\top}y_j^{cd}\right)\right]$$

$$=\frac{1}{2}\sum_{i,j}\sum_{\substack{(a,b)\\u^a,u^b}}\sum_{\substack{(c,d)\\v^c,v^d}}\lambda_i(u^a,u^b)\lambda_j(v^c,v^d)\left[\left(h_i^{ab\top}y_i^{ab}\right)^\top\left(h_j^{cd\top}y_j^{cd}\right)-2\left(h_i^{ab\top}y_i^{ab}\right)^\top\left(h_j^{cd\top}v_j^{cd}\right)\right]$$

$$=\underbrace{\frac{1}{2}\sum_{i,j}\sum_{\substack{(a,b)\\u^a,u^b}}\sum_{\substack{(c,d)\\v^c,v^d}}\lambda_i(u^a,u^b)\lambda_j(v^c,v^d)\left[\left(h_i^{ab\top}y_i^{ab}\right)^\top\left(h_j^{cd\top}y_j^{cd}\right)\right]}_{1^{st}Term}$$

$$\underbrace{-\sum_{i,j}\sum_{\substack{(a,b)\\u^a,u^b}}\sum_{\substack{(c,d)\\v^c,v^d}}\lambda_i(u^a,u^b)\lambda_j(v^c,v^d)\left(h_i^{ab\top}y_i^{ab}\right)^\top\left(h_j^{cd\top}v_j^{cd}\right)}_{2^{nd}Term} \tag{C.4}$$

For the definition of $\mathcal{H}$, let us consider the 1^{st} term of Equation (C.4):

$$= \frac{1}{2} \sum_{i,j} \sum_{\substack{(a,b) \\ u^a,u^b}} \sum_{\substack{(c,d) \\ v^c,v^d}} \lambda_i(u^a,u^b)\lambda_j(v^c,v^d) \left[\left(h_i^{ab\top} y_i^{ab} \right)^\top \left(h_j^{cd\top} y_j^{cd} \right) \right]$$

$$= \frac{1}{2} \sum_{i,j} \sum_{\substack{(a,b) \\ u^a,u^b}} \sum_{\substack{(c,d) \\ v^c,v^d}} \left[\lambda_i(u^a,u^b) \left(h_i^{ab\top} y_i^{ab} \right)^\top \left(h_j^{cd\top} y_j^{cd} \right) \lambda_j(v^c,v^d) \right]$$

$$= \frac{1}{2} \sum_{i,j} \sum_{\substack{(a,b) \\ u^a,u^b}} \sum_{\substack{(c,d) \\ v^c,v^d}} \left[\lambda_i(u^a,u^b) y_i^{ab\top} h_i^{ab} h_j^{cd\top} y_j^{cd} \lambda_j(v^c,v^d) \right]$$

$$= \frac{1}{2} \sum_{i,j} \left[\left(\sum_{\substack{(a,b) \\ u^a,u^b}} \lambda_i(u^a,u^b) y_i^{ab\top} h_i^{ab} \right) \left(\sum_{\substack{(c,d) \\ v^c,v^d}} h_j^{cd\top} y_j^{cd} \lambda_j(v^c,v^d) \right) \right]$$

Given that $\sum_{u^a,u^b} \lambda(u^a,u^b) = C$

$$= \frac{1}{2} \sum_{i,j} \left[\left(\sum_{(a,b)} C y_i^{ab\top} h_i^{ab} \right) \left(\sum_{(c,d)} C h_j^{cd\top} y_j^{cd} \right) \right]$$

If we design h_i^{ab} as follows where $h_i^{ab} \in R^{6\times(2D+4)}$ (Section (5.4)):

$$h_i^{ab} = \begin{bmatrix} Cx_i^b & 0^D & 0 & 0 & 0 & 0 \\ 0^D & Cx_i^b & 0 & 0 & 0 & 0 \\ 0^D & 0^D & C & 0 & 0 & 0 \\ 0^D & 0^D & 0 & C & 0 & 0 \\ 0^D & 0^D & 0 & 0 & C & 0 \\ 0^D & 0^D & 0 & 0 & 0 & C \end{bmatrix}$$

then we can easily rewrite the first term of Equation (C.2) as:

$$= \frac{1}{2} \sum_{i,j} \sum_{\substack{(a,b) \\ u^a, u^b}} \sum_{\substack{(c,d) \\ v^c, v^d}} \left[y_i^{ab\top} h_i^{ab} h_j^{cd\top} y_j^{cd} \right] (C \text{ has been absorbed in } h_i^{ab})$$

$$= \frac{1}{2} \sum_{i,j} \left[y_i^\top \underbrace{h_i h_j^\top}_{H_{ij}} y_j \right]$$

After the concatenation of label vector $y = [y_1, y_2, \ldots, y_N]$ and
the formation of $\mathcal{H}$ as the matrix of blocks H_{ij}; we can write the above summation as:

$$= \frac{1}{2} y^\top \mathcal{H} y \qquad \left(\text{comment: } \frac{1}{2} \text{ can easily be absorbed inside } \mathcal{H} \right)$$

$$= y^\top \mathcal{H} y$$

Similarly, for the definition of τ, consider the 2^{nd} term of Equation (C.4) which is linear in y:

$$-\sum_{i,j}\sum_{\substack{(a,b)\\u^a,u^b}}\sum_{\substack{(c,d)\\v^c,v^d}}\lambda_i(u^a,u^b)\lambda_j(v^c,v^d)\left(h_i^{ab\top}y_i^{ab}\right)^\top\left(h_j^{cd\top}v_j^{cd}\right)$$

$$=-\sum_{i,j}\left(\sum_{\substack{(a,b)\\u^a,u^b}}\lambda_i(u^a,u^b)h_i^{ab\top}y_i^{ab}\right)^\top\left(\sum_{\substack{(c,d)\\v^c,v^d}}\lambda_j(v^c,v^d)h_j^{cd\top}v_j^{cd}\right)$$

After formulating h_i^{ab} as mentioned above:

$$=-\sum_{i,j}\left(\sum_{(a,b)}h_i^{ab\top}y_i^{ab}\right)^\top\left(\sum_{(c,d)}h_j^{cd\top}\sum_{v^c,v^d}\lambda_j(v^c,v^d)v_j^{cd}\right)$$

Let $\kappa_j^{cd}=\sum_{v^c,v^d}\lambda_j(v^c,v^d)v_j^{cd}$

$$=-\sum_{i,j}\left(\sum_{(a,b)}h_i^{ab\top}y_i^{ab}\right)^\top\left(\sum_{(c,d)}h_j^{cd\top}\kappa_j^{cd}\right)$$

$$=-\left(\sum_{i}\sum_{(a,b)}h_i^{ab\top}y_i^{ab}\right)^\top\left(\sum_{j}\sum_{(c,d)}h_j^{cd\top}\kappa_j^{cd}\right)$$

$$=-\left(\sum_{i}h_i{}^\top y_i\right)^\top\left(\sum_{j}\sum_{(c,d)}h_j^{cd\top}\kappa_j^{cd}\right)$$

$$=-\left(\sum_{i}y_i{}^\top h_i\right)\left(\sum_{j}\sum_{(c,d)}h_j^{cd\top}\kappa_j^{cd}\right)$$

$$=-\sum_{i}y_i{}^\top\left(h_i\sum_{j}\sum_{(c,d)}h_j^{cd\top}\kappa_j^{cd}\right)$$

After the concatenation of label vector $y=[y_1,y_2,\ldots,y_N]$ and the formation of τ as follows:

$$\tau=\left[-h_1\left(\sum_{j}\sum_{(c,d)}h_j^{cd\top}\kappa_j^{cd}\right),-h_2^\top\left(\sum_{j}\sum_{(c,d)}h_j^{cd\top}\kappa_j^{cd}\right),\ldots,-h_N^\top\left(\sum_{j}\sum_{(c,d)}h_j^{cd\top}\kappa_j^{cd}\right)\right]$$

$$=-\sum_{i}y_i{}^\top\left(h_i\sum_{j}\sum_{(c,d)}h_j^{cd\top}\kappa_j^{cd}\right)=y^\top\tau$$

Hence, we have successfully transformed the quadratic part of Equation (C.2) into $y^\top(\mathcal{H}y+\tau)$ while the linear part of Equation (C.2) has already been transformed into $y^\top\Delta$. Therefore, we can rewrite Equation (C.2) as quadratic function of groundtruth labeling y as:

$$G(\lambda) = y^\top(\mathcal{H}y + \tau + \Delta)$$

C.3 Pictorial Representation of h and $\mathcal{H}$

We know the fact that feature function $\psi_i(x_i, y_i)$ is decomposable over the edges of graph (or a sequence in our examples). Therefore, we can rewrite $\psi_i(x_i, y_i)$ as follows:

$$\begin{aligned}\psi(x_i, y_i) &= \sum_{(a,b)} {h_i^{ab}}^\top y_i^{ab} \\ &= h_i^\top y_i\end{aligned}$$

Whenever h_i^{ab} has appeared in the dual equation (Equation (C.2)), it was always found with its corresponding multiplier $\lambda_i(u^b)$. Therefore, for a sequence of three frames, $x_i = \{x_i^a, x_i^b, x_i^c\}$ where $x_i^t \in R^D$, the equivalent h-matrix $\in R^{(2T_i+4(T_i-1))\times(2D+4)}$ is given as follows:

$$h_i = \begin{bmatrix} Cx_i^a & \mathbf{0} \in R^D & 0 & 0 & 0 & 0 \\ \mathbf{0} \in R^D & Cx_i^a & 0 & 0 & 0 & 0 \\ Cx_i^b & \mathbf{0} \in R^D & 0 & 0 & 0 & 0 \\ \mathbf{0} \in R^D & Cx_i^b & 0 & 0 & 0 & 0 \\ Cx_i^c & \mathbf{0} \in R^D & 0 & 0 & 0 & 0 \\ \mathbf{0} \in R^D & Cx_i^c & 0 & 0 & 0 & 0 \\ \mathbf{0} \in R^D & \mathbf{0} \in R^D & C & 0 & 0 & 0 \\ \mathbf{0} \in R^D & \mathbf{0} \in R^D & 0 & C & 0 & 0 \\ \mathbf{0} \in R^D & \mathbf{0} \in R^D & 0 & 0 & C & 0 \\ \mathbf{0} \in R^D & \mathbf{0} \in R^D & 0 & 0 & 0 & C \\ \mathbf{0} \in R^D & \mathbf{0} \in R^D & C & 0 & 0 & 0 \\ \mathbf{0} \in R^D & \mathbf{0} \in R^D & 0 & C & 0 & 0 \\ \mathbf{0} \in R^D & \mathbf{0} \in R^D & 0 & 0 & C & 0 \\ \mathbf{0} \in R^D & \mathbf{0} \in R^D & 0 & 0 & 0 & C \end{bmatrix}$$

Hence, for a case of three sequences i.e. $N = 3$; its corresponding $\mathcal{H}$-matrix will consist of blocks $H_{ij} = h_i h_j^\top$ as follows:

$$\mathcal{H} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}$$

Bibliography

Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas Guibas, and Li Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 1331–1338. IEEE, 2011a.

Jianxiong Xiao, James Hays, Krista Ehinger, Aude Oliva, Antonio Torralba, et al. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010.

Massimo Piccardi. The support vector machine (svm)and the structural svm, 2013. URL www-staff.it.uts.edu.au/~massimo/ShortCourseSPR/SPR_08_SVM_v3.pdf.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Andrew Y Ng. Support vector machines, 2003. URL http://cs229.stanford.edu/notes/cs229-notes3.pdf.

John Platt et al. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methodssupport vector learning*, 3, 1999.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

Trevor Hastie, Robert Tibshirani, et al. Classification by pairwise coupling. *The annals of statistics*, 26(2):451–471, 1998.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002.

Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 2004.

Alain Rakotomamonjy, Universit De Rouen, Francis Bach, Stphane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

Mehmet Gönen and Ethem Alpaydin. Localized multiple kernel learning. In *Proceedings of the 25th international conference on Machine learning*, pages 352–359. ACM, 2008.

Xinxing Xu, Ivor W Tsang, and Dong Xu. Soft margin multiple kernel learning. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(5):749–761, 2013.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484, 2005a.

Chun Nam Yu. *Improved learning of structural support vector machines: training with latent variables and nonlinear kernels.* PhD thesis, Cornell University, 2011.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. *Advances in neural information processing systems*, 16:25, 2004.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Mach. Learn.*, 77(1):27–59, 2009a.

Chun-Nam John Yu and Thorsten Joachims. Learning structural SVMs with latent variables. In *ICML*, pages 1169–1176. ACM, 2009.

Alan L Yuille and Anand Rangarajan. The concave-convex procedure. *Neural computation*, 15(4):915–936, 2003.

R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

Guodong Guo and Alice Lai. A survey on still image based human action recognition. *Pattern Recognition*, in press(0):–, 2014.

A. Gupta, A. Kembhavi, and L.S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1775–1789, 2009a.

Fadime Sener, Cagdas Bas, and Nazli Ikizler-Cinbis. On recognizing actions in still images via multiple features. In *Computer Vision ECCV 2012. Workshops and Demonstrations*, volume 7585 of *Lecture Notes in Computer Science*, pages 263–272, 2012.

Yang Wang, Hao Jiang, Mark S Drew, Ze-Nian Li, and Greg Mori. Unsupervised discovery of action classes. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1654–1661. IEEE, 2006.

Christian Thurau and Václav Hlaváč. Pose primitive based human action recognition in videos or still images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

Nazli Ikizler, R. Gokberk Cinbis, Selen Pehlivan, and Pinar Duygulu. Recognizing actions from still images. In *19th International Conference on Pattern Recognition, ICPR 2008*, pages 1–4, 2008.

D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. ISSN 0920-5691.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 886–893, June 2005.

Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *NIPS*, volume 2, page 3, 2000.

V. Delaitre, I. Laptev, and J. Sivic. Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *Proceedings of the British Machine Vision Conference*, pages 1–11, 2010.

I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2):107–123, 2005. ISSN 0920-5691.

Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vision*, 42(3):145–175, May 2001.

N. Ikizler-Cinbis, R.G. Cinbis, and S. Sclaroff. Learning actions from the web. In *2009 IEEE 12th International Conference on Computer Vision*, pages 995–1002. IEEE, 2009.

Bangpeng Yao and Li Fei-Fei. Grouplet: A structured image representation for recognizing human and object interactions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9–16, 2010.

Weilong Yang, Yang Wang, and Greg Mori. Recognizing human actions from still images with latent poses. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2030–2037, 2010.

Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas J. Guibas, and Li Fei-Fei. Action recognition by bases of action attributes and parts. In *International Conference on Computer Vision (ICCV)*, pages 1331–1338, 2011b.

M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67–92, January 1973.

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, September 2010. ISSN 0162-8828.

Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004.

Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Ninth IEEE International Conference on Computer Vision*, volume 2, 2003.

G. Mori. Guiding model search using segmentation. In *Proceedings of the Tenth IEEE International Conference on Computer Vision*, volume 2, pages 1417–1423, 2005.

Deli Pei, Zhenguo Li, Rongrong Ji, and Fuchun Sun. Efficient semantic image segmentation with multi-class ranking prior. *Computer Vision and Image Understanding*, 2013.

Dong Han, Liefeng Bo, and Cristian Sminchisescu. Selection and context for action recognition. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1933–1940, 2009.

Muhammad Muneeb Ullah, Sobhan Naderi Parizi, and Ivan Laptev. Improving bag-of-features action recognition with non-local cues. In *BMVC*, volume 10, pages 95–1. Citeseer, 2010.

A Criminisi. Microsoft research cambridge object recognition image database,http://research.microsoft.com/en-us/projects/objectclassrecognition/, 2004.

A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, 2005b.

Gregory F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(23):393 – 405, 1990.

Gaurav Sharma, Frédéric Jurie, and Cordelia Schmid. Expanded parts model for human attribute and action recognition in still images. In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–659. IEEE, 2013.

Fahad Shahbaz Khan, Rao Muhammad Anwer, Joost van de Weijer, Andrew D Bagdanov, Antonio M Lopez, and Michael Felsberg. Coloring action recognition in still images. *International Journal of Computer Vision*, 105(3):205–221, 2013.

Leonid Sigal, Alexandru O. Balan, and Michael J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *Int. J. Comput. Vision*, 87(1-2):4–27, March 2010.

Abhinav Gupta, Aniruddha Kembhavi, and Larry S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1775–1789, 2009b.

Vladimir N. Vapnik. *Statistical learning theory*. Wiley, 1998.

Pedro F. Felzenszwalb, David A. McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

Alexander G. Schwing, Tamir Hazan, Marc Pollefeys, and Raquel Urtasun. Efficient structured prediction with latent variables for general graphical models. In *ICML*, 2012.

Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

Long (Leo) Zhu, Yuanhao Chen, Alan Yuille, and William Freeman. Latent hierarchical structural learning for object detection. In *CVPR*, pages 1062–1069, 2010.

Yang Wang and Greg Mori. Hidden part models for human action recognition: Probabilistic vs. max-margin. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1310–1323, 2011.

Huizhong Duan, Yanen Li, ChengXiang Zhai, and Dan Roth. A discriminative model for query spelling correction with latent structural svm. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1511–1521, 2012.

Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *NIPS*, pages 1537–1544, 2005.

Linli Xu and Dale Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *AAAI*, pages 904–910, 2005.

Linli Xu, Dana Wilkinson, and Dale Schuurmans. Discriminative unsupervised learning of structured predictors. In *ICML*, pages 1057–1064, 2006.

Yu-Feng Li, Ivor W Tsang, James T Kwok, and Zhi-Hua Zhou. Tighter and convex maximum margin clustering. In *AISTATS 2009*, pages 344–351, 2009.

Yu-Feng Li, Ivor W Tsang, James T Kwok, and Zhi-Hua Zhou. Convex and scalable weakly labeled SVMs. *Journal of Machine Learning Research*, 14(1):2151–2188, 2013.

Renata CB Madeo, Clodoaldo AM Lima, and Sarajane M Peres. Gesture unit segmentation using support vector machines: segmenting gestures from rest positions. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 46–52. ACM, 2013.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Mach. Learn.*, 77(1):27–59, October 2009b.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39 (1):1–38, 1977.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.