Incorporating Supervision for Visual Recognition and Segmentation

Alex Yu Jen Shyr

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2011-116
http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-116.html

November 4, 2011

Incorporating Supervision for Visual Recognition and Segmentation

by

Alex Yu Jen Shyr

B.S. (University of British Columbia) 2006

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Michael I. Jordan, Chair
Professor Trevor Darrell
Professor Raquel Urtasun
Professor Bruno Olshausen

Fall 2011

The dissertation of Alex Yu Jen Shyr is approved.

Professor Michael I. Jordan, Chair	Date
Professor Trevor Darrell	Date
Professor Raquel Urtasun	Date
Professor Bruno Olshausen	Date

University of California, Berkeley
Fall 2011

Incorporating Supervision for Visual Recognition and Segmentation

Alex Yu Jen Shyr

Abstract

Incorporating Supervision for Visual Recognition and Segmentation

by

Alex Yu Jen Shyr

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Michael I. Jordan, Chair

Unsupervised algorithms which do not make use of labels are commonly found in computer vision and are widely applicable to all problem settings. In the presence of expert-labeled ground truth information, however, these algorithms are not optimal. Altering the unsupervised models to include labels is not always a straight forward modification. In this dissertation, we explore various ways to incorporate human supervision.

We first start with the task of visual sequence recognition and demonstrate ways to effectively make use of temporal information. Next, we tackle the problem of scene segmentation and devise a novel framework to discriminatively train a generative hierarchical model with nonparametric Bayesian priors; the methodology can be easily applied to other nonparametric Bayesian models. Finally, we approach the difficult problem of object segmentation and describe how shape priors can be infused into a generative Bayesian segmentation model. We demonstrate the effectiveness of our models and algorithms on datasets which are widely used by the research community and universally regarded as difficult. The dissertation concludes with active venues for future research.

Professor Michael I. Jordan
Dissertation Committee Chair

Contents

List of Figures

List of Tables

Acknowledgements

The completion of my doctorate program would have been impossible for a number of people. My three advisors have each given me directions and support vital to my development. Professor Mike Jordan has guided me from the beginning, nurturing in me a solid statistical background as well as encouraging me in my search for a suitable application domain. He is consistently the best resource for visionary research directions and points out every opportunity worthy of further investigation. Professors Trevor Darrell and Raquel Urtasun stepped in when I was searching for an application domain and assisted my transition to computer vision. Trevor brings with him the broad familiarity with all aspects of computer vision and a probing personality to always ask challenging questions; Raquel, on the other hand, worked closely with me and was a bridge between my machine learning background and the vision domain. She taught me to carry out experiments with an almost fanatic attention to detail and introduced me to her discipline for publication.

I also want to thank my colleagues, notably Kurt, Wei-Chun and Daniel, for priceless brainstorming sessions and discussions, as well as providing moral support during depressing times. Faculty members and researchers (Erik, Fei, Laurent and Ling) have helped shape my research direction as well. In particular, Erik's segmentation model makes up the bulk of my dissertation and the completion of my degree is not possible without his help.

Lastly, I would like to thank my family for their constant encouragements, and my girlfriend Harmony for her tireless support and caring.

Chapter 1

Introduction

With the rising popularity of distributed crowd-sourcing infrastructures such as Amazon Mechanical Turk, metadata in the form of annotations and labels are becoming more financially feasible to obtain and more readily available. In the domain of computer vision alone, labels come in many diverse forms, ranging from instance-level (categorizing an entire image) to pixel-level (labeling parts of an image) to semantic labels (providing meaningful descriptions of objects). Time-series data such as video sequences add yet another form of annotation to the mix.

As the amount of human labels increase, it is necessary for models to take advantage of the extra sources of information. *Unsupervised models* take unlabeled training data as input, and seek to describe the key features or characteristics of the data. Clustering, density estimation and principal component analysis (PCA) are popular examples of such models. These algorithms do not require ground truth labels and are applicable in all settings. However, since these models are task-independent, they suffer in the final classification or estimation performance. *Supervised models*, on the other hand, require labels alongside the training data, and seek to learn an optimal mapping function from the data space to the label space. These algorithms have the benefit of knowing the task at hand, and thus they often outperform their unsupervised counterparts. It should be noted that "supervised models" here refer to any model which makes use of labeled information. The level of supervision may vary, resulting in different subcategories of supervised algorithms such as semi-supervised learning, active/reinforcement learning, etc.

In this dissertation, we present novel frameworks for incorporating labels and supervision into existing models. We first address the problem of time-series classification, and show how to effectively include temporal information into an existing supervised model. We then tackle the segmentation problem and demonstrate how to incorporate labels into a previously unsupervised Bayesian model. The extension is carried out in two ways — via discriminative training using segmentation labels, and infusion of a shape prior to capture shape-level object properties. Although the particular models of choice might vary, we hope that our frameworks can be flexible and widely applicable.

The outline of the dissertation is as follows. In chapter 2, we provide a brief summary of the prior work in the relevant areas of dimensionality reduction, segmentation models and shape analysis techniques. In chapter 3, we begin with the task of sequence recognition, which involves

assigning an entire sequence with a category-level label. Our approach to the problem lies within a dimensionality reduction framework. We propose two different ways to include labels that are particularly effective for visual sequences, one through kernel design and the other via dynamic time warping (DTW). Since sequence alignment is a major component of analyzing any sequential data, we find that the approach with DTW is optimal. The contribution of our work is a method of estimating a low dimensional subspace which efficiently captures label and temporal information. Such a method can be used as a preprocessing step for many types of sequential data, and can improve the performance and computational efficiency for many existing classification algorithms. In our experiments, for simplicity, we assume the data has been segmented —that is, the object of interest has been identified and the background cropped away. We will relax this assumption in the latter sections of the dissertation.

The remaining chapters are devoted to the task of image segmentation, which partitions an image into homogeneous regions often resembling objects. The problem of segmentation is arguably more difficult than instance-level recognition, as each pixel requires a label. In chapter 4, we describe the fully Bayesian, generative Hierarchical Pitman-Yor (HPY) model introduced by (94), which will be our framework of choice due to its ability to model the empirically heavy-tailed distributions of visual statistics. In chapter 5, we apply the HPY model to the problem of scene segmentation, which involves segmenting the entire image, including foreground objects and background. We devise a novel way to discriminatively train the model with a nonparametric Bayesian prior; in particular, we constrain the original variational learning problem to obey segment-level and class-level annotations. Discriminative training of a nonparametric Bayesian model is itself still an open research problem, as various extensions of the graphical model have been proposed but no singular one has been proven superior in practice. Our contribution offers an alternative framework which leaves the graphical model intact, while directly guiding the model's MAP estimates to the correct ones during training.

In chapter 6, we further the extension of the HPY model and tackle the harder problem of object segmentation, which separates only a select number of foreground objects from background clutter. We augment our previously texture-based model with a shape prior, which attempts to capture global object properties. Observing that it is essential to perform quick shape matching and estimate pixel membership probability of a shape, we characterize the shape prior with a dual contour and mask representations. To produce these representations directly from shape masks, we adopt the Procrustes analysis to perform clustering with invariance to affine transformation. The final model is a segmentation model with a nonparametric Bayesian prior and a probabilistic shape prior. The primary contribution of this work is our introduction of a shape prior into a Bayesian model; a secondary contribution lies in our formulation of the k-Procrustes clustering for shapes.

The dissertation ends with a concluding analysis of our contributions to the field of supervised learning and visual recognition/segmentation, as well as venues of future research.

Chapter 2

Related Work

We will review prior research methods related to our work, which roughly spans across three major fields of study: dimensionality reduction, segmentation models and shape analysis techniques. Our first task is sequence recognition, and we have opted for the dimensionality reduction approach.

2.1 Dimensionality Reduction

A popular approach for dealing with high dimensional data is to learn a low dimensional representation. The motivation behind such a learning method is often that inference can be done more accurately and from a smaller number of examples on the learned low dimensional subspace, as the irrelevant or noisy dimensions will be appropriately weighted down. The reduction in memory requirement also increases the computational efficiency of the overall algorithm. The different dimensionality reduction techniques can be coarsely categorized as unsupervised and supervised, depending on whether the algorithm is task-dependent.

2.1.1 Unsupervised Models

Unsupervised learning techniques have been applied to computer vision problems with significant success. One of the most popular algorithm is the *Principal Component Analysis (PCA)* (40) due to its simple form, wide applicability and ease of implementation. Let the data be $\mathbf{X}$ and the lower dimensional representation be $\mathbf{Z}$. PCA seeks to find a linear projection $\mathbf{W}$ (ie., $\mathbf{Z} = \mathbf{WX}$), where the variance of the projected data is maximized. This is equivalent to the *Probabilistic Principal Component Analysis (PPCA)* (98), which assumes that data $\mathbf{X}$ is generated via a linear projection of the latent representation

$$\mathbf{X} = \mathbf{WZ} + \mathbf{b} + \epsilon. \tag{2.1}$$

Both $\mathbf{Z}$ and ϵ are assumed to be generated from a Gaussian distribution, $\mathbf{Z} \sim \mathcal{N}(\mathbf{Z}|0, \mathbf{I})$ and $\epsilon \sim \mathcal{N}(\epsilon|0, \sigma^2\mathbf{I})$. PPCA is one special case of the *Factor Analysis* models, where the distributions

are generalized to the forms $\mathbf{Z} \sim \mathcal{N}(\mathbf{Z}|0, \mathbf{P})$ and $\epsilon \sim \mathcal{N}(\epsilon|0, \mathbf{Q})$. PCA, in its original or extended forms, has been successfully used to model face appearance (100) and human motion (90).

Other linear techniques include the *Independent Component Analysis (ICA)* (42). Similar to PCA, tt also assumes the data is generated from a linear projection of the latent components (ie., $\mathbf{X} = \mathbf{WZ}$). However, instead of maximizing the variance of the projected data, it attempts to find components which are statistically independent from each other. ICA has been applied to the source separation problem, from which it is motivated, and other computer vision recognition tasks such as face recognition.

Going beyond linear methods, nonlinear dimensionality reduction techniques have also been explored, as they can model more complex interactions. *Kernel Principal Component Analysis (KPCA)* is a natural extension of PCA, where inner product matrix has been replaced by a kernel Gram matrix via the kernel trick. Graph-based techniques aim to exploit local neighborhood distances to approximate the geodesic distance in the manifold, without the explicit definition of a density distribution or a mapping function from latent to data space. *Locally Linear Embedding (LLE)* (83) assumes a smooth manifold is locally linear if there are sufficient number of samples; consequently, each data point can be constructed from its neighbors

$$\mathbf{X_i} \approx \sum_{j \in N(i)} d_{ij} \mathbf{X_j}, \tag{2.2}$$

where $N(i)$ is the neighborhood of $\mathbf{X_i}$ and d_{ij}s are constant coefficients. The coefficients d_{ij} can be obtained by minimizing the total reconstruction error. The same local linearity is then assumed for the latent coordinates

$$\mathbf{Z_i} \approx \sum_{j \in N(i)} d_{ij} \mathbf{Z_j}, \tag{2.3}$$

where d_{ij}s are the same coefficients obtained above. The latent representation $\mathbf{Z}$ can then be solved for via a minimization of the total reconstruction error. The entire minimization can be done in closed form.

Other graph-based algorithms include the *Laplacian Eigenmap* (74) and *Isomap* (97). The Laplacian Eigenmap algorithm constructs an adjacency matrix $\mathbf{A}$, treating each data point as a node in the graph and the edge weights are defined using a dissimilarity metric. The Laplacian of the graph is then defined as $\mathbf{L} = \mathbf{A} - \mathbf{D}$, where $\mathbf{D}$ is the degree matrix derived from $\mathbf{A}$. It then solves the generalized eigenvalue problem

$$\mathbf{L}v = \lambda \mathbf{D}v, \tag{2.4}$$

and the top eigenvectors are used as the latent coordinates. In the Isomap model, the dissimilarity metric between pairs of data points is the shortest-path distance (or the geodesic distance), which can be computed using the Dijkstra (20) or Floyd-Warshall (30; 107) algorithms. Both of these models employ a similar approach to spectral analysis and the kernel PCA previously mentioned; the differences lie in the construction of the kernel matrix, and whether graph geometries are taken into account.

These graph-based methods have been shown to be very effective when dealing with large datasets that are homogeneously sampled. However, they can suffer in the presence of inhomogeneous graphs that arise when dealing with noisy and sparse data. This is often the case in real world computer vision applications, unfortunately; for example, human motion datasets are comprised of a small number of motions from different subjects performing different activities. While these

databases are typically densely sampled in time, they are sparse in the motion style and activity type.

Non-linear probabilistic models usually characterize the data generation process as

$$\mathbf{X} = g(\mathbf{Z}, B) + \epsilon, \tag{2.5}$$

with ϵ having a Normal distribution, and can recover complex manifolds depending on the complexity of the function $g(\cdot)$. *Gaussian Process Latent Variable Model (GPLVM)* (53), an example of the above class of models, defines $g(\cdot)$ to be a linear combination of projection functions $g(\mathbf{Z}, B) = \sum_i b_i \varphi_i(\mathbf{Z})$. g can be marginalized out, resulting in the likelihood

$$p(\mathbf{X}|\mathbf{Z}, \beta) = \frac{1}{\sqrt{(2\pi)^{ND}|\mathbf{K}|^D}} \exp(-\frac{1}{2} tr(\mathbf{K}^{-1}\mathbf{X}\mathbf{X}^T)), \tag{2.6}$$

where $\mathbf{K}$ is a kernel matrix defined on the latent coordinates $\mathbf{K}_{ij} = k(\mathbf{Z}_i, \mathbf{Z}_j)$ and β are the parameters for the kernel function. Inference is then done using Bayes' rule and via computation of the posterior distribution $p(\mathbf{Z}|\mathbf{X}, \beta)$. Various extensions have been devised for the GPLVM models. These have received considerable attention in recent years. However, as the required optimization is non-convex and computationally expensive, they have only been applied to small databases typically composed of very few examples of a single activity. Graph-based methods and latent variable models have also been combined to capture local and global distances (103; 45). In (14), an approach based on *auto-encoders* was proposed.

2.1.2 Supervised Models

All the above techniques are unsupervised, and learn a low dimensional representation independent of the task labels. Supervised dimensionality reduction approaches try to estimate a low-dimensional representation which has sufficient information for predicting the output values. *Canonical correlation analysis (CCA)* (4) naturally extends PCA to find projections which maximally correlate with the labels. The model is originally designed to find subspaces which have maximal correlation for paired datasets, and involves solving a generalized eigenvalue problem. It has a non-linear kernelized extension, similar to PCA.

Given categorical labels, *linear Discriminant Analysis (LDA)* (29) is a popular discriminative algorithm which maximizes the separation between the different classes. In the case of two classes, the separation is defined to be the ratio of variance between classes to the variance within classes. In the general case of multiclass labels, the separation in the direction $\mathbf{w}$ is given by

$$S = \frac{\mathbf{w}^T \Sigma_b \mathbf{w}}{\mathbf{w}^T \Sigma \mathbf{w}}, \tag{2.7}$$

where $\Sigma_b = \frac{1}{C}\sum_{i=1}^{C}(\mu_i - \bar{\mu})(\mu_i - \bar{\mu})^T$ is the variance between classes, Σ is the usual variance within classes (assumed to be the same for each class), μ_i is the mean for class i, and $\bar{\mu}$ is the overall mean. The directions which maximize separation are the eigenvectors of $\Sigma^{-1}\Sigma_b$, corresponding to the lower dimensional subspace. The kernel trick can again be employed to extend the linear method to the non-linear *Generalized Discriminant Analysis (GDA)* (65).

Motivated by LDA, the *Discriminative GPLVM (DGPLVM)* (102) imposes a discriminative prior over the latent positions

$$p(\mathbf{Z}) \propto \exp(-\frac{1}{\sigma^2} tr(\Sigma^{-1}\Sigma_b)), \tag{2.8}$$

where Σ and Σ_b are the within-class and between-class variances defined above over the latent positions $\mathbf{Z}$, and σ^2 is a global scaling of the prior. When optimizing the posterior probability, this new prior translates to a new log likelihood with a discriminative regularization $\frac{1}{\sigma^2} tr(\Sigma^{-1}\Sigma_b)$. As a result, DGPLVM is now task-dependent and the latent positions are more informative regarding their labels.

While these models can capture the label information, most of the supervised dimensionality reduction techniques assume that the latent space and/or the data are generated from some restricted distribution. In the case of LDA, the conditional probability distribution for each label is assumed to be a normal distribution; in the case of the GPLVM, the generating prior for the data is assumed to be a Gaussian process. When the data do not follow the assumed distribution, the bias introduced by the assumption can significantly affect performance.

Sufficient dimension reduction (SDR) techniques (57) aim to find a low-dimensional space such that vectors in its orthogonal complement become conditionally independent of the output values. Fukumizu et al. (32) proposed the *kernel dimension reduction* (KDR) framework which, unlike other SDR techniques, does not make strong assumptions on the distribution of the input data. This is important in practice, as data in computer vision applications rarely satisfy these assumptions. These models form the foundation of our approach to sequence classification, and we will describe these models in more detail in the next chapter. Note that, to date, SDR has only been applied to static data, and has not been applied to common vision problems beyond learning a simple image manifold (73).

2.1.3 Dynamics

Temporal information is a slightly different type of label than the usual categorical assignments. They can be naively treated as another feature dimension; however, special care should be taken to ensure the information is utilized effectively. Many computer vision applications involve time-series data, as in the case of gesture classification and dynamic textures. In such settings, one would like to make use of the temporal correlation among the samples.

Hidden Markov Models (HMMs) and *Conditional Random Fields (CRFs)* (52) are popular techniques often used to model sequence data. In its simplest form, HMM imposes the Markov property on the discrete latent states — distribution of the latent state at time t only depends on the state at time $t-1$, as described by the transition probabilities. The observations are generated from the latent states via the emission probabilities. Inference for the model involves computing the distribution of the latent space, given a series of observations. The actual inference can be carried out with the forward-backward algorithm The discrete latent space can be generalized to the continuous domain by changing the Markov process to a *linear dynamical system (LDS)*. In applications such as tracking and localization where the system dynamics are simple and known, exact inference is tractable and the *Kalman filter* (44) can be used. Similar to the HMM, CRF imposes the Markov property on the latent states with respect to a graph; the conditional probability of a latent node is only dependent on its neighbors. In the case of sequence modeling, the graph of interest is a linear chain, and exact inference can be carried out as for HMM. (78) introduced the use of hidden states on a conditional random field to better model interactions, and (69) further extended this model to deal with unsegmented sequences. Unfortunately, these models require large amounts of training examples, especially when the data is high-dimensional.

Wang et al. (105) proposed the *Gaussian process dynamical model (GPDM)* and extended the

GPLVM to model dynamics. In addition to the mapping $g(\mathbf{Z})$ from the latent subspace to the observation space, another non-linear mapping is defined

$$\mathbf{Z_t} = f(\mathbf{Z_{t-1}}, A) + \epsilon' = \sum_i a_i \chi_i(\mathbf{Z_{t-1}}) + \epsilon. \tag{2.9}$$

A similar marginalization of the mapping parameters is carried out, resulting in two Gaussian processes — one modeling the dynamics and the other modeling data reconstruction. However, the learned latent space is sub-optimal as their technique is unsupervised and does not exploit relevant information for the final task.

We now turn to the image segmentation problem, with shape analysis being worthy of its own section. A brief survey of the existing methods are presented here.

2.2 Segmentation

Image segmentation refers to the process of partitioning an image into a number of segments, where the pixels within each segment exhibit similar visual characteristics. Here we will give a short summary of existing methods which do not explicitly model shape; the next section will be devoted for the broad topic of shape analysis.

2.2.1 Markov Random Field

In the past few years, approaches based on *Markov random fields (MRF)* have been popular for segmentation, as demonstrated by a large body of work (49; 47; 51). In these approaches, the image is modeled as an undirected graphical model, with the nodes being pixels and/or superpixels. Markov properties are imposed on the nodes, and the node potentials are defined in terms of the local evidence, and edge potentials are defined to encourage smoothness for neighboring pixels/superpixels with the same label.

As an example, given an image $\mathbf{D}$ and denoting binary labels by m_x for each pixel x, the *OBJ-CUT* (49) potential function to be minimized is defined as

$$\Psi(\mathbf{m}) = \sum_x (\phi(\mathbf{D}|m_x) + \sum_{y \in N(x)} (\phi(\mathbf{D}|m_x, m_y) + \psi(m_x, m_y))). \tag{2.10}$$

The emission distribution $\phi(\mathbf{D}|m_x)$ is defined as the RGB distribution of foreground pixels for $m_x = 1$, and the RGB distribution of background pixels for $m_x = 0$. This represents the global evidence, while the local evidence is expressed in the function $\phi(\mathbf{D}|m_x, m_y)$. This is set to 0 for $m_x = m_y$, and set to measure the RGB disparity and spatial distance between pixels x and y. Thus, for pixels that are labeled differently (ie., those that are separated by the segmentation boundary), closer RGB values and spatial distance will push the labels to be the same. The last part of the potential, $\psi(m_x, m_y)$ is the label prior using the Ising model

$$\psi(m_x, m_y) = \begin{cases} P & \text{if } m_x \neq m_y, \\ 0 & \text{if } m_x = m_y. \end{cases} \tag{2.11}$$

This particular prior will encourage neighboring labels to be the same, promoting smoothness across segments. Other potentials based on other information such as shape prior or layered structures can be arbitrarily added to the MRF energy function.

Several approaches to inference have been proposed for the MRF, such as graph cuts (11) and belief propagation (23). When the potentials are submodular (defined as $\forall X, Y, \psi(X \cup Y) + \psi(X \cap Y) \leq \psi(X) + \psi(Y)$), these MRFs can be solved to optimality using graph-cuts. We refer the reader to (11) for a review on graph-cuts. When the potentials are not sub-modular, approaches based on local moves, such as α moves (12) or fusion moves (55), are typically employed. Higher order cliques have also been introduced within this framework to incorporate longer range relationships. An alternative approach to do inference in graphical models is to use message passing algorithms, e.g., belief propagation (23). However, the computational and memory requirements of these approaches limits their applicability to large images. Several parallel implementations have been proposed to overcome these issues (59).

One particular form of MRF that directly defines a discriminative distribution of the latent states is the *Conditional Random Field (CRF)*, an example being the *TextonBoost* algorithm (87). We have previously described the CRF for sequence recognition using a linear chain graph. In the case of image segmentation, a regular grid graph can be used. Similar to the MRF potential shown above, *TextonBoost* also has unary potentials for color and location, as well as a pair-wise potential for edges based on color contrast. It, however, has an additional unary potential derived from texton-layout filters which are optimally combined via boosting to form a powerful classifier.

Inference is made easier since the conditional probabilities of the latent labels given the observations are modeled directly. Another way that supervision is used is through the fusing of contextual information (51). Context is added in the form of global constraints which usually specify class-co-occurrence and/or conditional dependence in the form of clique structure.

2.2.2 gPb Contour Detector

An effective contour or boundary detector can provide important information about the likelihood of a pixel's class assignment and can improve the performance of segmentation algorithms. A boundary is defined as a contour in the image plane which encompasses a change in pixel assignment from one object to another, whereas an edge (in traditional edge detection algorithms) represents an abrupt change in a low-level image feature such as color or brightness. The *global probability of boundary (gPb)* method (60) is arguably the current state-of-the-art contour detector, and we will describe it in detail.

The precursor to the gPb detector is the local *probability of boundary (Pb)* detector (62). Local cues from different image features such as rightness, color and texture gradients are considered, and these cues each estimate the posterior probability of a boundary at each pixel (centered at (x, y)). This b is accomplished by computing the difference in the feature channel on two halves of a disc of radius σ and separated by the diameter oriented at angle θ. The angle θ is sampled at 8 orientations. To accommodate for fine as well as coarse patterns, each of the three local cues are computed at three different scales σ on a logarithmic scale. These cues, denoted by $G_i(x, y, \theta)$, are linearly combined to form a multiscale oriented signal

$$mPb(x, y, \theta) = \sum_{i=1}^{9} \alpha_i G_i(x, y, \theta). \tag{2.12}$$

To introduce global information, an affinity matrix W is constructed using the *intervening contour* cue, which is the maximal mPb value along the line connecting a pair of pixels. Long-range interactions are captured in the process. Normalized Cuts, or equivalently spectral clustering, is performed on the affinity matrix by solving the generalized eigenvalue problem $(D-W)v = \lambda Dv$, where D is the corresponding degree matrix. The top k eigenvectors $\{v_j\}$ are treated as images and convolved with Gaussian directional derivatives, forming the oriented contour signals $sPb_{v_j}(x, y, \theta)$. These are again linearly combined to form the spectral signal

$$sPb(x, y, \theta) = \sum_{j=1}^{k} \frac{1}{\sqrt{\lambda_j}} sPb_{v_j}(x, y, \theta), \tag{2.13}$$

where λ_j is the corresponding eigenvalue to eigenvector v_j. The final gPb detector is a weighted sum of the local cues and the spectral signal

$$gPb(x, y, \theta) = \sum_{i=1}^{9} \beta_i G_i(x, y, \theta) + \gamma sPb(x, y, \theta). \tag{2.14}$$

The weights are learned via gradient descent on the F measure. The authors of (60) provide a detector trained on the Berkeley Segmentation dataset (61).

Arbelaez et al (3) apply the *oriented watershed transform (OWT)* of the gPb response to form regions, and subsequently constructs the *ultrametric contour map (UCM)*, defining a hierarchical segmentation by greedily merging regions. Segmentations can be obtained by thresholding the UCM at a particular level. As the threshold level is not known a priori, we use the UCM thresholded at a series of levels as baseline algorithms to the segmentation problem in chapter 5.

2.2.3 Two-step Ranking Model

The aforementioned methods all generate segmentations, and a subsequent ranking process can further increase the performance of the segmentation model. The winner of the 2009 PASCAL VOC Segmentation Challenge (22) uses such a two-step process — the *Constrained Parametric Min Cuts (CPMC)* algorithm (13) first generates region proposals by minimizing a graph-based potential function, similar to the MRF models. Given the foreground and background seed pixels, $\mathcal{V}_f$ and $\mathcal{V}_b$ respectively, the energy function is defined as

$$E(\mathbf{X}) = \sum_{u \in \mathcal{V}} D_\lambda(x_u) + \sum_{(u,v) \in \mathcal{E}} V_{uv}(x_u, x_v), \tag{2.15}$$

where the unary potential D_λ is defined as

$$D_\lambda(x_u) = \begin{cases} 0, & \text{if } x_u = 1,\ u \notin \mathcal{V}_b \\ \infty, & \text{if } x_u = 1,\ u \in \mathcal{V}_b \\ \infty, & \text{if } x_u = 0,\ u \in \mathcal{V}_f \\ f(x_u) + \lambda, & \text{if } x_u = 0,\ u \notin \mathcal{V}_f \end{cases} \tag{2.16}$$

and the pairwise potential V_{uv} as

$$V_{uv}(x_u, x_v) = \begin{cases} 0, & \text{if } x_u = x_v \\ g(u, v), & \text{if } x_u \neq x_v \end{cases} \tag{2.17}$$

λ is the foreground bias and $f(x_u)$ is chosen to measure the difference in the foreground and background RGB distributions. The similarity between adjacent pixels is defined as $g(u,v) = \exp(-\frac{\max(gPb(u),gPb(v))}{\sigma^2})$, using the gPb contour detector. To generate a wide variety of proposal segments, 30 λ values are chosen on a logarithmic scale; the foreground seed pixels $\mathbf{V}_f$ are placed on a regular grid, and the background seeds $\mathbf{V}_b$ are defined to be the image edge pixels (different variations such as the horizontal edges and vertical edges are included). The optimization problem is submodular and can be solved exactly with a parametric solver. The final pool of segments contain up to ten thousand regions. Segment-level features such as area and perimeter are then computed for each proposed region. A regression on the overlap with ground truth objects is run to determine the weight of each feature.

Endres and Hoiem (21) also proposed a similar algorithm, where, instead of using a graph-cut optimization framework, a CRF model is employed to generate segment proposals.

2.2.4 Nonparametric Bayesian Model

While very effective for certain tasks, MRFs have been shown to be inadequate for modeling the visual statistics of natural scenes (85). Recent studies show that a wide range of natural image statistics are distributed according to heavy-tailed distributions. Moreover, it is difficult to capture long-range dependencies with MRFs, as the pairwise potential is often defined only for neighboring pixels. These problems have been demonstrated not only for segmentation, but also for optical flow (denoising) (82), intrinsic images (108) and layer extraction (2). Bayesian models can explicitly model the prior distributions of natural visual statistics. In this dissertation, we build on top of the hierarchical Pitman-Yor process, which captures the power law behavior of the distribution over the number of objects per image as well as that of the size of natural segments.

Sudderth and Jordan (94) proposed an unsupervised probabilistic model for segmentation that is based on the *hierarchical Pitman-Yor process (HPY)*, which is a nonparametric Bayesian prior over infinite partitions. The HPY process is a generalization of the *hierarchical Dirichlet process (HDP)*, inducing a heavier-tailed power law prior. Confirming the findings of Sudderth et al., we show that the distribution over the size of natural segments as well as the frequencies that objects appear in an image follow a power law distribution. Thresholded Gaussian process are introduced to capture spatial coherence among regions. Moreover, this captures long-range dependencies among the observations, which are difficult to achieve with MRFs. More details about the model will be covered in chapter 4. Their approach, however, is unsupervised, and does not leverage the ever growing abundance of annotations and ground truth data, e.g., LabelMe. As a consequence, the inferred segmentations are not always accurate and have room for improvement.

Discriminative nonparametric Bayesian models have been proposed in the context of latent variable models, most notably the *Latent Dirichlet Allocation (LDA)* model devised by Blei et al (8). These approaches modify the LDA graphical model different ways; the *Supervised LDA (sLDA)* (7) adds a generative distribution of the label given the latent state, while the *Discriminative LDA (DiscLDA)* (50) uses a discriminative distribution of the latent state given the label. However, there is the question of how much discriminative power the modified likelihood can exert on the latent states. Unlike these approaches, we propose to maximize a variational lower bound on the log likelihood while requiring that the inferred label assignments coincide with the ground truth annotations. In contrast to other supervised approaches to nonparametric Bayesian models, our discriminative approach makes use of supervised data directly resulting in significant performance

improvements over the unsupervised model. Details of our model will be covered in chapter 5. In the future work section, we will develop extensions to the HPY model which are analogous to sLDA and DiscLDA.

2.3 Shape Analysis

We now turn to the modeling of shape, which is a global cue (in contrast to the low-level local cues previously discussed) and is essential for robust object recognition. Shape analysis has been a topic of interest within the computer vision community for a long time, yet it is still unsolved and remains an active research area. In this section we give a brief introduction to some of the techniques in the field.

2.3.1 Representation

The first problem encountered in shape analysis is deciding on a suitable representation of a shape instance.

Descriptor-based techniques compute features from individual shape masks and attempt to capture shape-specific characteristics of each region. The most intuitive features are the raw pixels themselves, after a global scaling to a standardized size; however, besides capturing co-occurrences of pixel locations, this representation is insufficient for characterizing most shapes. The *shape context* (5) features are proposed to solve the correspondence problem — that is, for each point on the first shape, we wish to find the best matching point on the second shape. For each point, the shape context features are computed by aggregating the locations of the remaining points in log-polar space, in radially invariant bins. The *histogram of oriented gradients (HOG)* (18) descriptors have been popular due to their robustness. Gradients are first computed by convolving the image with a derivative mask. Each pixel then contributes a weighted vote for an orientation histogram channel based on the orientation of its gradient; the votes are subsequently accumulated over local spatial regions, and normalized over rectangular or circular blocks, similar to the binning scheme for shape context features. (10) extended the work and proposed the spatial *pyramid of HOG (PHOG)*, rendering the descriptors scale invariant. Following the pyramid match kernel framework and the spatial layout theme proposed by (54), a Hog vector is computed for each grid cell at each resolution level. The final PHOG descriptor is a concatenation of all the HOG vectors.

Instead of computing densely sampled descriptors from the entire shape mask, **landmark-based techniques** aim to identify interest points which are invariant to translation, rotation and scaling. The multi-scale *Harris corner detector* (58) defines a second moment operator on the image and identifies local maxima in the scale-spatial space. Blob detection algorithms such as the *maximally stable extremal regions (MSER)* (63; 31) find correspondence points between two images of different viewpoints under affine transformation. The shapes are then represented by a bag of these interest points. Alternatively, given a set of landmark points which are consistent across the training data, the *point distribution model (PDM)* (17) further models the variation in the shapes with a Gaussian prior assumption, using PCA to pick out the main axes, or principal components.

Boundary or **contour-based techniques** also attempt to describe the shape based on the mask boundary. Splines and elliptical fourier analysis (27) have been applied to interpolate and

characterize the contour with a few coefficients. Other techniques (109) capitalize on the duality between the problems of region membership and contour identification. (76) proposes a unified CRF framework that combine the landmark-based approaches and the contour-based models.

2.3.2 Alignment

Once the shape representation has been determined, a wide variety of algorithms have been proposed to match and align one shape to another. Given two edge maps X and Y, the classic *Chamfer distance* (9) measures the average closest distance between pairs of points from the two maps:

$$d_{chamfer} = \frac{1}{|X|} \sum_i \min_j ||X_i - Y_j||. \tag{2.18}$$

The Chamfer distance has been successfully used for pedestrian detection (33) and hand tracking (91). A similar metric is the *Hausdorff distance*, where, instead averaging over all points in shape x, the maximum is taken. Another matching criterion is the *normalized cross correlation*, which uses a two-dimensional convolution between a template T and a test image I to find locations of sufficient overlap:

$$d_{corr}(u,v) = \frac{\sum_{x,y}(I(x,y) - \bar{I})(T(x-u,y-v) - \bar{T})}{\sum_{x,y}(I(x,y) - \bar{I})^2 \sum_{x,y}(T(x-u,y-v) - \bar{T})^2}, \tag{2.19}$$

where $\bar{I}$ and $\bar{T}$ are means over the image and template, respectively. These matching criteria do not take scale into consideration, and in practice either the template or the test image is rescaled over a range of scale values. Another caveat with these matching schemes is that object transformations are not explicitly modeled. In cases where the objects undergo deformations or transformations, the above criteria are not sufficient to provide the optimal matching.

The *active contour model (ACM)* (46) characterizes an object's contour as a deformable spline, or the so called *snake*, and tries to minimize an energy function based on intensity, edges, smoothness and regularity. The snakes can take scale into consideration, as Gaussian smoothing is done on the image. It is extended by the *active shape model (ASM)* (17), which replaces the spline model with the point distribution model to leverage the shape distribution within a training set. One disadvantage of the ASM is it uses only shape constraints for deforming and matching an object. This shortcoming is addressed by another extension, the *active appearance model (AAM)* (16), which considers not only shape but also texture information across the object. This model has been widely applied for analyzing faces and medical images.

The above methods model deformations as a local transformation, and is more susceptible to getting trapped in local minima. *Procrustes analysis* (38; 64) aims for a global, affine transformation and poses the problem explicitly as an optimization. Given two sets of N aligned points X and Y, the matching criterion is as follows

$$d_{proc}(X,Y) = \min_{b,T,c} \sum_{i=1}^{N} ||X_i - (bY_iT + c)||_2^2, \tag{2.20}$$

where b is the scaling parameter, T is the rotation matrix and c characterizes translation. As this optimization can be solved efficiently with singular value decomposition, it will be the foundation

on which we build our shape prior, and will be described in more detail in chapter 5. Another such techniques that model global transformation is *congealing* (41), which uses probability distributions to characterize the image pixels and models the matching problem as an affine transformation. It then aligns one image to another by minimizing entropy.

The *pictorial structure* models (28; 26) represent an object as a collection of parts arranged in a deformable configuration; they serve as a middle ground between modeling local and global transformations. The appearance of each part is modeled separately, and the deformation is characterized by interactions between pairs of parts. Specifically, a pictorial structure model is represented by an undirected graph $\mathcal{G} = (V, E)$ where the vertices correspond to parts and the edges represent the existence of a connection between parts. Under the model, an object can be expressed as a configuration $L = (l_1, \ldots, l_n)$, where each l_i specifies the location of part v_i. The matching cost for a particular configuration L is then

$$d_{pict}(L) = \sum_i m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j), \tag{2.21}$$

where $m_i(l_i)$ represents the match cost for each part and d_{ij} is the deformation cost for connected pairs of parts. In particular, these costs are defined under a statistical framework. Using Bayes' rule, the posterior distribution of a configuration L is

$$\begin{aligned} p(L|I, \theta_a, \theta_c) &\propto p(I|L, \theta_a, \theta_c)p(L|\theta_a, \theta_c) &(2.22)\\ &\propto \prod_i p(I|l_i, \theta_a) \prod_{(v_i,v_j)\in E} p(l_i, l_j|\theta_c), &(2.23)\end{aligned}$$

where I is the image, θ_a is the appearance model parameter and θ_c is the connection parameter. The parameters are estimated by minimizing the above likelihood; note that minimizing the log likelihood is equivalent to finding the optimal configuration under the criterion in Eq.2.21. Pictorial structure models have been successfully applied to tracking, pedestrian detection and human pose estimation.

Felzenszwalb et al (24) proposed the discriminatively trained *deformable part model* as an extension of the pictorial structure model. To model the appearance of each individual part, the authors employ the pyramid of HOG features at different resolutions to capture shape information while staying invariant to changes in lighting and scale. Each object is characterized by a mixture of deformable part model, and each component is comprised of a global template (or *root filter*) and a few deformable part templates (or *part filters*). The final match score of a model is then a linear combination of the scores at the global and part levels. The differentiating of (24) is that the model parameters (ie, the weights) are discriminatively trained using a latent SVM framework. Given labeled examples $\{(x_i, y_i)\}$ in the form of bounding boxes, an objective analogous to classic SVM is defined as

$$L(\beta) = \frac{1}{2}||\beta||^2 + C\sum_i \max(0, 1 - y_i \max_z\{\beta\Phi(x_i, z)\}, \tag{2.24}$$

where β is the model parameter, z is the model configuration and $\Phi(\cdot)$ is the concatenation of the HOG feature vectors at different resolutions. This proposed framework has been very successful for detection, and has become a building block for a number of detection and segmentation algorithms.

Chapter 3

Sequential Kernel Dimension Reduction

Many computer vision problems involve high dimensional datasets that are computationally challenging to analyze. In such cases it is desirable to reduce the dimensionality of the data while preserving the original information in the data distribution, allowing for more efficient learning and inference. Aside from computational reasons, it is often the case that the data lies on a lower dimensional manifold. In such circumstances, projecting the data into the manifold space allows information to be captured more robustly and the representation is less susceptible to noise. Subsequent learning done in the projected space usually outperforms modeling the data in their original feature space.

With the above benefits, dimensionality reduction techniques are often employed as preprocessing steps for raw data. Since these techniques are agnostic with respect to the task at hand (e.g., classification), they are, understandably, mostly unsupervised in nature. Supervised methods have also been explored, as they make use of label information and bring context to the models. In this chapter, we focus on a particular framework, the *kernel dimension reduction* (KDR), as it makes no strong assumption on the distribution of the input data and, as a result, possesses advantages over other models such as LDA or GPLVM.

KDR is an instance of *sufficient dimension reduction (SDR)* techniques, whose objective is to find a low dimensional subspace where vector in its orthogonal complement are conditionally independent of the output values. To date, KDR has only been applied to static data. In our extension of KDR, we investigate how supervision can be most effectively added for classifying and recognizing visual sequences.

In this work, we extend KDR to model time-series data and design kernels that capture dynamics, periodic motions and multi-class classification. Our approach combines spatial, temporal and periodic information in a principled manner, and learns an optimal manifold without assuming any distribution of the data. In particular, we propose two ways of combining this information: multiple kernel learning and building regularizers that exploit the manifold structure of the dynamics.

We demonstrate the effectiveness of our approach on classifying human gestures and activities from video, motion capture data and dynamic textures with large intra-category variations. Our approach is shown to be superior to unsupervised methods (i.e., PCA), to classification in the observation space using nearest neighbor (NN) and support vector machines (SVMs), structured prediction using SVM-HMM (1), the original KDR (32), and sequence classification methods such as HMMs, CRFs (52) and HCRFs (106). This work is a modified version of our publication (89).

3.1 Sufficient Dimension Reduction

In this section we briefly review the Sufficient Dimension Reduction paradigm (57). Let $\mathbf{x}$ be the set of measurable covariates, with $\mathbf{x} \in \Re^D$, and let $\mathbf{y}$ be the output variables. In supervised learning, the goal of *sufficient dimension reduction* (SDR) is to estimate a low-dimensional representation $\mathbf{z}$ that is sufficient for the prediction task, with

$$\mathbf{z} = \mathbf{W}\mathbf{x} \ . \tag{3.1}$$

$\mathbf{W}$ is a projection matrix to a d-dimensional space, and $\mathbf{z} \in \Re^d$, with $d \ll D$. One of the key advantages of SDR with respect to other supervised and unsupervised dimensionality reduction techniques is that it makes no assumption on the form of the distribution of $\mathbf{x}$.

The SDR criterion can be captured formally as the following conditional independence assertion

$$\mathbf{y} \perp\!\!\!\perp \mathbf{x} | \mathbf{z} \ . \tag{3.2}$$

This means that given the low-dimensional variable $\mathbf{z}$, the original features of $\mathbf{x}$ are conditionally independent of the output $\mathbf{y}$. In the statistical sense, $\mathbf{z}$ is *sufficient* for estimating $\mathbf{y}$. Intuitively, this means that the original data $\mathbf{x}$ can now be discarded and $\mathbf{z}$ has captured all the information necessary for estimating the labels $\mathbf{y}$.

Kernel Dimension Reduction (KDR) (32) maps the random variables $\mathbf{x}$ and $\mathbf{y}$ to reproducing kernel Hilbert spaces (RKHS) and characterizes conditional independence using cross-covariance operators

$$\Sigma_{yy|x} = \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy}. \tag{3.3}$$

Note that $\Sigma_{yy|x} \leq \Sigma_{yy}$ as the second term is positive semidefinite. Intuitively, conditioning on $\mathbf{x}$ reduces uncertainty (73). We note that the form of the conditional covariance operator is similar to the covariance matrix of the conditional distribution $p(\mathbf{y}|\mathbf{x})$ when $\mathbf{x}$ and $\mathbf{y}$ are jointly Gaussian.

As formalized in (32), $\Sigma_{yy|x} = \Sigma_{yy|z}$ if and only if $\mathbf{y} \perp\!\!\!\perp \mathbf{x}|\mathbf{z}$. The "magnitude" of $\Sigma_{yy|z}$ indicates the amount of conditional independence. Pursuing the SDR objective in Eq. 3.2, KDR is typically learned by minimizing the trace or the determinant of the matrix $\Sigma_{yy|z}$. In practice, the trace is usually more robust and outperforms the determinant version. The optimization problem is formulated as

$$\begin{aligned} \min \quad & tr\left[\mathbf{K}_y^c(\bar{\mathbf{K}}_z^c + \epsilon\mathbf{I})^{-1}\right] \\ \text{subject to} \quad & \mathbf{W}^T\mathbf{W} = \mathbf{I} \end{aligned} \tag{3.4}$$

where $\mathbf{I}$ is the identity matrix, $tr[]$ is the trace, $\Sigma_{yy|z}$ is defined in Eq. (3.3), and $\mathbf{K}^c$ denotes the centered kernel matrix

$$\mathbf{K}^c = (\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T)\mathbf{K}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T)^T \tag{3.5}$$

with **1** a vector of all ones.

In (32) it was shown that KDR performs well on a variety of tasks, where the training and testing data are i.i.d. samples from the joint distribution $p(\mathbf{x}, \mathbf{y})$. However, in many computer vision applications, one has to deal with time-series data, where samples are now correlated in time.

3.2 Sequence Kernel Dimension Reduction

In this section we develop a novel KDR formulation for sequence data. The idea is that we would like the latent coordinates of similar input observations that are close in space, time and, in the case of periodic motions, phase to be close in latent space.

Following (32), we rephrase the *Sequence Kernel Dimension Reduction* (S-KDR) problem of estimating the projection matrix $\mathbf{W}$ that minimizes $tr[\hat{\Sigma}_{yy|z}]$, where $\hat{\Sigma}_{yy|z}$ is the empirical estimate of $\Sigma_{yy|z}$, as

$$\begin{aligned} \min \quad & tr\left[\mathbf{K}_y^c(\bar{\mathbf{K}}_z^c + \epsilon\mathbf{I})^{-1}\right] \; + \; \lambda R(\mathbf{W}) \\ \text{subject to} \quad & \mathbf{W}^T\mathbf{W} = \mathbf{I} \end{aligned} \tag{3.6}$$

where λ is a constant, and R a regularizer.

$\mathbf{K}_y^c$ can be computed using a kernel that measures output label similarities. Here we are interested in multi-class sequence classification. We define a distance metric which is 0 for points of the same class and 1 for points of different classes. Note that this distance metric is equivalent to the Hamming distance between indicator vectors that indicate the class label. To smooth the kernel, we use an RBF kernel on top of this distance metric.

Different strategies can be used to combine the temporal, spatial and phase information. We now propose various kernels that capture this information as well as regularizers that exploit the manifold structure of the dynamics.

3.2.1 Building individual kernels

Probably the simplest way to combine the different sources of information is to build individual kernels and combine them in a fashion similar to the *multiple kernel learning (MKL)* paradigm. In the MKL setting, the goal is to learn a linear combination of kernels $K = \sum_{k=1}^{M} c_k K_k$, with an optimal set of weights. It has been successfully applied to the SVM framework. In our case, we will combine the individual kernels using a product with equal weighting

$$\bar{k}_z = k_x(\mathbf{x}_i, \mathbf{x}_j) \cdot k_t(t_i, t_j) \cdot k_p(\mathbf{z}_i, \mathbf{z}_j) \tag{3.7}$$

where $\bar{\mathbf{K}}_z = \{\bar{k}_z(\mathbf{z}_i, t_i, \mathbf{z}_j, t_j)\}$. Note that these kernels are restricted to be Mercer kernels, i.e., the resulting Gram matrix is positive semidefinite for any possible data. We now design suitable kernels for $\mathbf{K}_x$, $\mathbf{K}_t$ and $\mathbf{K}_p$.

Observations: The observation kernel should encourage latent coordinates of similar (input) observations to be similar. We use an RBF kernel such that

$$k_x(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j\|_2^2}{2\theta_x^2}\right) \tag{3.8}$$

with $\mathbf{x}_i$ as data from a single frame.

Dynamics: The dynamics kernel should encourage points that are close in time to be close in latent space. We use a bias plus an RBF kernel to model the dynamics

$$k_t(t_i, t_j) = 1 + \exp\left(-\frac{\|t_i - t_j\|_2^2}{2\theta_t^2}\right)\delta_{i,j} \tag{3.9}$$

where $\delta_{i,j} = 1$ if the i-th and j-th data points are from the same sequence, and 0 otherwise.

Phase: For periodic motions it is desirable for points with similar phase to be close in latent space. If the phase of the observations is known a priori one could build a periodic kernel by mapping the one-dimensional phase variable ϕ into a two-dimensional variable $\mathbf{u}(\phi) = (\cos(\phi), \sin(\phi))$(103) such that

$$\hat{k}_p^c(\mathbf{z}_i, \mathbf{z}_j) = 1 + \exp\left(-\frac{\sin^2(\frac{\phi_i - \phi_j}{2})}{\theta_p^2}\right)\delta_{i,j} \ . \tag{3.10}$$

This is very similar to the periodic covariance function defined in (79); however, there the input features are assumed to be periodic, while we assume the projected latent angles are periodic.

Since we only want to align in phase latent coordinates of motions that are from the same sequence, we set $\delta_{i,j} = 1$ when the i-th and j-th datapoints are from the same sequence, and 0 otherwise. While for some applications one can have a reasonable estimate of the phase, in this paper we tackle the more challenging scenario where the phase of the motion is unknown, and has to be estimated at the same time as the embedding. In particular we seek to express the phase of each point as a function of the latent coordinates. Using $\sin^2(\phi) + \cos^2(\phi) = 1$, we can express the kernel in Eq. (3.10) as a function of the cosine of the phase increment $\phi_{z_1,z_2} = \phi(\mathbf{z}_1) - \phi(\mathbf{z}_2)$. Using the fact that $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|_2 \cdot \|\mathbf{b}\|_2 \cos(\phi_{ab})$, we can finally write

$$k_p(\mathbf{z}_i, \mathbf{z}_j) = 1 + \exp\left(\frac{1}{2\theta_p^2}\frac{\tilde{\mathbf{z}}_i^T \tilde{\mathbf{z}}_j}{\sqrt{||\tilde{\mathbf{z}}_i||^2 \cdot ||\tilde{\mathbf{z}}_j||^2 + \eta}}\right)\delta_{i,j} \tag{3.11}$$

where the $\tilde{\mathbf{z}_i}$ are the centered latent coordinates computed as $\tilde{\mathbf{z}}_i = \mathbf{z} - \bar{\mathbf{z}}_{s_i}$, with $\bar{\mathbf{z}}_{s_i}$ the mean value of the latent coordinates of each sequence, and η is a regularization parameter. Note that even though $\mathbf{z} = \mathbf{W}\mathbf{x}$, we have explicitly stated the dependency on the latent space in the kernel. As shown in the experiments, even if this kernel is designed for periodic motions, it can also model non-periodic ones.

It should be noted that the form of the kernel functions having a bias allows for both multiplicative and additive effects. These correspond to the "and" and "or" relationships between the different kernels, and enable the final kernel function to capture more complex correlations within the data.

3.2.2 Dynamic Time Warping kernels

The dynamic kernels introduced above encourage points that are close in time, phase and in observation space to be close in latent space. When the dynamics of the different sequences are well structured, one can make use of more sophisticated kernels to capture this structure. The main problem is sequence alignment, which, if done well, can replace and improve upon our previous temporal and dynamics kernels. Assuming such an alignment is performed and a kernel function k_{DTW} which measures the distance of two frames in the aligned space exists, we can construct the final kernel as

$$\bar{\mathbf{k}}_z = \mathbf{k}_x(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \cdot [1 \ + \ \mathbf{k}_{DTW}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})]\delta_{y_i,y_j}, \tag{3.12}$$

where $\mathbf{k}_x$ is the previously defined observations kernel, and δ_{y_i,y_j} is defined to be 1 when $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are from the same class, and 0 otherwise. Note that $\bar{\mathbf{k}}_z$ is a matrix defined for pairs of sequences, unlike the kernel in Eq. (3.7) that was defined in terms of individual frames. However, the elements of $\bar{\mathbf{k}}_z$, $\bar{\mathbf{k}}_z(\mathbf{x}_r^{(i)}, \mathbf{x}_s^{(j)})$, are still defined on a per frame basis.

Dynamic Time Warping (DTW) solves the problem of computing distances between two sequences of different lengths, by allowing nonlinear alignment from one to the other. This distance is also known as the edit distance or Levenshtein distance in other literatures. The alignment is typically done by solving the following optimization problem

$$\begin{aligned} \min_{\psi,\theta} \quad & \textstyle\sum_{k=1}^{L} d(x_{\psi_k}^{(j)}, x_{\theta_k}^{(p)}) \\ s.t. \quad & 1 \leq \psi_1, \psi_L \leq |\mathbf{x}^{(j)}|, \psi_i \leq \psi_{i+1}, i = 1, ..., L-1 \\ & 1 \leq \theta_1, \theta_L \leq |\mathbf{x}^{(p)}|, \theta_i \leq \theta_{i+1}, i = 1, ..., L-1 \end{aligned} \tag{3.13}$$

where d is the local distance metric, typically Euclidean, $\mathbf{x}^{(j)}$ is the j^{th} sequence, L_j and L_p are the lengths of the two sequences, and $L \leq L_j + L_p$ is the number of warping frames. This problem can be solved using dynamic programming in linear time.

It should be noted that the DTW formulation can be directly interpreted as a kernel on sequences, as shown in the DTAK-SVM (86) framework. It uses the closely related problem:

$$\begin{aligned} \max_{\psi,\theta} \quad & \textstyle\sum_{k=1}^{L} k(x_{\psi_k}^{(j)}, x_{\theta_k}^{(p)}) \\ s.t. \quad & 1 \leq \psi_1, \psi_{L_j} \leq |\mathbf{x}^{(j)}|, \psi_i \leq \psi_{i+1}, i = 1, ..., L_j - 1 \\ & 1 \leq \theta_1, \theta_{L_p} \leq |\mathbf{x}^{(p)}|, \theta_i \leq \theta_{i+1}, i = 1, ..., L_p - 1, \end{aligned} \tag{3.14}$$

where $k(x_{\psi_k}^{(j)}, x_{\theta_k}^{(p)})$ is the frame-level kernel function, not a distance metric as in our setup. The authors of (86) then employ this kernel in the SVM framework for sequence classification. The primary difference between Eq. 3.13 and 3.14 is one minimizes disparity and the other maximizes similarity. In practice, we find that the original DTW formulation with Euclidean distance as metric performs better in forming alignments.

Figure 3.1. **Dynamic textures database**: Illustration of 2 instances of each of the 10 dynamic texture categories, i.e., branches, cloud, lab, flower, fountain, grass, leaves, ripple, sea_anemone, waves. Note that the variation in appearance within a single category is very large.

Once the warpings are estimated, we can compute a DTW kernel by converting distances into similarities for the those frames which are aligned, and 0 otherwise; in practice, we pass the distance metric through an inverse exponential function $f(x) = \exp^{-x}$. We smooth the results of the DTW by convolving the resulting kernel with a Laplace kernel k_{lap}, defined by $k_{lap}(i,j) = \exp^{-\frac{|i-j|}{\theta_{dtw}}}$. This will allow each frame to be correlated to a few frames surrounding its aligned frame. The final kernel is then computed according to 3.12.

3.2.3 Choice of regularizers

Additional regularizers could be employed in order to exploit the manifold structure underlying the dynamics. In this paper we propose two different regularizations: an L_2 weighted distance and a regularizer based on the Laplacian.

In order to encourage the latent coordinates of warped points to be close in latent space we employ a weighted squared loss, with weights given by the DTW kernel

$$R(\mathbf{W}) = \sum_{i,j} (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W} k_{DTW}(\mathbf{x}_i, \mathbf{x}_j) \mathbf{W}^T (\mathbf{x}_i - \mathbf{x}_j) \tag{3.15}$$

Note that $k_{DTW}(\mathbf{x}_i, \mathbf{x}_j)$ is a scalar.

A widely employed regularizer in semi-supervised learning is the Laplacian. Alternatively we can construct the regularizer

$$R(\mathbf{W}) = tr\left[\mathbf{W}^T \mathbf{X}^T (\mathbf{D} - \mathbf{K}_{DTW}) \mathbf{X} \mathbf{W}\right] \tag{3.16}$$

where $\mathbf{D}$ is a diagonal matrix with elements $D_{ii} = \sum_j k_{DTW}(\mathbf{x}_i, \mathbf{x}_j)$. As shown in our experiments, in practice the L_2 regularization outperforms the Laplacian. This suggests that the manifold structure exhibited by the DTW metric is hard to capture with a Laplacian graph.

3.2.4 Optimization

The optimization problem of Eq. (3.6) is solved via projected gradient descent, since the projection matrix $\mathbf{W}$ is constrained to be orthogonal. More specifically, the gradient is computed

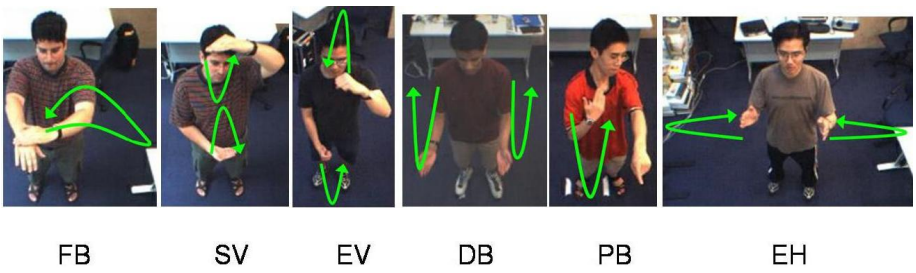


Figure 3.2. **Arm gesture database**: Illustration of the 6 gesture classes: FB - Flip Back, SV - Shrink Vertically, EV - Expand Vertically, DB - Double Back, PB - Point and Back, EH - Expand Horizontally. Each image is an abbreviation of a gesture class, where the fingertip motion trajectories are depicted in green. The direction of the arrow symbolizes the direction in which the gesture is performed.

as

$$\begin{aligned} & \frac{\partial}{\partial \mathbf{W}}(tr\left[\mathbf{K}_y^c(\bar{\mathbf{K}}_z^c+\epsilon\mathbf{I})^{-1}\right] \;+\; \lambda R(\mathbf{W})) \\ = \;& tr\left[\mathbf{K}_y^c\frac{\partial}{\partial \mathbf{W}}(\bar{\mathbf{K}}_z^c+\epsilon\mathbf{I})^{-1}\right] \;+\; \lambda\frac{\partial}{\partial \mathbf{W}}R(\mathbf{W}) \\ = \;& -tr\left[\mathbf{K}_y^c(\bar{\mathbf{K}}_z^c+\epsilon\mathbf{I})^{-1}(\frac{\partial}{\partial \mathbf{W}}(\bar{\mathbf{K}}_z^c)(\bar{\mathbf{K}}_z^c+\epsilon\mathbf{I})^{-1}\right] \;+\; \lambda\frac{\partial}{\partial \mathbf{W}}R(\mathbf{W}). \end{aligned}$$

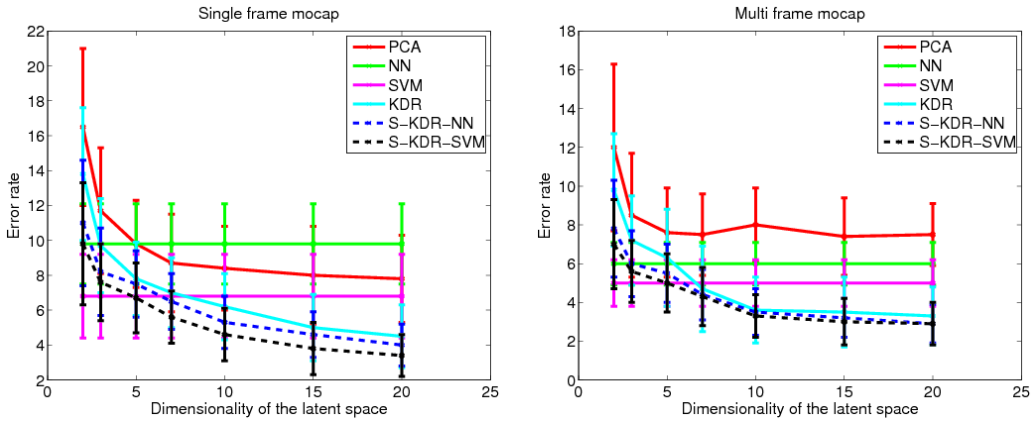
For each choice of kernel and regularizer, their partial derivatives with respect to $\mathbf{W}$ are calculated separately. We then perform a line search in the direction of the negative gradient to choose the optimal step size. To project $\mathbf{W}$ back to the space of orthogonal, we set $\mathbf{W}$ to be its own SVD decomposition.

As our objective function is nonconvex, we also employ simulated annealing on the kernel widths $\theta_x, \theta_t, \theta_p$, and θ_{dtw}. For all experiments we estimate the parameters using cross-validation. While the objective involves a matrix inversion, the computation bottleneck actually resides in the line search and gradient computation step. Leveraging the efficiency of GPUs and their abundance in computational cores, we have implemented the optimization using NVIDIA's CUDA framework with a speed up of 100x over the original Matlab implementation.

For inference, we use the estimated projection matrix $\mathbf{W}$ to compute the latent coordinates of the test points $\mathbf{x}_*$, such that $\mathbf{z}_* = \mathbf{W}\mathbf{x}_*$, and use NN and SVM as the classifiers in the low dimensional space. Note that more sophisticated classifiers could be used; however, we choose NN since it provides a reliable measure of the quality of the latent space that does not dependent on parameters that need to be tuned.

3.3 Experimental Evaluation

We demonstrate the effectiveness of our approach for classifying motion capture data, categorical dynamic textures and video sequences of human gestures.



(Single frame estimation) (Multi-frame estimation)

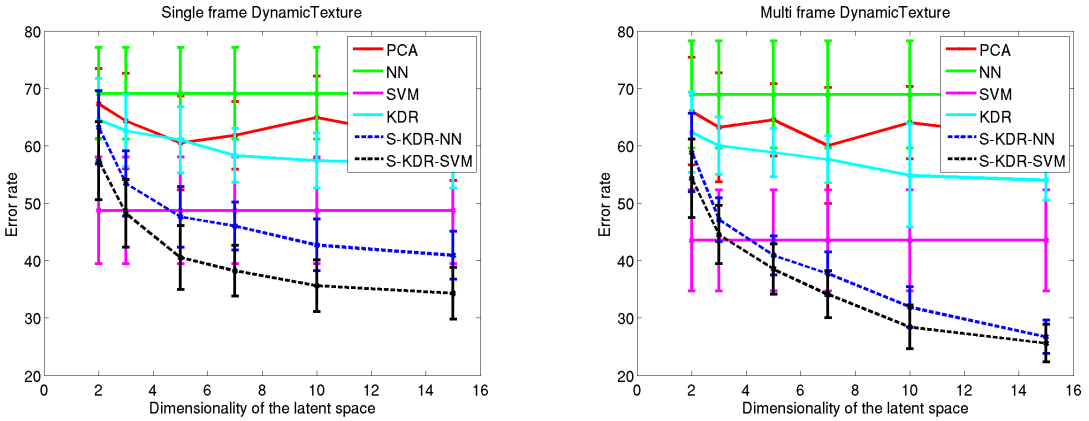
Figure 3.3. **Classification error for the Mocap database** as a function of the latent space dimensionality.

Mocap data: We use motion capture data of walking, running and jumping performed by different subjects from the CMU Mocap database (67). Each observation is a vector of 62 features, where the first 3 dimensions are the spatial velocities, and the remaining dimensions are joint angles that characterize the pose. We subsample the Mocap data by a factor of 4 so that the frame rate is 30Hz. The length of the different sequences varies from 65 to 100 frames.

Dynamic Textures: We use the DynTex database (66) to define 10 different categories of dynamic textures: branches, cloud, lab, flower, fountain, grass, leaves, ripple, sea_anemone, and waves. We scaled each image to be of dimension 25×25, resulting in 625D observations. We then perform PCA extract the top 40 eigenvectors as the feature space. For each dynamic texture, we took the first 200 frames subsampled by a factor of 4, so that each sequence has a temporal duration of 50 frames. Fig. 3.1 depicts examples of the different categories. Note that there is a large intra-class variation. Other results reported on this database are instance level recognition, where each video is segmented and divided into training and testing. In contrast, our experiment is a category recognition experiment.

Arm Gesture Dataset: We use the gesture database of (106) that is composed of six gestures: Expand Horizontally (EH), Expand Vertically (EV), Shrink Vertically (SV), Point and Back (PB), Double Back (DB) and Flip Back (FB). The users were asked to perform these gestures in front of a stereo camera. The stereo-tracking algorithm of (19) was used to estimate the head, torso, arms and forearms. Following (106), for each frame, a redundant parameterization composed of joint angles and relative coordinates of the arm joints define the 20D input observations. The gestures were performed by 13 users, and on average 90 gestures were collected per class. Fig. 3.2 illustrates the different gestures. We subsample the data by a factor of 2; the length of the different gestures varies from 14 to 42 frames.

Head Gesture Dataset: The head gesture data consists of interactions between 16 human participants and an embodied agent (106). The participants interactions were recorded, resulting



(Single frame estimation) (Multi-frame estimation)

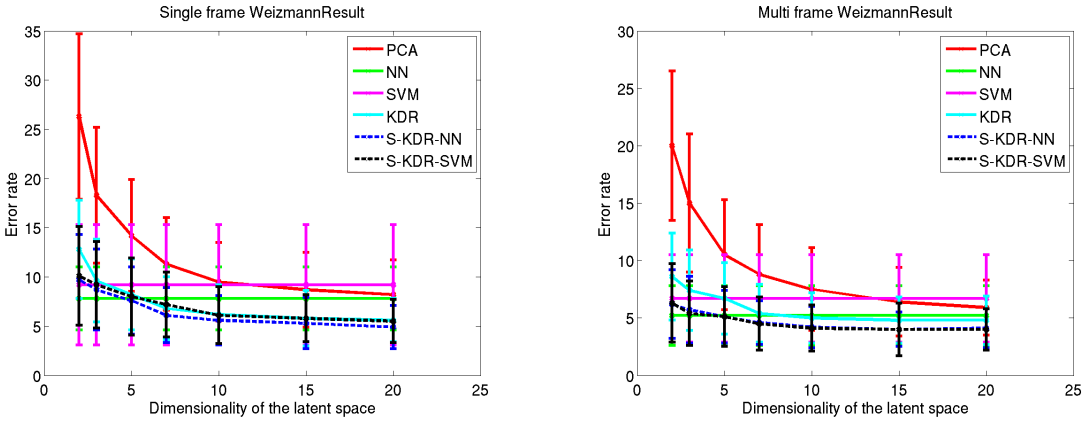
Figure 3.4. **Classification error for the Dynamic texture dataset** of (66) as a function of the latent space dimensionality.

in a total of 152 head nods, 11 head shakes and 179 miscellaneous sequences. The gestures were tracked using an adaptive view-based appearance model which captures the user appearance in different poses (70). The observations consist of the FFT of the 3D angular velocities recovered by the tracker. Each observation forms a 51D vector. We subsample the data by a factor of 2; the length of the gestures varies dramatically from 18 to 908 frames.

Activity Recognition: The activity recognition benchmark of (37) consists of 9 different subjects performing 10 different actions, including running, walking, skipping, jumping jack, waving and bending. The video sequences are low-resolution (180×144) with relatively uniform background and stationary camera. We first perform segmentation by utilizing background subtraction on the joint color and motion (i.e., optical flow) space. Based on the segmented video sequences, we normalize the bounding boxes and compute HOG features by overlaying 7×9 cell blocks with 5 histogram channels. The resulting feature space is of dimension 315. Finally, for computational reasons, we truncate the sequences at 50 frames and subsample them by a factor of 2, as this is enough to capture the dynamics.

For all databases, we compare our approach (i.e., multiple kernel learning of Eq. 3.7) to the following baselines: classification in the observation space using NN and non-linear SVMs, PCA, SVM-HMM (1), and the original KDR (32). SVM-HMM discriminatively trains a k-th order Hidden Markov Model (HMM) using the Structural Support Vector Machine formulation (SVM-Struct) (99). Given an input sequence of feature vectors, the model predicts a sequence of labels according to a linear discriminant function. SVM-HMM learns an emission weight vector for each k-th order label sequence and one transition weight vector between adjacent labels. We report results of classifying each data point independently, and combining the classifiers from the whole sequence by voting. Note that the latter assumes that the test data is segmented. The error bars in all figures represent ± 1 standard deviations. To avoid clutter, the SVM-HMM baseline is only shown in the text. The performance of SVM-HMM is consistently worse than SVM in the observation space.

Fig. 3.3 depicts classification error averaged over 5 splits as a function of the latent space



(Single frame estimation) (Multi-frame estimation)

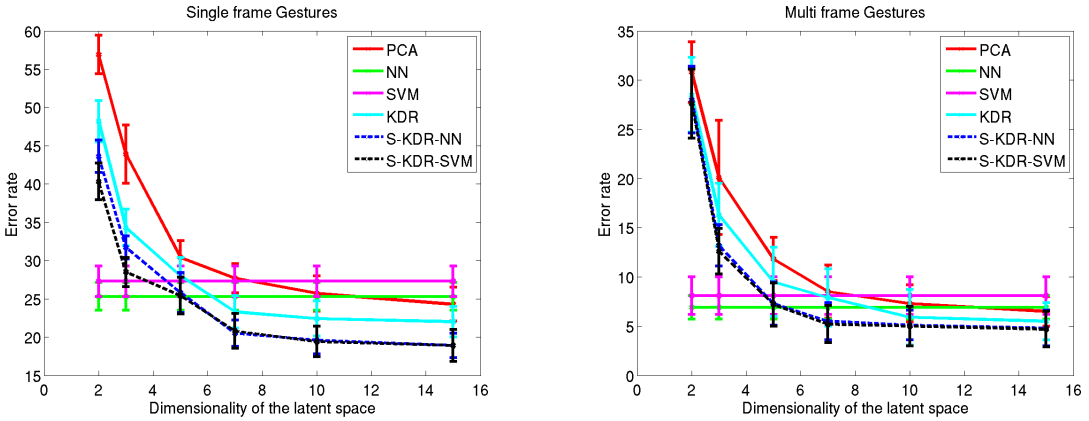
Figure 3.5. **Classification error for the Weizmann dataset** of (37) as a function of the latent dimensionality.

dimensionality for the Mocap database. For each class, 5 examples were used for training and 20 for testing. Our approach consistently outperforms all the baselines even when using very low-dimensional spaces. Note that NN and SVM in the observation space and SVM-HMM accuracies do not vary with the dimensionality since these methods do not learn a latent space. The average error of SVM-HMM was $7.4 \pm 1.2\%$ for single frame and $5.2 \pm 1.2\%$ for multi-frame. Fig. 3.3 (left) depicts the error rate when doing single frame classification, i.e., every frame is independent. Fig. 3.3 (right) shows classification error when combining the single frame classifiers by voting. As expected voting results in better performance since it combines information from all the frames in the sequence. However, the single frame estimation does not assume that the sequences are segmented. Our approach results in extremely good performance; $3.4 \pm 1.2\%$ classification error for single frame estimation, and $2.9 \pm 1.0\%$ for sequence classification.

Fig. 3.4 shows classification error averaged over 5 splits as a function of the dimensionality of the latent space for the dynamic texture database (66). 3 examples per class were used for training and 5 for testing. The correct classification rate of SVM-HMM is $36.1 \pm 5\%$ for single frame and $34.4 \pm 7.2\%$. Our approach significantly outperforms all the baselines even with low-dimensional latent spaces. Note that this is an extremely hard problem since one has to classify 10 dynamic texture categories with very few examples and very large intra-class variations, as shown in Fig. 3.1. As a result, the performance of the baselines is as low as 34%, while for our approach is 75%.

Fig. 3.5 shows classification error averaged over 5 splits for the Weizmann dataset. For each class, 4 sequences were used for training and 5 for testing. The average error rate of SVM-HMM is $9.8 \pm 5.8\%$ and $7.0 \pm 3.8\%$ for single-frame and multi-frame estimation, respectively. Note that SVM in the original space overfits, and NN works better. Our approach consistently outperforms PCA and SVM, with a $4.9 \pm 2.2\%$ error rate for single frame estimation and a $4.1 \pm 1.7\%$ error rate for sequence classification.

Fig. 3.6 shows classification error for the arm gestures dataset of (106) averaged over 5 splits. For each class 10 examples were used for training and 100 for testing. The different gestures are shown in Fig. 3.2. The performance of SVM-HMM is $67.5 \pm 3.2\%$ for single frame and $82.2 \pm 3.5\%$ for multi frame. Our approach outperforms the baselines when using single frame or multi-frame



(Single frame estimation) (Multi-frame estimation)

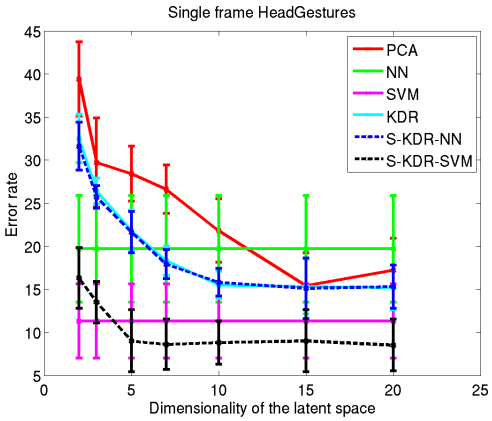
Figure 3.6. **Classification error for the Arm Gestures dataset** of (106) as a function of the latent dimensionality.

	N	Accuracy (%)
HMM $w = 0$	80	84.22
CRF $w = 0$	80	86.03
CRF $w = 1$	80	81.75
HCRF (one-vs-all) $w = 0$	80	87.49
HCRF (multi-class) $w = 0$	80	91.64
HCRF (multi-class) $w = 1$	80	93.85
S-KDR-SVM $w = 0$	10	**95.3**

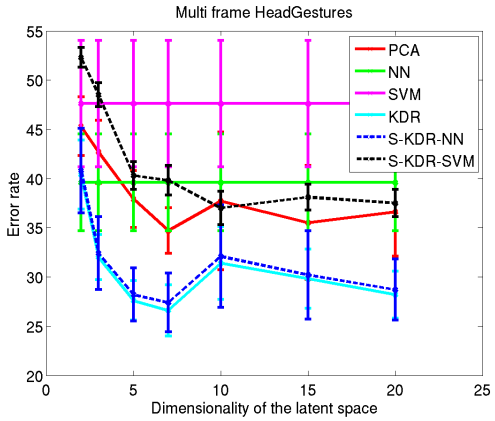
Table 3.1. Comparison of our approach (S-KDR)) HMM, CRF and HCRF on the arm gesture dataset, as reported in (106).

(voting) classification, resulting in $81.1 \pm 1.6\%$ for single frame and $95.3 \pm 1.8\%$ for multi-frame. As shown in Table 3.1, our approach also results in better performance than HMMs, CRFs and HCRFs (106). Moreover, we only require 10 training examples per class, while the baselines were trained with approximately 80 examples per class. Note also that there is a large benefit for this database when using information from multiple frames. This is because, even though the different gestures have some poses in common, the overall gesture is very discriminative.

Error rates averaged over 5 splits for the head gesture database of (106) are shown in Fig. 3.7. For each class 5 examples are used for training and 30 for testing. Note that, unlike with the other databases, incorporating information from multiple frames decreases performance. This is to be expected since head nods, head shakes and miscellaneous have very similar poses, sometimes for more than 50% of the length of the sequence. Moreover, failures in the monocular tracking make SVM approaches fail in the multi-frame setting. The mean error of the SVM-HMM is $28.8 \pm 5.7\%$ for single frame and $39.1 \pm 4.3\%$ for multi-frame. Comparisons of our approach to HMMs, CRFs and HCRFs are shown in Table 3.2. Our approach outperforms all the baselines and it is trained using only 10% of the data used to train the other baselines, resulting in $8.5 \pm 3\%$ mean error for single frame and $28.7 \pm 3.1\%$ for multi-frame.



(Single frame estimation)



(Multi-frame estimation)

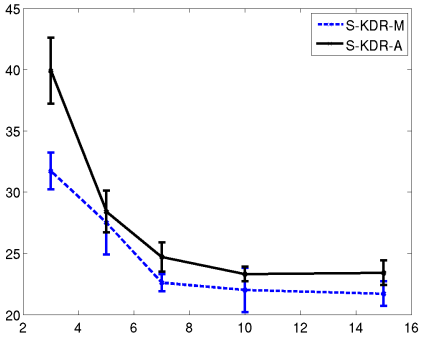
Figure 3.7. **Classification error for the Head Gestures dataset** of (106) as a function of the latent dimensionality.

	N	Accuracy (%)
HMM $w = 0$	171	65.33
CRF $w = 0$	171	66.53
CRF $w = 1$	171	68.24
HCRF (multi-class) $w = 0$	171	71.88
HCRF (multi-class) $w = 1$	171	85.25
S-KDR-SVM $w = 0$	15	**91.5**

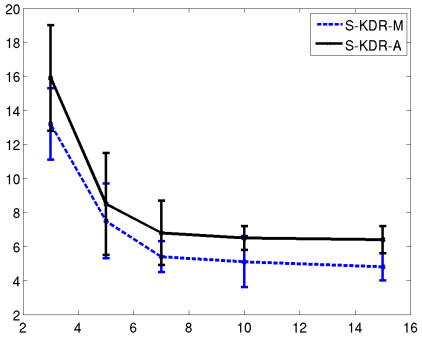
Table 3.2. Comparison of our approach (S-KDR) with HMM, CRF and HCRF on the head gesture dataset, as reported in (106).

Not only does our approach outperform the baselines, but, more importantly, the standard errors are much smaller. This implies that S-KDR consistently learns latent spaces that are good for the classification task. We also investigate other ways of combining the different sources of information. In particular, we compare the multiplicative kernel of Eq. (3.7) to an additive kernel, $\bar{\mathbf{K}}_z = \mathbf{K}_x + \mathbf{K}_t + \mathbf{K}_p$. As shown in Fig. 3.8 the multiplicative kernel outperforms the additive one.

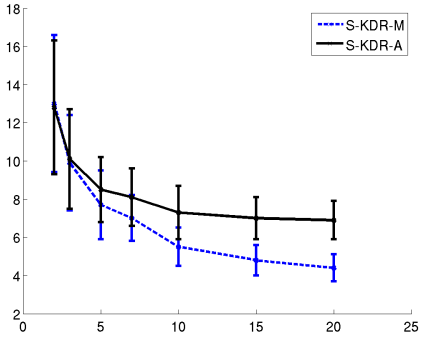
We now evaluate the effectiveness of the dynamic time warping kernels. For the Weizmann, Mocap and the Arm Gesture datasets, the dynamics are well-structured and relatively distinct, allowing for accurate computation of the time warpings. For each of these datasets, we compare KDR and S-KDR to 3 different ways of incorporating DTW: L_2 regularization (S-KDR-L2-DTW), Laplacian regularization (S-KDR-Lap-DTW) and the kernel of Eq. (3.12) (S-KDR-DTW). In the former two cases we add the regularizations to the KDR objective, with $\bar{\mathbf{k}}_z = \mathbf{k}_x$. We believe that this is a fair comparison with S-KDR, since then the dynamic information is utilized only once. In all three cases, we find that NN outperforms SVM; as a consequence we only report NN results. Furthermore, since the new latent spaces are optimal for sequence alignment, we perform an additional per-sequence classification using DTW in the latent space. As shown in Fig. 3.9 the results for all three datasets are similar: L_2 regularization typically improves the S-KDR performance while the Laplacian regularization often degrades it. The kernel combination in Eq.



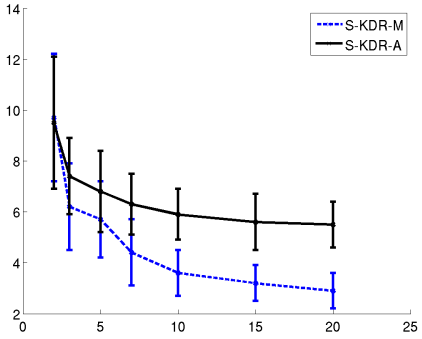
(Single frame arm gesture data)



(Multi-frame arm gesture data)



(Single frame Mocap data)



(Multi-frame Mocap data)

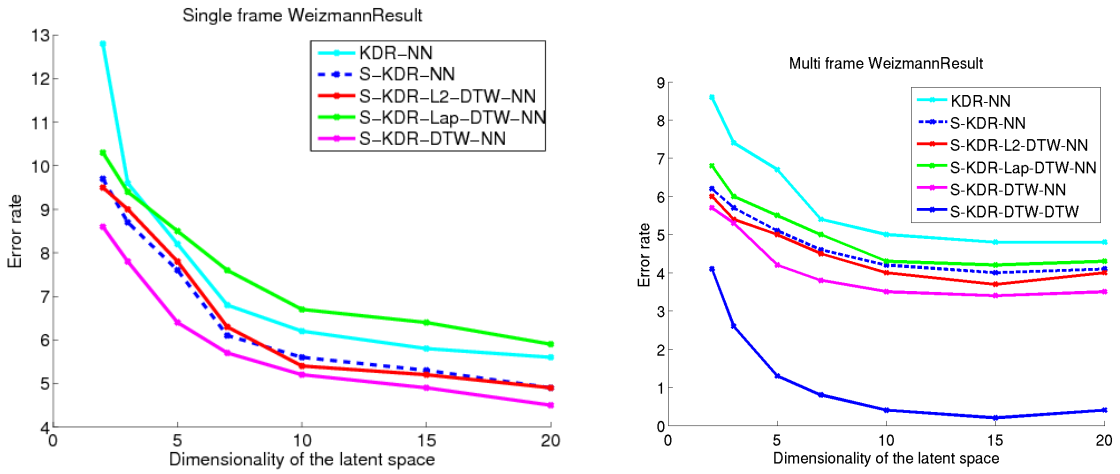
Figure 3.8. **Multiplicative vs additive kernels**, showing classification errors for single frame and multi-frame estimation

(3.12) uniformly achieves the best performance. Moreover, the DTW classifier in the latent space achieves state-of-the-art results on the Weizmann dataset (i.e., 99.8%). This suggests that when there is sufficient structure in the dynamics, the DTW kernel correctly captures both linear and nonlinear aspects of dynamics, and DTW is the optimal classifier.

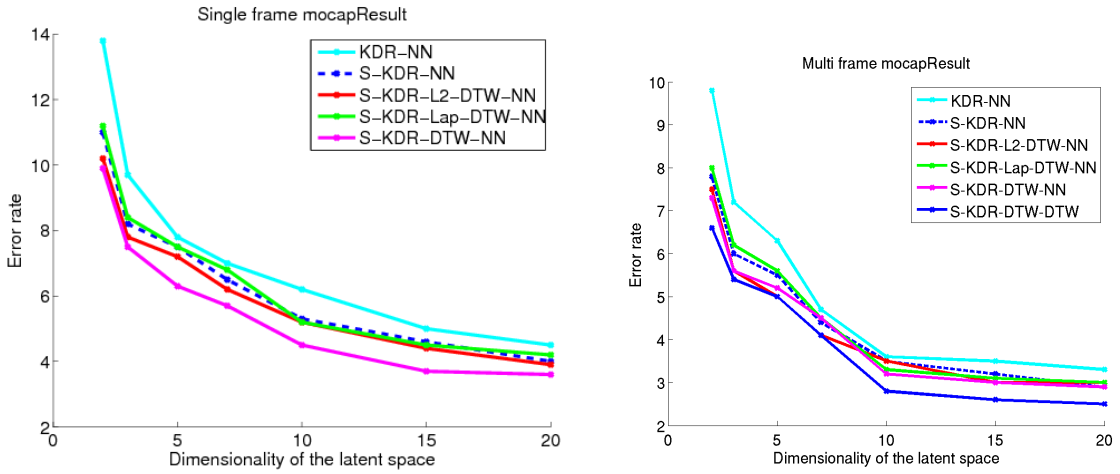
3.4 Conclusion

In this chapter, we investigated two ways of modeling time-series data in the context of dimensionality reduction. We assumed that the data has already been segmented—that is, the relevant and essential regions of the input sequences have already been extracted. For instance, the motion capture data has been concisely represented with joint angles and coordinates; for the activity recognition dataset, the human subject is posed in front of a homogeneous background and segmentation is trivial via background subtraction.

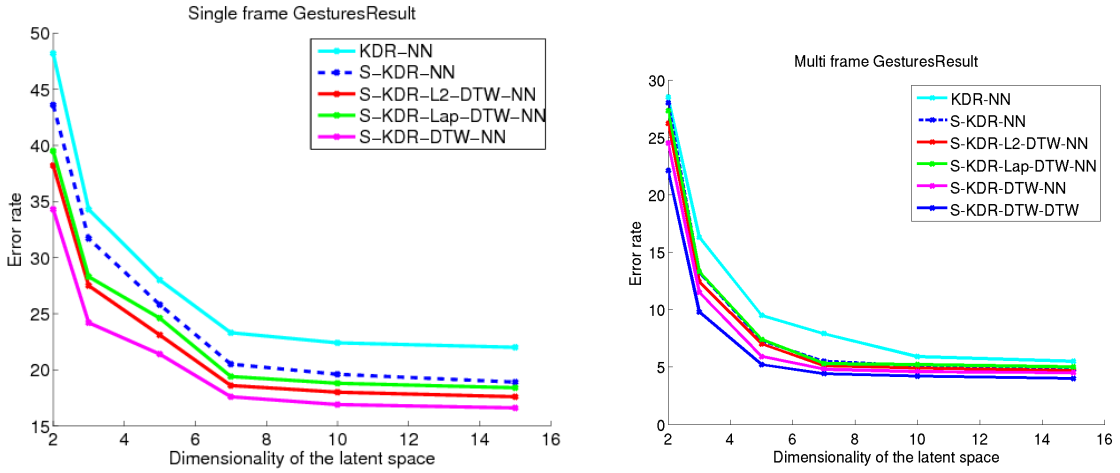
In the remaining chapters, we switch from the recognition problem to the more difficult segmentation problem. As the task is difficult enough for individual frames let alone video sequences, we focus on the image segmentation task.



(Weizmann)



(Mocap)



(Arm Gesture)

Figure 3.9. **Classification error for the Weizmann (37), Mocap (67) and Arm Gesture (106) datasets.** Comparison of different methods of incorporating dynamics structure with the DTW kernel. Using the DTW kernel consistently achieves the best performance.

Chapter 4

A Review of Hierarchical Pitman-Yor Processes

Image segmentation is the process of partitioning an image into homogeneous regions or *segments* often resembling objects, with the goal of simplifying the image and describing it with a meaningful representation. A segmentation algorithm takes as input an image in its pixel forms, and outputs a set of segments. More precisely, the algorithm assigns a label to each pixel using its visual characteristics. It can be approached as a clustering problem, where each segment is a cluster of pixels. K-means and normalized cuts are common techniques that take this approach; however, they suffer from

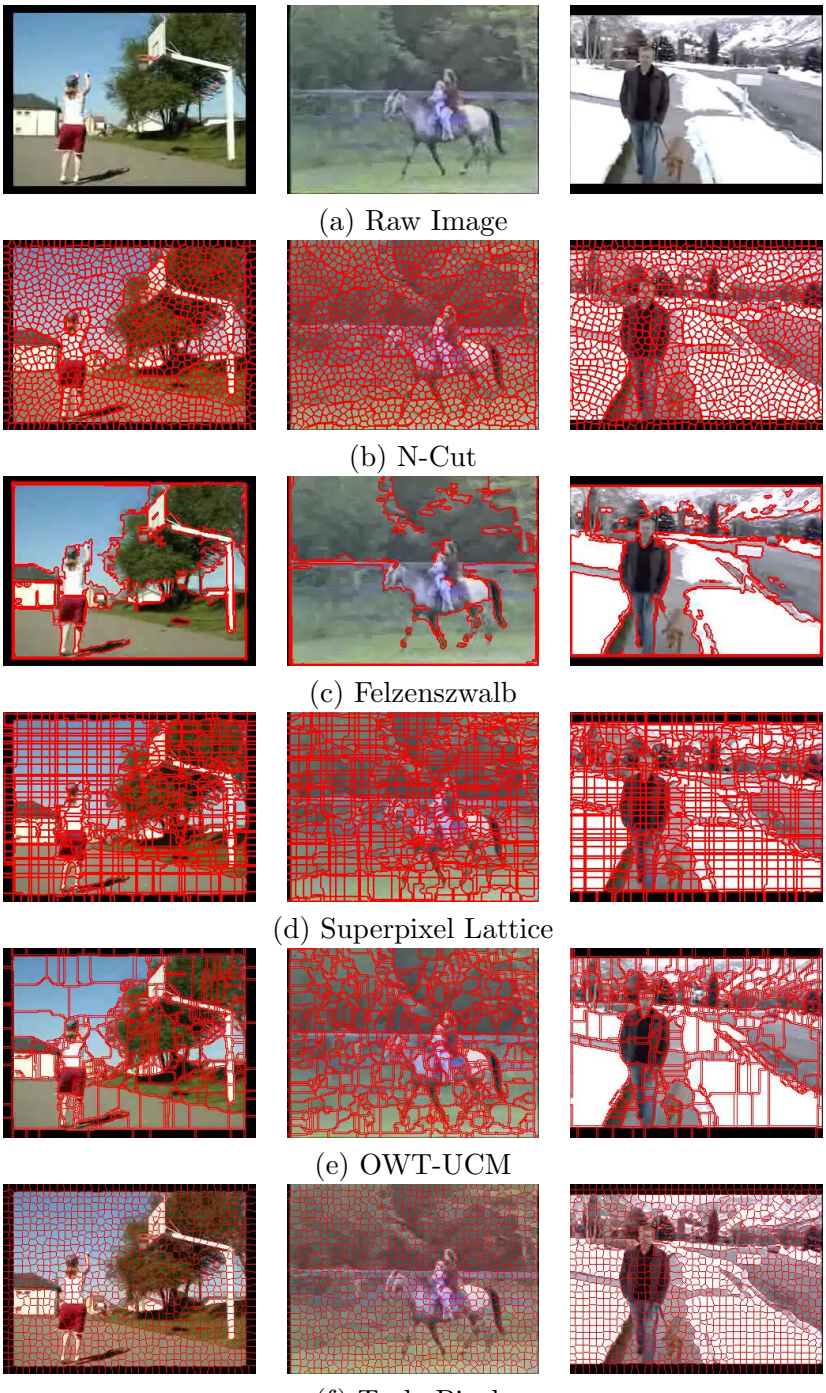
- The need for a predetermined number of clusters, and
- A lack control over the empirical distribution of the clustering statistics such as cluster sizes.

. To address these two shortcomings, we consider the class of nonparametric Bayesian models, which can adaptively adjust the number of clusters according to the data and explicitly model the prior distribution of cluster sizes. In particular, we will use the generative segmentation model proposed by Sudderth and Jordan in (94).

In this chapter, we give a brief overview of Sudderth's model. As the task of modeling individual pixels is quite computationally costly, we will first describe the preprocessing step of clustering pixels into superpixels.

4.1 Building blocks: Superpixels

We begin by taking a bottom-up approach and preprocess each image into a set of *superpixels*. Superpixels are groups of pixels that are perceptually consistent, and are natural representations



(a) Raw Image

(b) N-Cut

(c) Felzenszwalb

(d) Superpixel Lattice

(e) OWT-UCM

(f) TurboPixels

Figure 4.1. Comparisons of superpixel algorithms on sample images of the categories basketball, horseback riding and dog walking.

of an image. Most importantly, superpixelation improves the computational efficiency of any algorithm, as it reduces hundreds of thousands of pixels to at most a few thousand superpixels.

However, by using only low-level cues to construct locally consistent regions, it is possible that object boundaries are destroyed in the preprocessing process.

There is ample literature on how to compute superpixels. This sections compares various different methods, including graph-based techniques (25; 68; 81; 71) and edge/contour-based algorithms ((56; 3)). Normalized cut (N-Cut) (81; 71) constructs a graph where the nodes are pixels and the edge weights depend the maximal crossing edge, and performs spectral clustering on the graph. Felzenszwalb (25) constructs a similar but sparser graph, and performs iterative merging with internal difference as a criterion. Superpixel lattice (68), likewise, builds a similar graph, but iteratively bisect the image by respecting edges and via a dynamic programming scheme. The oriented watershed transform and ultrametric contour map (OWT-UCM) starts with an edge detector and forms closed, non-self-intersecting contours via the Ultrametric Contour Map; the algorithm then hierarchically construct regions by using a watershed approach. TurboPixels (56) randomly generates a seed set of superpixel centers, and repeatedly grow each cluster with speed depending on a combination of geometric flows.

Most of the above methods make use of an edge detector, and the state-of-the-art detector of choice is the gPb detector (60). GPU implementation of the algorithm is available (15), achieving significant computation speedup.

In Fig. 4.1, we present sample superpixel outputs of the above algorithms. From qualitative and visual comparisons, as well as considering computational requirements, we settle on TurboPixel as our method of choice for computing superpixels.

4.2 A review of hierarchical Pitman-Yor processes

We now give a brief overview of the hierarchical Pitman-Yor (HPY) model for visual scenes (94), which is a generalization of the hierarchical Dirichlet process (HDP) (96). For a detailed treatment on the subject, the readers are referred to Erik Sudderth's thesis (93).

Recall that the Dirichlet process is a measure on probability measures, and consists of a scaling parameter α and a base measure $\mathbf{G_0}$. There are multiple ways to specify the process, but we will focus on the stick-breaking construction. If a measure $\mathbf{G}$ is drawn from a Dirichlet process $\mathbf{G} \sim DP(\alpha, \mathbf{G_0})$, then

$$\begin{aligned}
\mathbf{G} &= \sum_{k=1}^{\infty} \pi_k \delta_{\phi_k} \text{ almost surely, where} \\
\phi_k &\sim \mathbf{G_0}, \\
\pi_k &= \omega_k \prod_{i=1}^{k-1} (1-\omega_i), \text{ and} \\
\omega_k|\alpha &\sim Beta(1, \alpha).
\end{aligned}$$

Note that the marginal distribution of π is an exponentially decaying probability distribution. We denote such a distribution by $\pi \sim GEM(\alpha)$, which places a prior distribution over the sizes of the sticks (or equivalently, the cluster sizes).

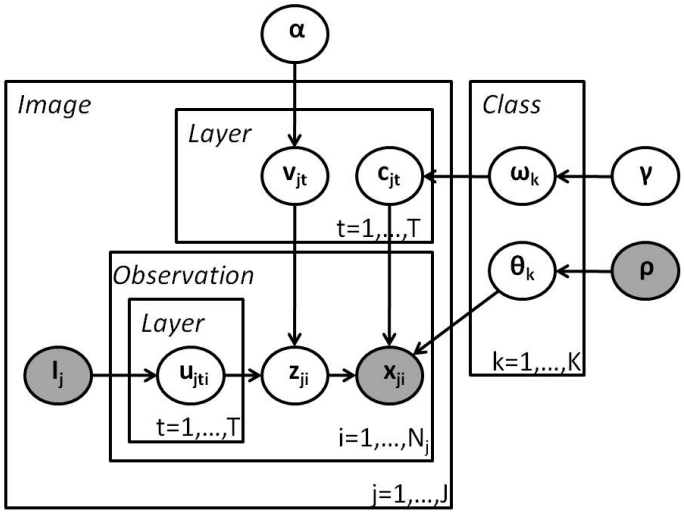


Figure 4.2. Graphical model of the hierarchical Pitman-Yor Process (HPY), applied to natural scene segmentation (94). Each observation x_{ji} is assigned to layer z_{ji}, an indicator variable. The assignment depends on the thresholded GPs $\mathbf{u}_{jt}$ and layer probabilities v_{jt}, which are generated from the PY stick-breaking prior $GEM(\alpha_a, \alpha_b)$. Each layer is assigned to class c_{jt}, another indicator variable, which follow the PY prior $GEM(\gamma_a, \gamma_b)$ where w_k are the stick lengths. Each class has an associated appearance model θ_k.

The Pitman-Yor process (77), denoted by $\pi \sim GEM(\gamma_a, \gamma_b)$, is an extension of the previously formulated Dirichlet process. Similarly, it can be defined using the stick-breaking construction as

$$\begin{aligned} \pi_k &= w_k \prod_{l=1}^{k-1}(1-w_l) = w_k(1-\sum_{l=1}^{k-1}\phi_l), \text{ with} \\ w_k &\sim Beta(1-\gamma_a, \gamma_b + k\gamma_a), \end{aligned} \tag{4.1}$$

where γ_a, γ_b are hyperparameters satisfying $0 \leq \gamma_a < 1$ and $\gamma_b > -\gamma_a$.

The π's are the partition probabilities, while the w_k's are the stick lengths. Note that we recover a Dirichlet process, specified by a single concentration parameter γ_b, when $\gamma_a = 0$. When $\gamma_a > 0$, the stick lengths progressively increase in their expected values and the partition probabilities follow a power-law distribution. This is important as many natural visual statistics follow heavy-tailed distributions, and the PY prior is ideal for modeling these. While the PY process is a prior on infinite partitions, only a finite subset of partitions will have positive probabilities greater than a threshold ϵ. Hence, the PY process implicitly imposes a prior on the number of partitions.

In the HPY model for visual scenes (94), Pitman-Yor process priors are placed over the distributions of global class categories and segment sizes. Class assignments c_{jt} are sampled from $\phi \sim GEM(\gamma_a, \gamma_b)$, which is the stick-breaking prior described above, with w_k being the stick length. Similarly, the layer assignment probabilities v_{jt} are sampled from $\pi \sim GEM(\alpha_a, \alpha_b)$. Fig. 4.2 shows the directed graphical model. Using techniques discussed in the previous section, each image is segmented into superpixels, which are from now on treated as the observed data units x_{ji}. Each data point x_{ji} is then assigned to a cluster, or *layer*, with probability

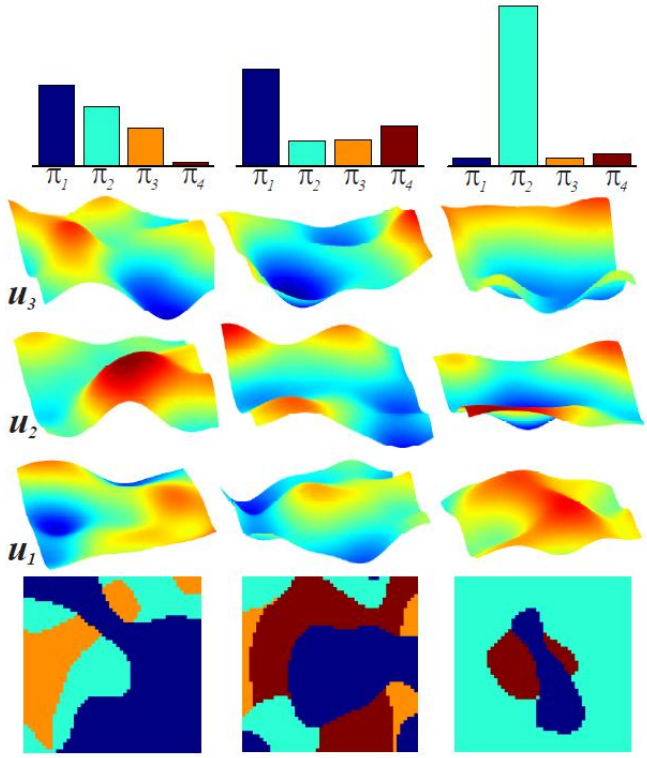


Figure 4.3. Examples of Gaussian Processes u_t for each layer t, thresholds for each layer π_t and the final layer assignment according to the assignment rule 4.2. (Figure borrowed from (94))

$$P[z_{ji} = t | z_{ji} \neq t-1, \ldots, 1] = P[u_{jti} < \Phi^{-1}(v_{jt})] = v_{jt},$$

where Φ is the CDF of the Gaussian distribution and we have introduced a zero mean Gaussian process (GP) $\mathbf{u}_{jt}$ for each layer t. This construction ensures that the conditional layer assignment probabilities are the stick lengths in the Pitman-Yor process (w_k of Eq. 4.1), and the marginal layer assignment probabilities (equivalently, layer sizes) follow a power law distribution, as desired. Moreover, by formulating the GPs, spatial dependencies can be introduced through their covariance functions (covered in the next section). These thresholded GPs completely determine the layer assignment of each superpixel, with the assignment rule being

$$z_{ji} = \min\{t | u_{jti} < \Phi^{-1}(v_{jt})\}. \tag{4.2}$$

Examples of GPs and their thresholds, along with the final layer assignments are shown in Fig. 4.3.

To facilitate sharing of layer parameters, each layer is associated with a global object class c_{jt} with an appearance model parameter θ_k, representing means of the appearance distribution belonging to the class. The multinomial emission probability is then

$$p(x_{ji} | z_{ji} = t, c_{jt} = k, \theta) = Mult(x_{ji} | \theta_k). \tag{4.3}$$

Modeling Spatial Dependencies: Thus far, the observations $\mathbf{x}$ are related only via their appearance parameters. We employ a few mechanisms to capture spatial dependencies. First, the bottom-up superpixelation process over-segments each image, introducing local consistency within each superpixel. Furthermore, recall that in the HPY model, each layer is associated with a zero mean GP over $\mathbf{u}_{jt}$. Note that the above derivations do not depend on the GPs' covariance function. If the GPs have diagonal covariance functions, the model is spatially independent. More general covariances can encode affinities among pairs of observations.

In particular, for an image j, we employ a covariance that incorporates intervening contour cues based on the gP_b contour detector (3) described in chapter 2,

$$W_j(x_i, x_{i'}) = (1 - gP_b(x_i, x_{i'}))^{\sigma_{gpb}} \exp(-\frac{||\bar{x_i} - \bar{x_{i'}}||^2}{2\sigma_{sp}^2}), \tag{4.4}$$

where σ_{sp} and σ_{gpb} are constants, $\bar{\mathbf{x}}_i$ is the centroid of the i-th superpixel and $gP_b(x_i, x_{i'})$ is the maximal gP_b response along the line between the two superpixels' centroids. This forces a high disparity between two superpixels which are divided by a sharp edge.

To induce sparsity, we also included a neighborhood parameter, ϵ_n; the covariance entry is zero if the corresponding superpixel centroids are more than ϵ_n pixels apart; otherwise, the value is the same as above. Since W_j is a covariance matrix, it is required to be positive semi-definite (PSD). To ensure that the covariance is PSD, we compute the eigen-decomposition of W_j and retain only the eigenvalues that are at least ϵ_{eig} times the maximal eigenvalue. This is done for robustness and computational reasons (the feature dimension becomes smaller).

4.3 Variational Learning

The HPY model described in the previous section has a considerably complex likelihood distribution, and a direct likelihood maximization is not plausible. Popular choices for approximate inference are *Markov chain Monte Carlo (MCMC)* methods and *variational learning*. MCMC techniques approximate a target distribution (in our case, the posterior likelihood) by constructing a Markov chain and sampling from it, and there is a large body of works describing their numerous variants. Rejection sampling methods such as *Metropolis-Hastings* (39) generates a random sample from a proposal distribution, and rejects samples based on the ratio of the proposal and target likelihoods. *Gibbs sampling* (34) is useful for sampling from a multivariate distribution, and forms the Markov chain by sampling from each variable's leave-one-out conditional distribution; different ways of selecting the variable to sample result in different versions of the sampler (e.g., the blocked Gibbs sampler and collapsed Gibbs sampler). *Slice sampling* (72) is yet another technique which first introduces and samples an auxiliary variable. *Particle filter* (36) and other *sequential Monte Carlo (SMC)* methods are based on simulation, and can model arbitrarily complex distributions with sufficient number of samples or *particles*. All the above MCMC algorithms face the problem of convergence, as the Markov chain can take a long time to converge to the stationary distribution.

Alternatively, variational inference iteratively improves the lower bound on the likelihood, which often converges quickly in practice. Independent and factorized posteriors (referred to as *variational posteriors* q) are proposed, and variational methods attempt to approximate the original full likelihood with the simplified proposed distributions.

Formally, let $\mathbf{x}$ be observed variables, $\mathbf{h}$ be hidden variables and let θ denote model parameters. Following the derivations in (6), the full log likelihood can be lower bounded with the introduction of variational posterior distribution $q(\mathbf{h}|\mathbf{x})$:

$$\begin{aligned}
\log p(\mathbf{x}|\mathbf{w}) &= \log \sum_{\mathbf{h}} p(\mathbf{x}, \mathbf{h}|\mathbf{w}) \\
&= \log \sum_{\mathbf{h}} q(\mathbf{h}|\mathbf{x}) \frac{p(\mathbf{x}, \mathbf{h}|\mathbf{w})}{q(\mathbf{h}|\mathbf{x})} \\
&\geq \sum_{\mathbf{h}} q(\mathbf{h}|\mathbf{x}) \log \frac{p(\mathbf{x}, \mathbf{h}|\mathbf{w})}{q(\mathbf{h}|\mathbf{x})} \\
&= E_q[\log p(\mathbf{x}, \mathbf{h}|\mathbf{w})] + H_q(\mathbf{h}), \qquad (4.5)
\end{aligned}$$

where we have used Jensen's inequality to bring the log function inside the summation. It is worthwhile to note that the second term of the last expression is the entropy. A further connection to the Kullback-Leibler (KL) divergence, a metric for probability distributions, can be drawn, as the difference between the full log likelihood and the approximation is

$$KL(q(\mathbf{h}|\mathbf{x})||p(\mathbf{x}, \mathbf{h}|\mathbf{w})) = -\sum_{\mathbf{h}} q(\mathbf{h}|\mathbf{x}) \log \frac{p(\mathbf{x}, \mathbf{h}|\mathbf{w})}{q(\mathbf{h}|\mathbf{x})} \geq 0. \qquad (4.6)$$

Therefore, maximizing the lower bound in Eq. 4.5 is equivalent to minimizing the KL divergence; essentially, variational methods try to reduce the approximation gap of the proposed variational posteriors.

Following (94), we train the HPY model with a mean field variational approximation. Detailed derivations can be found in the Appendix A.

A completely factorized variational posterior is introduced as follows

$$q(\mathbf{u}, \mathbf{v}, \mathbf{c}, \mathbf{w}, \theta) = \prod_{k=1}^{K} q(w_k|\omega_k) q(\theta_k|\eta_k) \times \prod_{j=1}^{J} \prod_{t=1}^{T} q(v_{jt}|\nu_{jt}) q(c_{jt}|\kappa_{jt}) \prod_{i=1}^{N_j} q(u_{jti}|\mu_{jti}),$$

where the distributions are,

$$\begin{aligned}
q(\theta_k|\eta_k) &= Dir(\eta_k) \\
q(\mathbf{c}_j|\kappa_j) &= Mult(\mathbf{c}_j|\kappa_j) \\
q(w_k|\omega_{k,a}, \omega_{k,b}) &= Beta(w_k|\omega_{k,a}, \omega_{k,b}) \\
q(\bar{v}_{jt}|\nu_{jt}, \delta_{jt}) &= N(\bar{v}_{jt}|\nu_{jt}, \delta_{jt}) \\
q(u_{jti}|\mu_{jti}, \lambda_{jti}) &= N(u_{jti}|\mu_{jti}, \lambda_{jti}),
\end{aligned}$$

with $\bar{v}_{jt} = \Phi^{-1}(v_{jt})$.

For tractability, we truncate the variational posterior at T layers and K classes by setting $q(v_{jT} = 1) = 1$ and $q(w_K = 1) = 1$. We then train the model by optimizing the lower bound on the marginal likelihood

Algorithm 1: HPY
Initialize variational parameters
for N iterations
for each image j
for each layer t = T-1, ..., 1
Update GPs and thresholds $(\mu_{\mathbf{jt}}, \lambda_{\mathbf{jt}}, \nu_{jt}, \delta_{jt})$ with gradient descent
Update class assignments $\kappa_{\mathbf{j}}$ with closed form solution
Incrementally update class appearance model η
Incrementally update stick breaking distribution for classes $\omega_{,\mathbf{a}}$ and $\omega_{,\mathbf{b}}$
if (variational lower bound $\mathcal{L}$ converges within ϵ)
break
end

Table 4.1. HPY variational learning algorithm using gradient descent

$$\begin{aligned} \log p(\mathbf{x}|\alpha,\gamma,\rho) &\geq H(q) + E_q[\log p(\mathbf{x},\mathbf{z},\mathbf{u},\mathbf{v},\mathbf{c},\mathbf{w},\theta|\alpha,\gamma,\rho)] \\ &= E_q[\log p(\mathbf{x},\mathbf{z},\mathbf{u},\mathbf{v},\mathbf{c},\mathbf{w},\theta|\alpha,\gamma,\rho)] + H_q(\mathbf{u},\mathbf{v},\mathbf{c},\mathbf{w},\theta). \qquad (4.7) \\ & \qquad (4.8) \end{aligned}$$

As noted previously, this is equivalent to minimizing the KL-divergence between p and q. The optimization is done through a combination of closed-form updates and gradient descent. The overall algorithm is described in Table 4.1.

4.4 Inference

Inference in the unsupervised HPY model produce layer-level and class-level segmentations using the marginal likelihoods $P_q(z_{ji} = t)$ and $P_q(c_{jt} = k)$. The assignment probabilities are calculated using the variational posteriors, whose parameters are estimated at training time:

$$\begin{aligned} P_q(z_{ji} = t) &= \Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt} + \lambda_{jti}}}) \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau} + \lambda_{j\tau i}}})) \qquad (4.9) \\ P_q(c_{jt} = k) &= \kappa_{jtk} \\ &= \exp\{\sum_{i,l} x_{ji,l} P_q(z_{ji} = t)\, E_q \log \eta_{k,l} + \sum_{k'=1}^{k} E_q \log \omega_{k',l}\}, \text{ with} \qquad (4.10) \\ E_q[\log \eta_{k,l}] &= \Psi(\eta_{k,l}) - \Psi(\sum_{l} \eta_{k,l}), \text{ and} \\ E_q[\log \omega_{k',l}] &= \Psi(\omega_{k',t}) - \Psi(\omega_{k',t} + \omega_{k',b}). \end{aligned}$$

Here $\Psi()$ is the digamma function.

To generate the final segment-level and class-level segmentations, we use the MAP estimates of the marginal likelihoods $\arg\max_t P_q(z_{ji} = t)$ and $\arg\max_k P_q(c_{jt} = k)$. It is possible to use the full marginal assignment distributions and perform a post-processing step to ensure more coherent regions are produced.

The segmentation model described thus far is unsupervised. When annotations and labels are available, it would be beneficial to make use of them during training. In the next chapter we will discuss an extension of the variational learning procedure that naturally makes use of label information.

Chapter 5

Supervised Hierarchical Pitman-Yor Process for Scene Segmentation

In this chapter, we apply the hierarchical Pitman-Yor process, proposed by (94) and described in the previous chapter, to the task of scene segmentation, where the entire image is segmented. Moreover, we propose a novel framework to discriminatively train the model, where annotations are introduced into the variational learning as constraints. We will demonstrate the effectiveness of our approach on a subset of the LabelMe dataset (84) and present the empirical results. This work is a modified version of our publication (88).

5.1 Supervised Hierarchical Pitman-Yor Model

In this section we present our supervised hierarchical Pitman-Yor process model for scene segmentation. Recall from Eq. 4.7 in the previous chapter that we train the model with a variational approach, by optimizing the lower bound on the marginal likelihood

$$\begin{aligned} \log p(\mathbf{x}|\mathbf{w}) &\geq E_q[\log p(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{v}, \mathbf{c}, \mathbf{w}, \theta|\alpha, \gamma, \rho)] - E_q[\log q(\mathbf{u}, \mathbf{v}, \mathbf{c}, \mathbf{w}, \theta)] \\ &\equiv \mathcal{L} \end{aligned}$$

Also from the previous chapter, inference in the unsupervised HPY model produce layer-level and class-level segmentations using the MAP estimates $\arg\max_t P_q(z_{ji} = t)$ and $\arg\max_k P_q(c_{jt} = k)$, where the assignment probabilities are

$$
\begin{aligned}
P_q(z_{ji}=t) &= \Phi(\frac{\nu_{jt}-\mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}})\prod_{\tau=1}^{t-1}(1-\Phi(\frac{\nu_{j\tau}-\mu_{j\tau i}}{\sqrt{\delta_{j\tau}+\lambda_{j\tau i}}})) \\
P_q(c_{jt}=k) &= \exp\{\sum_{i,l} x_{ji,l} P_q(z_{ji}=t)\, E_q \log \eta_{k,l} + \sum_{k'=1}^{k} E_q \log \omega_{k',l}\},\ \text{with} \\
E_q[\log \eta_{k,l}] &= \Psi(\eta_{k,l}) - \Psi(\sum_{l} \eta_{k,l}),\ \text{and} \\
E_q[\log \omega_{k',l}] &= \Psi(\omega_{k',t}) - \Psi(\omega_{k',t}+\omega_{k',b}).
\end{aligned}
$$

The last two expectations are computed from the Dirichlet posterior assumption, while the equation for $P_q(c_{jt}=k)$ stems from the closed-form update for κ_{jtk}.

We now develop a novel optimization framework for adding supervision to the previously unsupervised HPY model. Let $\mathcal{A}$ be the set of annotations. We are provided with two types of annotations:

$$
\begin{aligned}
\mathcal{A} &= \left\{(a^s_{ji}, a^c_{ji}) \mid a^s_{ji} \in \{1,\ldots,T\}, a^c_{ji} \in \{1,\ldots,K\}\right\} \qquad (5.1) \\
\text{where } a^s_{ji} &: \text{segment-level annotation, and} \\
a^c_{ji} &: \text{class-level annotation.}
\end{aligned}
$$

Segment-level annotations describe the layer assignment of each observation, while class-level annotation describes the class assignment of each layer. Note that we have imposed an absolute ordering on the layers, due to the stick-breaking construction of the layer model; in practice, we sort the different layers in decreasing order of their sizes. This is partially due to the heuristic that foreground objects/layers are usually bigger.

We apply supervised constraints to the variational optimization problem, forcing the MAP assignment estimates to agree with the annotations. Learning the supervised HPY can then be formulated as the following maximization problem

$$
\begin{aligned}
\max \quad & \mathcal{L} \\
s.t. \quad & \forall (j,i) \in \mathcal{A}, P_q(z_{ji}=a^s_{ji}) \geq \max_t P_q(z_{ji}=t) \\
& \forall (j,i) \in \mathcal{A}, P_q(c_{ja^s_{ji}}=a^c_{ji}) \geq \max_k P_q(c_{ja^s_{ji}}=k)
\end{aligned}
$$

with respect to μ_{jti}, λ_{jti}, ν_{jt} and δ_{jt}. Note that the above probabilities are already defined in terms of these variables. The formulation is similar to the one used in structured prediction (95). Instead of using universal conditions (i.e., correct assignment probability greater than all assignment probabilities), we opt for the above condition involving the max operator to reduce the number of conditions. Since the objective function is already highly non-convex, we conjecture that having too many constraints (one per layer) will affect the convergence of the algorithm.

We transform the above optimization problem into a single objective by adding slack variables

$$
\begin{aligned}
\max \quad & \mathcal{L} - \sum_{(i,j)\in\mathcal{A}} (C^s \zeta^s_{ji} + C^c \zeta^c_{ji}) \\
s.t. \quad & \forall (j,i) \in \mathcal{A}, P_q(z_{ji}=a^s_{ji}) + \zeta^s_{ji} \geq \max_t P_q(z_{ji}=t) \\
& \forall (j,i) \in \mathcal{A}, P_q(c_{ja^s_{ji}}=a^c_{ji}) + \zeta^c_{ji} \geq \max_k P_q(c_{ja^s_{ji}}=k)
\end{aligned}
$$

At the maxima, the inequality constraints will be tight and become equalities. Reminiscent of the Lagrangian, the objective ℓ with soft constraints can then be defined as

$$\ell = \mathcal{L} - C^s \sum_{(i,j)\in\mathcal{A}} (\max_t P_q(z_{ji} = t) - P_q(z_{ji} = a^s_{ji})) - C^c \sum_{(i,j)\in\mathcal{A}} (\max_k P_q(c_{ja^s_{ji}} = k) - P_q(c_{ja^s_{ji}} = a^c_{ji}))$$

Ideally, the Lagrangian can be converted into its dual form by introducing dual variables for each constraint. In this case, however, the dual formulation is difficult to solved in close form so we decided on the simplified version stated above.

Maximizing ℓ defines the optimization problem we solve to learn the discriminative HPY model. The coefficients C^s and C^c determine the relative weighting the model puts on minimizing the KL divergence and minimizing the segmentation error. Ideally, these coefficients would be cross-validated; in the experiments, we set these constants to 1.

5.1.1 Learning

In the unsupervised model, gradient descent is carried out independently for each image and layer. With the segment-level and class-level constraints, the layers and images are now dependent, complicating the optimization. Since adding all constraints to the optimization is computationally expensive, we derive a cutting plane type algorithm that selects, during each iteration, the most violated constraint and adds it to the optimization. We now explain the learning process in detail (see also Table 5.1).

First, we initialize the multinomial parameters of the appearance model, setting them according to the class-level annotations. Note that it is possible that our truncated value for the number of classes, K, is less than the number of global class categories. We first sort the classes in descending frequency, and lump the truncated classes into a "background" class. For classes that have not been observed in the training data, we randomly sample their appearance model from the background class, which may exhibit extensive intraclass variability. We then initialize all the other variational posteriors randomly.

Within each image, we first train the variational parameters in the unsupervised fashion via gradient descent on the objective $\mathcal{L}$, as described in the previous chapter. Next, we permute the layers to minimize the number of violated constraints. This is necessary since we imposed an absolute ordering on the layers. The problem of computing the optimal permutation can be formulated as a bipartite matching on a graph where the nodes are the assignment labels and the edge weights are the number of agreements in the layer assignments - or more simply, the confusion matrix. The intuition behind this reformulation is that a permutation is an independent edge set (or a *matching*) since each assignment id can only be permuted to one other id, and vice versa. The matching is carried out with the Hungarian algorithm (48).

Once the layers are properly permuted, we iteratively identify the pair of layers with the most violated constraints, as motivated by the cutting-plane method in (43). This corresponds to finding the largest off-diagonal entry in the new confusion matrix. Given this pair of layers, we find the set of observations which violate these segment-level constraints, as well as the subset of observations which violate the class-level constraints. We now perform gradient descent on these sets of nodes,

Algorithm 2: DHPY

for each k = observed, non-background class
  Set $\eta_k = \frac{\sum_{j,i} \mathbf{1\{a^c_{ji}=k\}x_{j,i}}}{\sum_{j,i} \mathbf{1\{a^c_{ji}=k\}}}$
for each k = unobserved class
  Initialize η_k randomly from background class
Initialize table assignments
Initialize class assignments κ_{jt} from a^c_{ji}
Initialize $\mu_{jti}, \lambda_{jti}, \nu_{jt}, \delta_{jt}$
for each image j = 1 to J
  Run unsupervised HPY training
  for each i = 1 to N_j
    Set assign(i) = $P_q(\min\{t|u_{jti} < \Phi^{-1}(v_{jt})\} = t)$
  Set M = getOptimalPermutation(assign, a^s_{ji})
  Permute layers according to M
  Construct new confusion matrix C
  Do while $\sum_{s\neq t} C(s,t)$ stabilizes:
    Set s^*, $t^* = \arg\max_{s\neq t} C(s,t)$
    Set $k^* = \arg\max_k P_q(c_{j,s^*} = k)$
    Set $\mathbf{I}^* = \{i|P_q(z_{ji} = t^*) < P_q(z_{ji} = s^*)\}$
    Set $\mathbf{I}^{**} = \{i \in \mathbf{I}^*|P_q(c_{j,s^*} = a^c_{ji}) < P_q(c_{j,s^*} = k^*)\}$
    Set $\partial\mathcal{L}^s = \partial\mathbf{1}_{\{i\in\mathbf{I}^*\}}(P_q(z_{ji} = t^*) - P_q(z_{ji} = s^*))$
    Set $\partial\mathcal{L}^c = \partial\mathbf{1}_{\{i\in\mathbf{I}^{**}\}}(P_q(c_{j,s^*} = k^*) - P_q(c_{j,s^*} = a^c_{ji}))$
    Run gradient descent with $\partial\mathcal{L} - C^s\partial\mathcal{L}^s - C^c\partial\mathcal{L}^c$
end

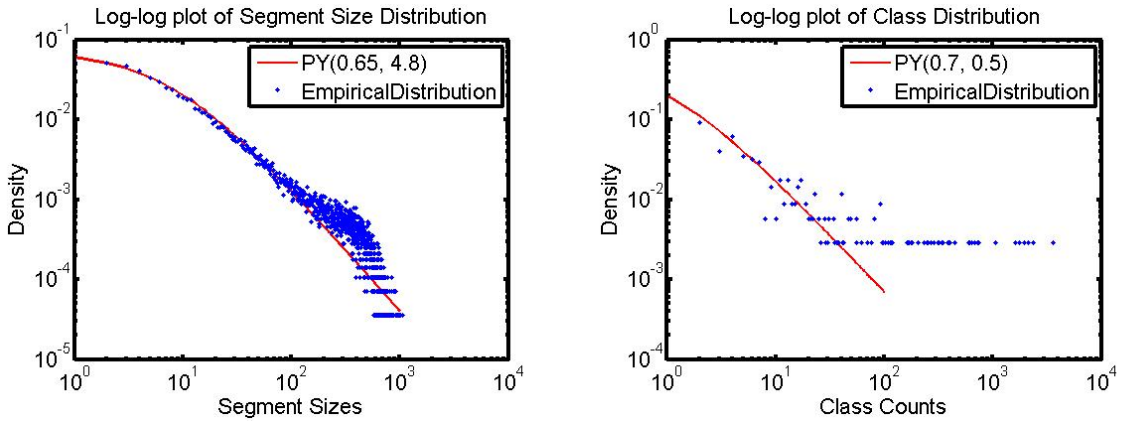
function getOptimalPermutation(assign, assign_gt)
Construct confusion matrix C:
for n = Range(assign)
  for m = Range(assign_gt)
    $C(n,m) = |\{(i,j)|\text{assign}(i) = n, \text{assign\_gt}(j) = m\}|$
Run Hungarian algorithm with weight matrix C
return matching M

Table 5.1. Discriminative HPY (DHPY) learning algorithm using Gradient Descent

using the subgradient method suggested in (80) which approximates

$$\frac{\partial}{\partial x}\max f(x) \quad \approx \quad \max\frac{\partial f(x)}{\partial x} \in \mathcal{S},$$

where $\mathcal{S}$ is the set of subgradients of $\max f(x)$. This is necessary since the max function is not differentiable. We proceed until there are no more violated constraints. In practice, however, this is rarely accomplished; hence, we iterate until the number of violated constraints stabilizes. This proves to be a robust exit condition and convergence is consistently quick; however, there is no guarantee that the actual MAP estimates are consistent.



(a) Segment Sizes (b) Class Frequencies

Figure 5.1. (a,b) Power-law empirical distributions across all categories and their fitted Pitman-Yor processes.

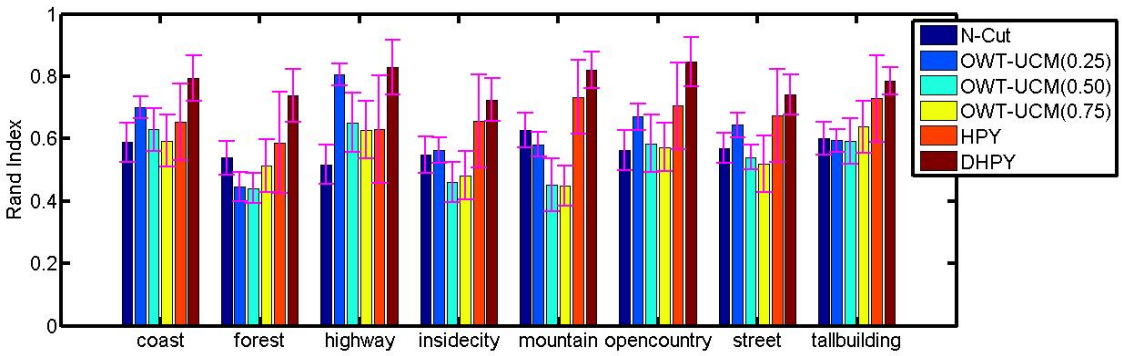
5.2 Experimental Evaluation

We validate our approach on the natural scene dataset of (75; 94), which is a subset of the LabelMe (84) database. The dataset consists of eight categories and a total of 2,688 images, each comprising a number of manually segmented polygons with a semantic text label. For all experiments, we use $\sigma_{sp} = 300$, $\sigma_{gpb} = 0.3$, $\epsilon_n = 200$, and $\epsilon_{eig} = 0.001$, which are estimated via cross-validation.

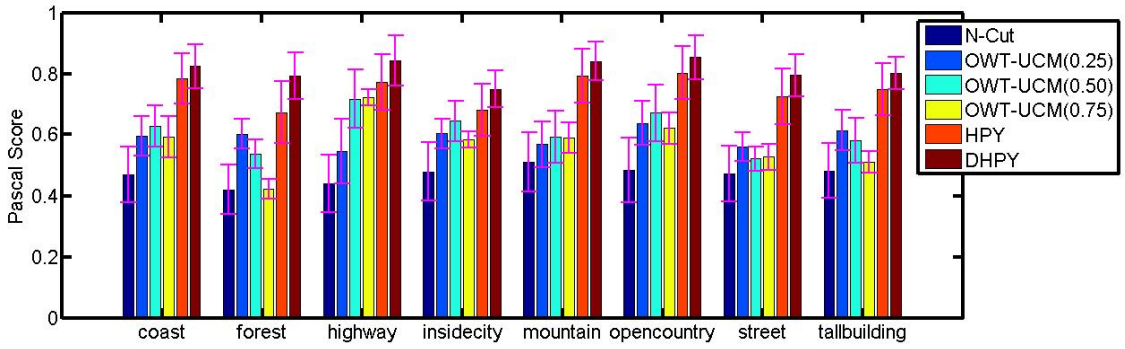
Each image is first preprocessed into roughly 1,000 superpixels. We use a local texton histogram (87) quantized to 64 bins and a color histogram quantized to 100 bins as features. For each image, the segments are sorted in decreasing order, with the largest segment assigned to be layer 1. The segment label for each superpixel is then computed as the majority segment id among the encompassing pixels. The class label is taken to be the most frequent unigram, accounting for plurals and ignoring labels marked as occluded. The empirical distributions of the segment sizes (in terms of superpixels) and class counts are shown in Fig. 5.1. The asymptotic linearity in the loglog plot is evidence of a power law distribution; fitted Pitman-Yor priors are shown.

A single HPY and DHPY model is trained for each category. In our experiments, we set the number of global classes K to be 100 (across all scene categories); we bundle the truncated classes into a background class. For computation reasons, we set the number of segments T to be the same as the number of global classes. The crucial class-level PY hyperparameters γ_a, γ_b are set to their fitted values (0.7 and 0.5 respectively). Similarly, the segment-level PY hyperparameters α_a, α_b are set to the globally fitted values (0.65 and 4.8). Note that for the DHPY, α_a and α_b can be learned given the training images for each class; however, we would need to modify the graphical model to have class-specific prior hyperparameters α_a and α_b.

The segmentation is computed as the class with the maximal posterior probability of assignment. Results are reported with respect to the ground truth (manual segmentation) in terms of two metrics: the Rand index (101) and the Pascal score. The Rand index measures the similarity



(Rand Index)



(Pascal Score)

Figure 5.2. Performance across the 8 categories for our approach and the baselines. Best viewed in color.

between two data clustering schemes. Given two cluster assignments X and Y, it is defined as

$$RandIndex(X,Y) \quad = \quad \frac{\#\text{ Agreements between X,Y}}{\text{All possible pairs}} \tag{5.2}$$

This measure is a number between 0 and 1, and does not require a nominal labeling scheme.

The Pascal score, on the other hand, measures the number of correctly labeled segments, normalized by the size of the combined area. It is also known as the Jaccard index. The definition is

$$Pascal(X,Y) \quad = \quad \frac{X \cap Y}{X \cup Y}. \tag{5.3}$$

Note that the Rand index does not require a particular assignment permutation, while the Pascal score depends on the permutation. The problem of computing the optimal Pascal score (with the optimal permutation) can be formulated as a bipartite matching problem, as explained in the previous section (see Table 5.1). The PASCAL VOC challenge (22) uses a metric very similar to our Pascal score—our Pascal score is computed at the level of superpixels, while the other is a per-pixel measure. Nevertheless, since our method of computing superpixels, TurboPixels, produces clusters of approximately the same size, the two scores can be regarded as equivalent.

We compare our supervised model (DHPY) with the unsupervised HPY model, as well as Normalized Cuts (Ncut) and the thresholded Oriented Watershed Transform - Ultrametric Contour

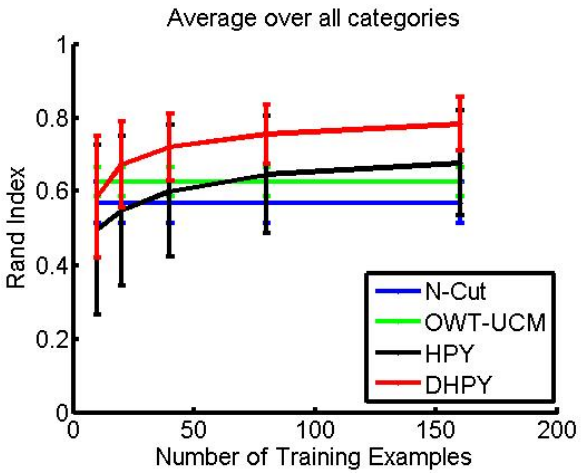


Figure 5.3. Performance in terms of Rand Index as a function of the number of training examples aggregated across categories

Map (OWT-UCM) (3). The unsupervised HPY model is initialized with the same parameters as the DHPY. The Ncut baseline is performed on the covariance matrix used in our thresholded Gaussian processes 4.4, which makes use of the discriminative gPb detector as well as an Euclidean smoothness metric. For each image, we compute Ncut assuming the number of clusters is the number of segments in the ground truth annotation. The OWT-UCM baseline is built on top of the state-of-the-art gPb contour detector; given a threshold value, closed regions can be obtained by computing the connected components in the image. Since we do not know the optimal threshold a priori, we run three OWT-UCM baselines, with the threshold being 0.25, 0.50 and 0.75.

The results across the 8 categories are shown in Fig. 5.2, in terms of the Rand index and the Pascal score respectively. Our supervised approach improves the performance of the unsupervised model while reducing the variance. Across all categories, our model achieves a Rand index of 0.7848 ± 0.0696 and a Pascal score of 0.8125 ± 0.0686. In terms of the Rand Index averaged over all categories, our method achieves an improvement of 0.1136 over the closest competitor with a p-value of 0.0223, which is statistically significant at the 5% level.

We also evaluate the Rand index performance as a function of the number of training examples. Fig. 5.2 depicts the Rand index averaged among all categories, while Fig. 5.4 depicts the index for each individual category. The OWT-UCM baseline is chosen to be the best one out of the three thresholded versions. Our model converges to the asymptotic performance more quickly, while incurring a smaller variance with fewer training data. Finally, we show segmentation outputs from the different models in Fig. 5.5.

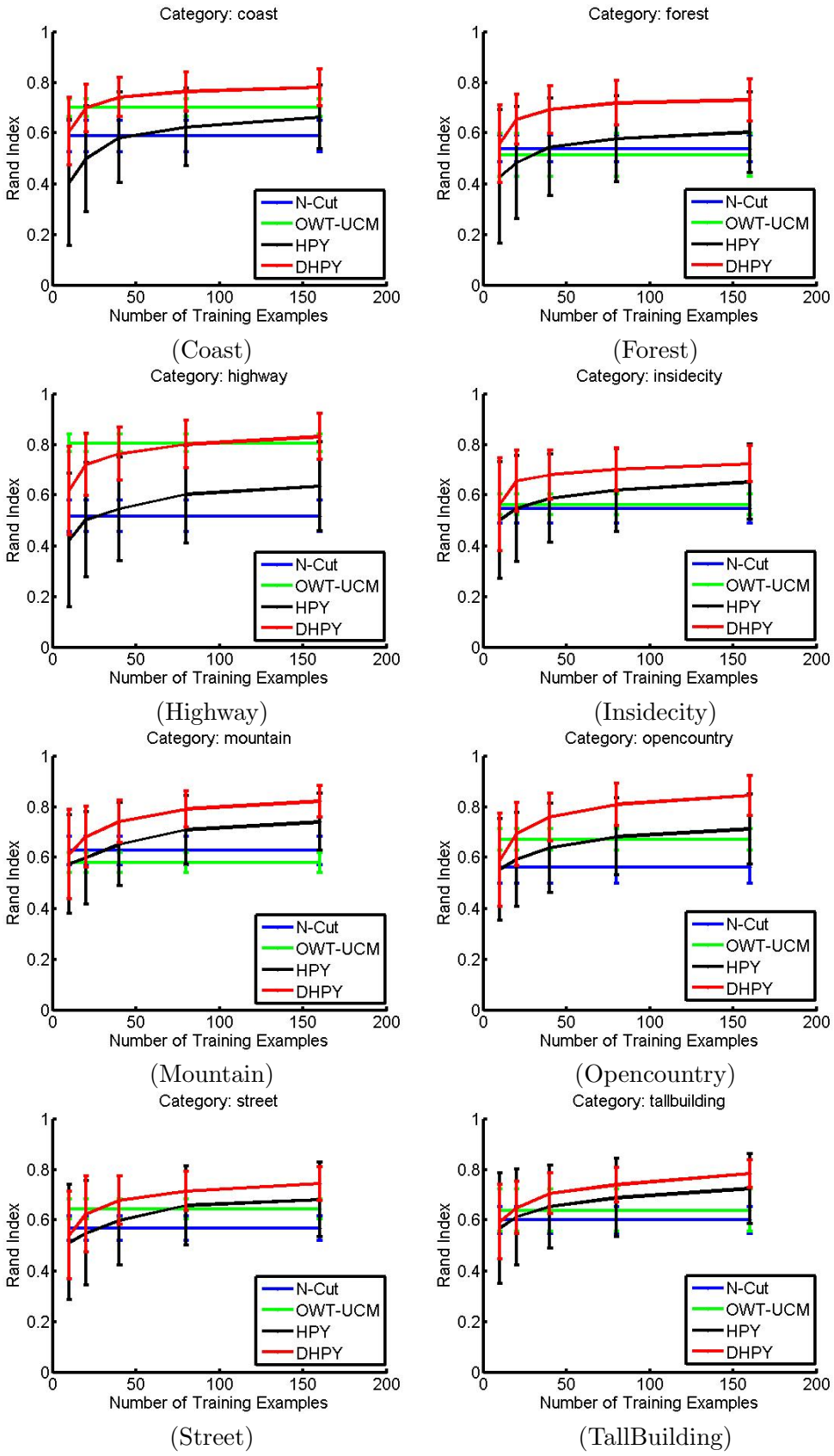
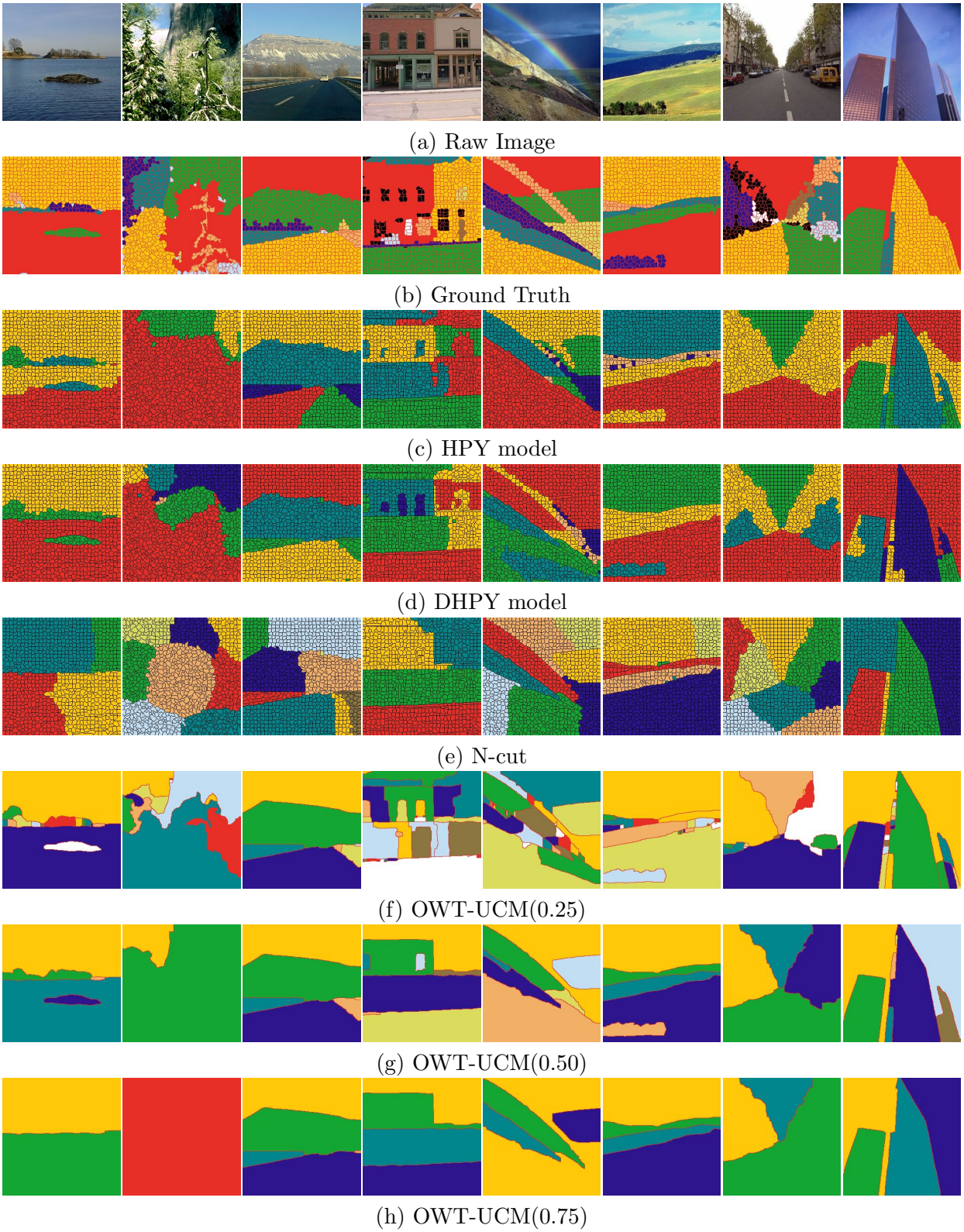


Figure 5.4. Rand Index as a function of the number of training examples for the different categories.



(a) Raw Image

(b) Ground Truth

(c) HPY model

(d) DHPY model

(e) N-cut

(f) OWT-UCM(0.25)

(g) OWT-UCM(0.50)

(h) OWT-UCM(0.75)

Figure 5.5. Segmentation results for categories *coast*, *forest*, *highway*, *insidecity*, *mountain*, *opencountry*, *street*, *tallbuilding*. The different colors represent different segments. The superpixel boundaries are also displayed for rows (b)-(e). Best viewed in color.

Chapter 6

Hierarchical Pitman-Yor Process for Object Segmentation

In chapters 4 and 5, we have been focusing on the problem of scene segmentation, where the goal is to identify and segment all regions within an image. In real world applications, however, it is often desirable or even necessary to segment out certain objects of interest. Detection is therefore a crucial operation and often a preprocessing procedure to segmentation models. Given an image, humans instinctively perform a preliminary figure-background separation, and can quickly identify the visual context and the principal objects. From the annotation collection perspective, it is also more economical to request only a few segmentations from users, than to segment the entire scene.

Thus far, the appearance model used in our model is mainly texture-based. While this suffices for scene segmentation, the notion of shape is crucial for characterizing and detecting objects. With a texture-based set of features, two people wearing different colors and styles of clothing will likely be classified as two different object classes; only by directly representing shape will the task be done correctly.

In this chapter, we will tackle the problem of object segmentation. We propose an extension to the exiting HPY model. For simplicity, we will be modifying the unsupervised HPY model in chapter 4. In particular, we augment each object class with its own shape prior. We will first present the modified graphical model and derive the corresponding variational inference procedure. It is worthwhile to note that Sudderth (92) has devised a transformed Dirichlet process, a precursor to the HPY model, in which the Gaussian means are allowed to be perturbed. However, the particular transformation is limited to translation. We then introduce the classic Procrustes analysis and demonstrate how to formulate a suitable shape prior that can be incorporated into the HPY model. As a byproduct of the analysis, we devise a novel algorithm to cluster shape automatically, which is invariant to affine transformation. Finally, we conclude with an evaluation of our extended HPY model.

6.1 HPY model with Shape Prior

Recall that in the HPY model, superpixels are segmented into layers using thresholded Gaussian processes. These GPs have discriminative covariance functions which utilize the gPb detector, and encourage superpixels without strong edges between them to be clustered within the same layer. They, however, have zero as their means, placing no preference on which layer the superpixels get assigned to. We will add class-dependent means to these previously zero-mean GPs, and allow these means (or shape templates) to undergo affine transformations. This will enable the layer assignment probabilities to be influenced by shape.

We introduce a function $F(\mathbf{S}; a)$, which outputs a final shape mask of the template mask $\mathbf{S}$ under the affine transformation a. Since the HPY model operates on superpixels, we average the final shape mask within each superpixel; hence, superpixels on the boundary of the final shape mask will have a $F(\cdot)$ value between 0 and 1. As $\mathbf{S} \in [0, 1]$, we scale and translate $F(\mathbf{S}; a)$ to fall in $[-1, 1]$. The affine transformation a will be determined by matching each shape template to the current layer proposal.

Adding shape prior, the new model assumptions are

$$\mathbf{u_{jt}}|I_j, a_{jt}, c_{jt}, \vec{S_k} \sim \mathcal{N}(F(\mathbf{S_{c_{jt}}}; a), \mathbf{W_j}). \tag{6.1}$$

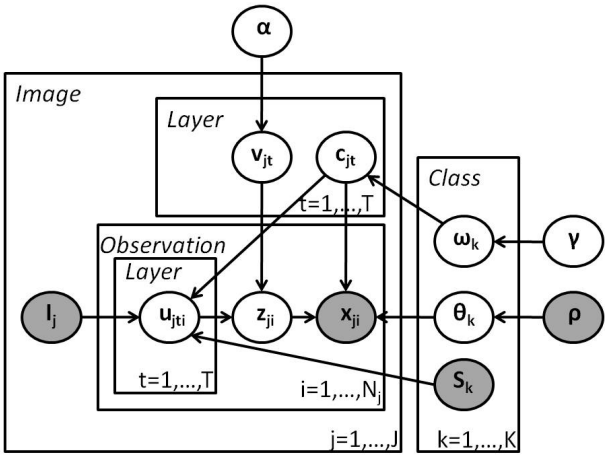


Figure 6.1. HPY model with Shape Prior

The updated model is depicted in Fig.6.1. Note that the class assignment variable c_{jt} for each layer now determines both the current layer's appearance model and shape prior. Since $u_{jt} - F(\mathbf{S_{c_{jt}}}, a_{jt}) \sim \mathcal{N}(0, \mathbf{W_j})$, as before, the layer assignment decision rule now becomes:

$$z_{ji} = \min\{t | u_{jti} - F(\mathbf{S_{c_{jt}}}, a_{jt})_i < \Phi^{-1}(v_{jt})\}, \tag{6.2}$$

and the corresponding variational posterior probability $\mathbb{P}_{\shortparallel}(z_{ji} = t | z_{ji} \neq t-1, ..., 1)$ becomes

$$\begin{aligned}\mathbb{P}_{\shortparallel}(z_{ji} = t | z_{ji} \neq t-1) &= \mathbb{P}_{\shortparallel}(u_{jti} - F(\mathbf{S}_{\mathbf{c_{jt}}}, a_{jt})_i < \bar{v_{jt}}) \\ &= \Phi(\frac{\nu_{jt} - \mu_{jti} + F(\mathbf{S}_{\mathbf{c_{jt}}}, a_{jt})_i}{\sqrt{\delta_{jt} + \lambda_{jti}}}).\end{aligned}$$

Detailed derivations of the new update rules are shown in Appendix B. In particular, the class assignment posterior is now updated not only based on appearance, but also on the level of match for the proposed mask at each layer. Thus far, we have not specified a particular algorithm for shape representation. Any suitable shape prior and representation should

- Enable efficient and robust matching, invariant to translation, scaling and rotation, and
- Be capable of representing real-valued probability of a pixel/superpixel's membership to a particular shape.

The next section will be devoted to the Procrustes analysis, and a detailed extension of how we will use it to achieve the above properties and for integrating into the HPY model.

6.2 Procrustes Analysis and Clustering

Procrustes analysis (38; 64) is a classical technique for alignment, and it aims to find the optimal affine transformation for matching two sets of points. Given two sets of N aligned points $\mathbf{X} \in \Re^{N \times d}$ (the *template*) and $\mathbf{Y} \in \Re^{N \times d}$, the task of matching $\mathbf{Y}$ to $\mathbf{X}$ is formulated as the following optimization problem

$$d(\mathbf{X}, \mathbf{Y}) = \min_{b,T,c} \sum_{i=1}^{N} ||X_i - (bY_iT + c)||_2^2, \tag{6.3}$$

where $b \in \Re$ is the scaling parameter, $T \in \Re^{d \times d}$ is the projection matrix and $c \in \Re$ characterizes translation. Note that T is a rotation matrix which can possibly include a reflection component, with $\det[T] = \pm 1$. The optimization can be solved efficiently via the singular value decomposition (SVD) algorithm. For comparison across templates of different sizes, we scale the distance by the sum of squared error of $\mathbf{X}$ from its mean:

$$d_p(\mathbf{X}, \mathbf{Y}) = \frac{d(\mathbf{X}, \mathbf{Y})}{\sum_{i=1}^{N} ||X_i - \bar{X}||_2^2}. \tag{6.4}$$

We call this the *Procrustes distance*, which is the matching error between a pair of coordinate vectors. In order to apply Procrustes analysis to shapes, we require each shape mask to be represented with a set of aligned landmarks. Without settling on a particular technique to extract landmarks, we represent each shape with the coordinates sampled along its contour. Since Procrustes analysis operates on coordinate vectors with the same number of points, we subsample each shape's contour down to a standard number of points.

Generalized Procrustes analysis (GPA) is an extension of the ordinary Procrustes analysis, and it superimposes, instead of a pair of shapes, a set of shape masks onto a central "mean" vector, with the criterion of each pair-wise estimation being the Procrustes distance. The algorithm is carried out with a two-step iterative procedure

1. Matching phase: given a mean vector, match each coordinate vector onto the mean shape.
2. Estimation phase: given the transformed shapes, estimate the mean shape.

The computation of the mean shape deserves some treatment. A simple arithmetic average of each aligned point is the first choice of estimation method; however, it is easily influenced by outliers, which adversely affects the algorithm's convergence. We will use a weighted average, where the weighting w is inversely proportional to the Procrustes distance d_p. In particular, we apply the function $w(d_p) \propto \exp(-\alpha\sqrt{d_p})$, where α can be manually specified or cross-validated for each object category. It is possible to devise more sophisticated mechanisms such as computing the spatial distribution of each aligned point (similar to the approach used in the Point Distribution model (17)).

Note that we need to address the issue of landmark alignment (also known as the *correspondence problem*), as the input vectors to Procrustes analysis are assumed to be aligned. We observe that the problem is equivalent to choosing a suitable starting location for the contour representation. As the Procrustes distance can be computed efficiently for each configuration, we simply search over each starting index and choose the one with the minimal distance. The GPA algorithm is outlined in Table 6.1.

Algorithm 3: Generalized Procrustes Analysis
Input: Shapes $\mathbf{X_1}$, ..., $\mathbf{X_n}$ represented by landmark coordinates
Output: Mean shape $\bar{\mathbf{X}}$

Randomly sample a shape from $\{\mathbf{X_i}\}$ to be $\bar{\mathbf{X}}$'
Set $\bar{\mathbf{X}} = \vec{0}$
while $||\bar{\mathbf{X}} - \bar{\mathbf{X}}'|| > \epsilon$
  Set $\bar{\mathbf{X}} = \bar{\mathbf{X}}$'
  % Matching Phase
  for each i = 1 to n
    % Search for best aligned configuration
    for each i' = 1 to n
      Match $\mathbf{X_i}$ starting with the i'^{th} point to $\bar{\mathbf{X}}$ using Procrustes Analysis

    Let d_p be the minimal distance, and
     $\mathbf{Z_i}$ be the corresponding transformed coordinates

  % Estimation Phase
  Set $\bar{\mathbf{X}}$' = Mean shape of the $\mathbf{Z_i}$'s
end
return $\bar{\mathbf{X}}$

Table 6.1. Generalized Procrustes Analysis, with optimal alignment

Given a set of shapes, the GPA algorithm finds the average shape that is invariant to translation,

rotation and scaling. However, it is often desirable to find clusters of shapes which are invariant to affine transformation, analogous to the popular k-means clustering. We borrow intuitions from k-means clustering and formulate the *k-Procrustes clustering*. It follows a similar iterative scheme of

1. Given cluster mean shapes $\{\bar{\mathbf{X}}_\mathbf{k}\}$, match and assign each shape to the closest cluster.
2. Given cluster assignments, estimate cluster means.

The cluster means can be computed with GPA. The clustering algorithm is then repeated until there is no more change to each shape's cluster assignment. The algorithm is outlined in Table 6.2.

Algorithm 4: k-Procrustes Clustering
Input: Shapes $\mathbf{X_1}$, ..., $\mathbf{X_n}$ represented by landmark coordinates
Number of clusters K
Output: Shape cluster means $\bar{\mathbf{X}}_\mathbf{1}$, ..., $\bar{\mathbf{X}}_\mathbf{K}$

```
Randomly sample K shapes from {X_i} to be {X̄_k}
while (true)
    Set X̄ = X̄'
    % Matching Phase
    for each i = 1 to n
      for each k = 1 to K
        Match X_i to X̄_k using Procrustes Analysis, allowing for best alignment
      Assign X_i to the best matched cluster
    if (assignment changes)
      break

    % Estimation Phase
    for each k = 1 to K
      Set X̄_k = GPA (X_i's assigned to cluster k)
end

% Post-processing
Prune clusters with a small number of closely aligned examples
Merge clusters which have similar means

return {X̄_k}
```

Table 6.2. k-Procrustes Clustering

We have included several post-processing methods to improve the resulting clusters. Pruning small clusters gets rid of outlier shapes which only have one or two instances; these instances can be rarely occurring shapes or, more frequently, a result of human annotation error. Next, we combine clusters by computing the Procrustes distance between pairs of cluster means and hierarchically merging them if the distance is above a threshold; if merging occurs, the cluster means are recomputed using GPA. This removes similar and redundant clusters that are introduced with the random initialization.

Similar to k-means clustering, the quality of the k-Procrustes clustering can be measured by the amount of variance explained, which is the ratio of the between-group variation to the total variation. The only modification here is that the Euclidean distance has been replaced by the Procrustes distance. The final k-Procrustes clustering algorithm is repeated, each time with a different initialization; the clusters with the best quality is then chosen.

We have thus far formulated an efficient and robust way of matching shapes, representing each mask as a series of points sampled along its contour. What remains is to characterize the real-valued membership probability. The naive method of assigning 0 to pixels outside the mean shape (and 1 to pixels inside) will not generalize well. From the k-Procrustes clustering procedure, we obtain both the mean shapes and the transformed points of each shape onto its assigned cluster mean. We can then apply the same transformation to each shape mask and compute a weighted average of the masks obtain a smoothed mask. The weighting scheme previously used for computing the shape means can be apply here. The output is then a membership probability distribution over pixels belonging to a shape cluster, which is invariant to affine transformation and robust to outliers.

To summarize our treatment of shape, each shape is captured with two characterizations:

1. *Contour representation*: points sampled along the contour of the shape mask, depicting the 0-1 decision boundary. These points are used for efficient matching and for computing the proper transformation. This representation will be responsible for estimating the affine transformation a as defined in Section 6.1.

2. *Mask representation*: weighted mean of aligned masks, expressing the membership probability of an object over pixels. Averaged across superpixel boundaries, this distribution will be the $F(\mathbf{S}; a)$ function described in Section 6.1.

This dual representation of shape will constitute the shape element $\mathbf{S}_k$ for each object class. The training procedure for the HPY model with shape prior is summarized in Table 6.3.

6.3 Experimental Evaluation

We will validate the effectiveness of our shape representation and the extended HPY model with shape prior on the popular and challenging PASCAL VOC Segmentation challenge dataset (22). The dataset consists of 1,928 images, each containing one or more objects belonging to 20 different object categories. The object categories range from rigid objects such as airplanes, boats, cars and tables to non-rigid objects such as cats, horses and people. Groundtruth object-level and class-level segmentations are provided for training.

6.3.1 Shape prior

We first examine the shape clusters generated by the k-Procrustes clustering. We set the number of clusters (k) to be 10; for post-processing, we prune clusters which have 3 or less members, and merge clusters whose means have Procrustes distance less than 0.1. The cluster results for airplanes and horses are shown in Figures 6.2 and 6.3. The contour and mask representations are plotted for each cluster, and the 5 shape instances with the smallest Procrustes distance are displayed. For

Algorithm 5: HPY with Shape Prior

Initialize variational parameters
for N iterations
  for each image j
    for each layer t = T-1, ..., 1
      Compute MAP estimate for current layer $\hat{S_{jt}} = \arg\max_i P_q(z_{ji} = t)$
      Convert $\hat{S_{jt}}$ from superpixels to pixels, and compute contour coordinates
      for each class k = 1, ..., K
        Compute affine transformation a_{jtk} by matching $\hat{S_{jt}}$ to contour representation $\mathbf{S}_k$
        Compute $F(\mathbf{S}_k; a_{jtk})$ by applying transformation a_{jtk} to mask representation $\mathbf{S}_k$
      Update GPs and thresholds $(\mu_{\mathbf{jt}}, \lambda_{\mathbf{jt}}, \nu_{jt}, \delta_{jt})$ with gradient descent
    Update class assignments $\kappa_{\mathbf{j}}$ with closed form solution
    Incrementally update class appearance model η
    Incrementally update stick breaking distribution for classes $\omega_{,\mathbf{a}}$ and $\omega_{,\mathbf{b}}$

  if (variational lower bound $\mathcal{L}$ converges within ϵ)
    break
end

Table 6.3. Variational learning algorithm for the HPY model with shape prior

the remaining 18 categories, the contours and masks are shown in Figure 6.4 without the shape instances.

At a quick glance, the Procrustes clustering theme is capable of capturing different shape modes within an object class. Simple rigid objects such as bottles, buses, sofas, trains and monitors can be succinctly summarized by one shape. The more complex rigid shapes have more modes, either due to intraclass variability (such as the different types of chairs and boats) or difference in perspective (such as the front and side views of a bike). Non-rigid objects inherently generate more variations due to their configurable poses. In the case of horses, these variations are captured succinctly within a small number of clusters; for instance, the side-view cluster (cluster 2 in Fig. 6.3 contains a wide range of head tilt angles with the same body position. In other categories such as person, the wide range of poses resulted in a larger number of clusters.

It is worthwhile to note that our k-Procrustes clustering has not addressed occlusion, as can be seen from the constituents of Cluster 1 for horses (in Fig. 6.3). This suggests that a partial matching scheme, instead of the current complete Procrustes matching, can improve the clustering quality. We will outline a few potential extensions to address this shortcoming in the future work section.

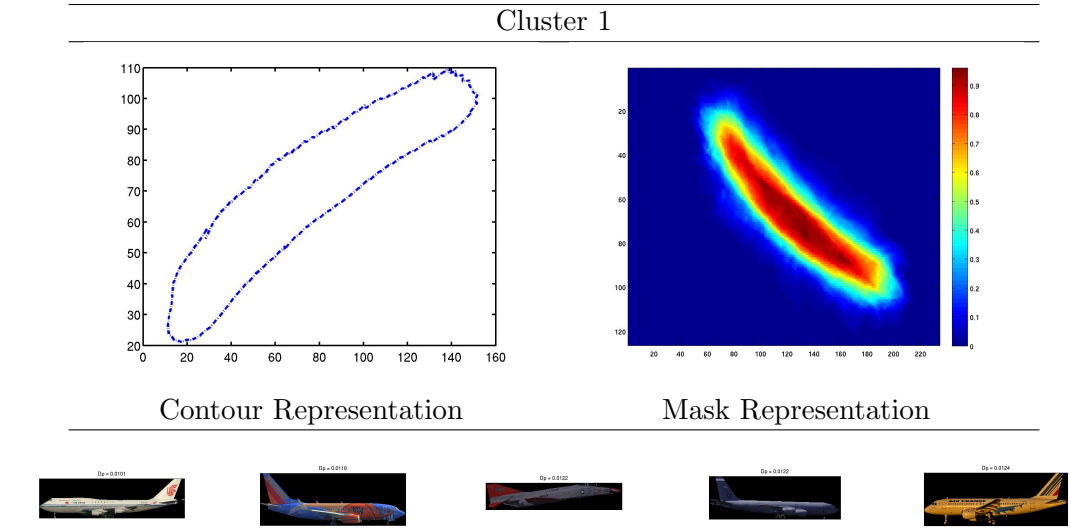


Figure 6.2. Shape clusters obtained via Procrustes clustering for the object class *Airplane*

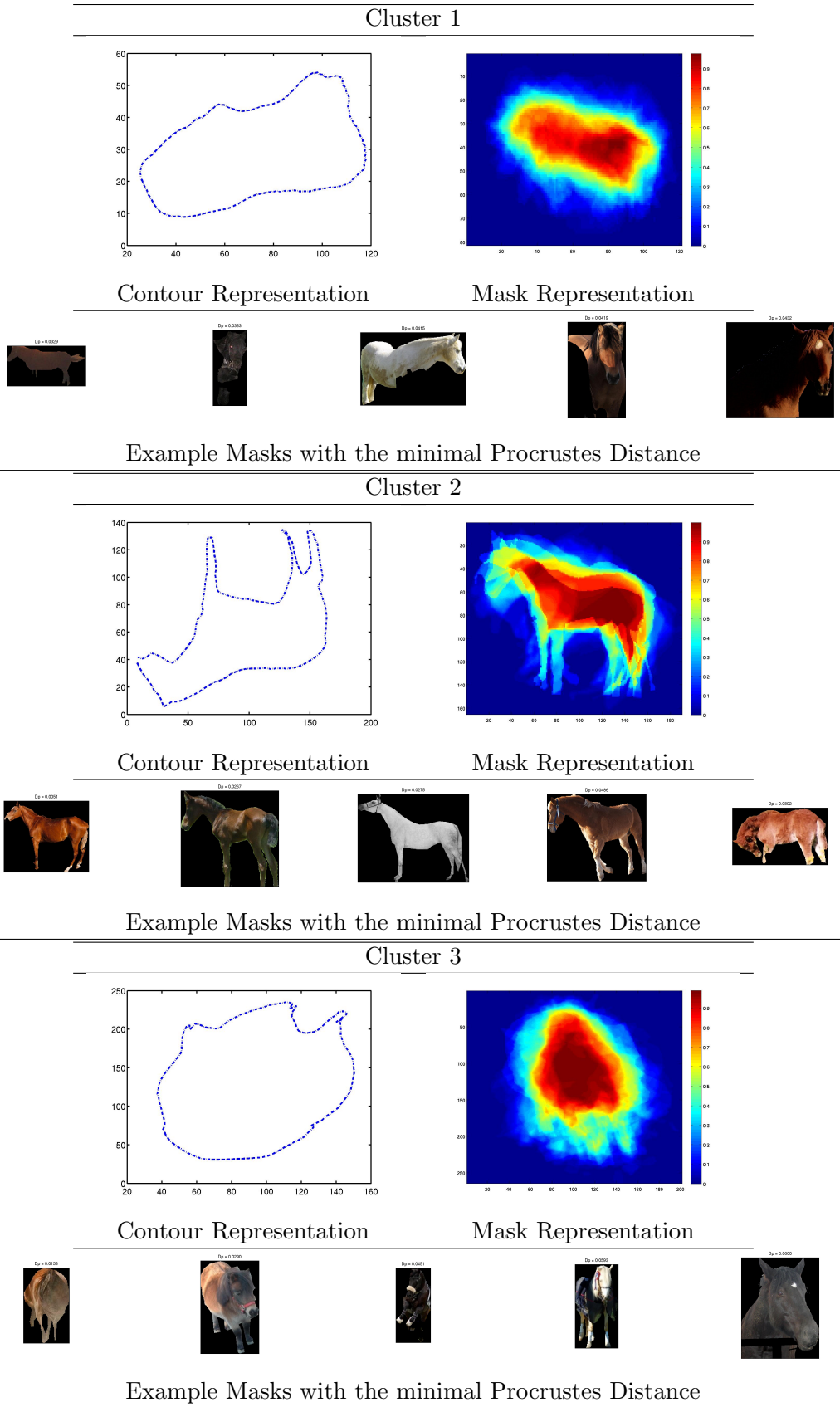
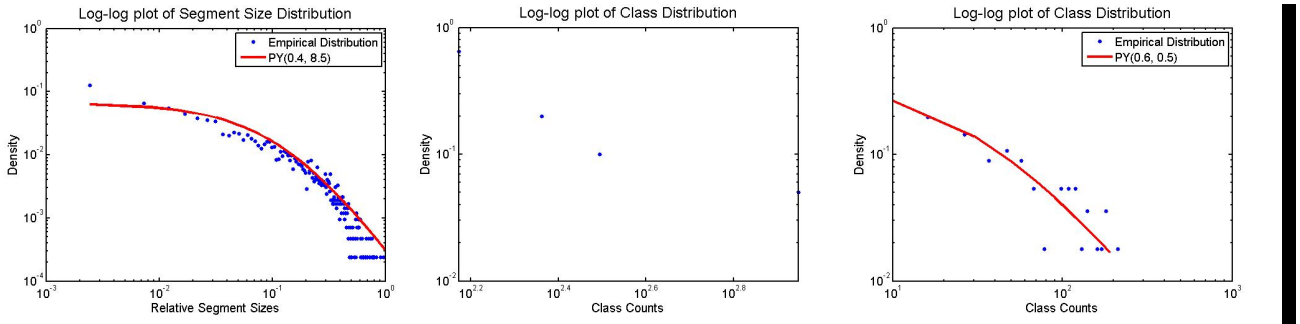


Figure 6.3. Shape clusters obtained via Procrustes clustering for the object class *Horse*

Figure 6.4. Shape clusters obtained via Procrustes clustering for the remaining object classes

6.3.2 HPY model with shape prior

We now apply our extended HPY model to the segmentation task. As for the LabelMe dataset in chapter 5, we first confirm the prior assumptions of the HPY model. The empirical distributions of the segment sizes and class counts are plotted in Figure 6.5. The original 20 class categories have too few data points to fit a PY process (shown in Figure 6.5(b)). After applying k-Procrustes clustering, the number of classes are expanded to 56 and there is enough samples for the class distribution to be fitted.



(a) Segment Sizes (b) Class Frequencies (c) Augmented Class Frequencies

Figure 6.5. (a,b,c) Power-law empirical distributions across all categories and their fitted Pitman-Yor processes.

Following the variational training procedure outlined in Table 6.3, an initialization scheme for the variational posterior parameters is required. A potential choice is to run a few iterations of the original HPY training; however, the layer MAP estimates initialized this way will be purely based on texture and color. As our shape matching depends on the contour representation of the layer MAP estimate, we need an initialization scheme which is shape-based. The discriminatively trained deformable parts detector (24) is an obvious candidate, which is also a component of many existing segmentation models. A detector is trained for each object class, and outputs the detection bounding boxes. Given these bounding boxes, we set the layer MAP estimates via the constrained variational optimization framework described in chapter 5. In the case where detection fails, we resort to the original HPY training as the default initialization scheme.

Extra attention is necessary to strike a balance between the original texture-based model and the new shape-based estimation. Firstly, the discriminative power of the spatial covariance matrix $\mathbf{W}_j$ (defined in Eq. 4.4) should be relaxed, as objects might span across boundaries with sharp texture or color contrast; to achieve this relaxation, we decrease σ_{gpb} from 0.3 to 0.2. Furthermore, objects under the same category (e.g. horse) but with different shape labels (e.g. side-view vs. front-view) can share the same appearance model; this is reflected in the variational updates for the appearance variational posterior η_k. Lastly, when computing the contour coordinates of a layer estimate, if the MAP estimate has disjoint regions, we attempt to "stitch" together regions based on the closest points of contact.

A single HPY model with shape prior is trained for each category. In our experiments, we set the number of global classes K to be 100, and the number of segments T to be 10. The ratio of the training set size to the test set size is set to be 50-50, and we run the experiments over 5 splits of the data. One should note that this experimentation scheme is different from the usual PASCAL VOC validation procedure, where the same training set, validation set and test set are used. The

	Overall	Background	Airplane	Bicycle	Bird	Boat	Bottle	Bus
HPY	19.6	68.1	27.4	7.3	12.5	23.6	14.7	38.2
Detector	18.2		28.2	4.6	12.2	21.5	19.6	43.2
HPYShape	27.8	75.4	42.3	11.6	26.7	29.4	25.1	50.4
		Car	Cat	Chair	Cow	Table	Dog	Horse
HPY		35.6	12.4	1.5	14.6	10.6	10.3	14.3
Detector		33.8	10.5	1.7	17.8	15.2	11.4	13.6
HPYShape		47.3	16.1	3.2	24.7	12.8	18.4	23.6
		Motor Bike	Person	Plant	Sheep	Sofa	Train	Monitor
HPY		28.9	5.6	8.7	22.5	10.2	12.1	21.6
Detector		31.4	10.6	12.4	24.5	15.7	24.5	30.4
HPYShape		42.6	14.3	14.3	32.1	14.6	28.3	31.3

Table 6.4. Segmentation performance of HPY and HPYShape across the 20 categories of the PASCAL VOC Segmentation challenge, in terms of Pascal score.

Pitman-Yor prior hyperparameters are set to their fitted values (as shown in Fig. 6.5). At test time, each of the 20 category models is run and the layer with the highest posterior probability of class assignment is iteratively chosen. Since we only care about the 20 object categories, we map all other classes into the "background" class.

Our HPY model with shape prior is quantitatively compared with the original unsupervised HPY model, as well as the initialization scheme with the deformable parts detector. The metric is the Pascal score, as defined in Eq. 5.3. Results across the 20 categories, the background and the overall average are shown in Table 6.4 and Fig. 6.6. We can see that the addition of the shape prior consistently improves the segmentation in all categories, from an overall Pascal score of 19.6 to 27.8. There is also a consistent improvement of our segmentation model over the initialization scheme, except for categories *Dining Table* and *Sofa*, from an overall score of 18.2 to 27.8. These exceptional categories are well suited for the bounding box initialization scheme; however, as they are often occluded by objects (e.g., on the table or sofa), our shape model is not optimal.

The categories where the highest improvements are observed include trains (16.2), planes (14.9), motor bikes (13.7) and buses (12.2). In these cases, the objects are distinct from their surroundings (such as sky and road), and the intermediate proposed segments are correctly matched to their respective shape classes. The same can be said for some non-rigid shapes such as birds (14.2), cows (10.1), sheep (9.6) and horses (9.3), where the objects of interest are distinctively colorful or unique compared to the background.

While significant improvement is achieved by augmenting the HPY model with a shape prior, its performance is still not competitive enough against the state-of-the-art object segmentation algorithms. The winner of the 2010 PASCAL VOC Segmentation challenge (35) adopts the CRF framework, with unary potentials based on a local classification score and a global classification score of each superpixel, the usual pairwise smoothness potential and a *harmony potential*. The harmony potential represents the likelihood of different combinations of class labels, and is a prior distribution on the power set of the class labels. The authors of (35) have devised a ranked subsampling method to efficiently select the labels most likely to appear in the optimal configurations, which can be adapted for the branch-and-bound framework. This method is currently the leader in most of the categories, with an overall score of 40.1. The runner up model is the two-step *Constrained*

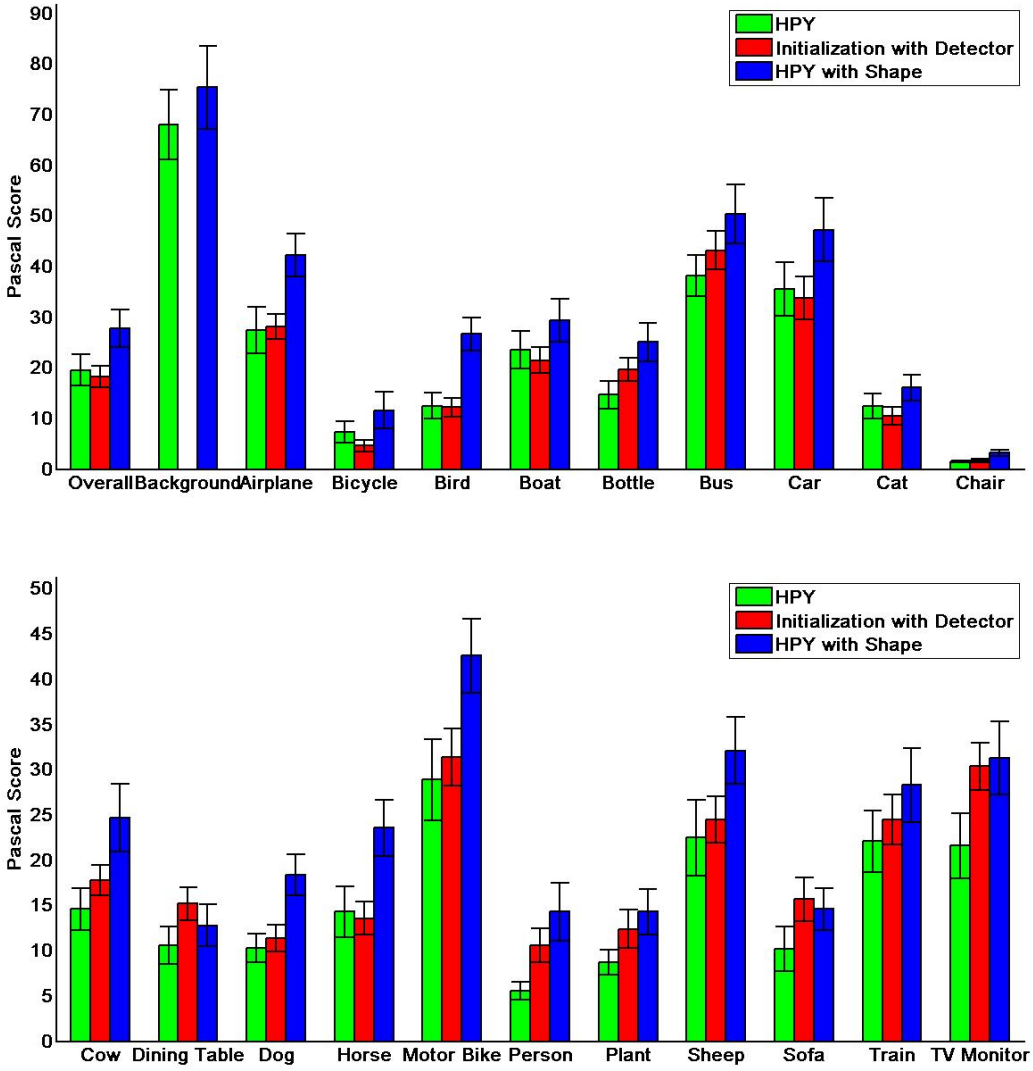


Figure 6.6. Segmentation performance of HPY, the deformable parts detector initialization scheme and HPYShape across the 20 categories of the PASCAL VOC Segmentation challenge.

Parametric Min Cuts (CPMC) algorithm (13) described in detail in chapter 2. It achieves an overall score of 39.7.

To gain more insight into the results, we will show some qualitative object-level segmentations from the original HPY model and the new HPYShape model, before carrying out the background class mapping. The results are shown in Figures 6.7 and 6.8. In the first set of examples in Fig. 6.7, we display some examples where the shape prior has proved useful and improves the segmentation. As the original model is based entirely on texture and color, objects such as the bottle and bird that exhibit a wide variation in color and texture will be segmented into separate regions. The shape prior is able to strike a balance between low-level boundaries (given by gPb contour cues)

and high-level object shapes. In these cases, the contour representation is capable of matching each region to the correct shape class.

In the second set of examples (Fig. 6.8), we can see the shortcomings of our shape representation. In the first and last images (two bottles and a jet), the segmentations become dominated by appearance as the background is of similar texture and color as the objects. In the second image, while the yellow bird is correctly segmented with a small partial occlusion, the second occluded white bird is incorrectly labeled as its contour is unrecognizable. The third and fourth images show that humans are particularly difficult, as their shapes often conform to its surrounding objects, rendering our contour representation useless. The fifth image of a bike demonstrates, yet again, the contour representation is ineffective for objects with interior holes.

These observations demonstrate that our introduction of the shape prior shows promising results. The main areas of weakness can be summarized as follows:

- **Shape features**
  The current HPYShape model depends heavily on the initializing detection to generate a reasonable region proposal. The inclusion of shape features such as HOGS/PHOGS as another feature channel will allow the model to make use of shape information more effectively. The Dirichlet hyperparameters ρ can be cross-validated to optimize for a balance between the different feature channels.

- **Unsupervised HPY**
  HPYShape in its current form is almost unsupervised, since the shape prior is generated from annotated shape masks. By switching to the discriminative DHPY model described in the previous chapter, better appearance parameters can be learned from training data.

- **Class label co-occurrence**
  Learning from the 2010 VOC segmentation challenge winner, imposing a prior on class label co-occurrence for adjacent superpixels can greatly improve results. Moreover, as the HPY model also assigns a class to background regions, the improvement can potentially be greater; for instance, objects in front of blue skies are likely to be planes.

- **Contour representation of shape**
  As demonstrated in the examples of bicycles and chairs, the contour representation of shape is inadequate for capturing object categories whose masks often have holes or comprise of disjoint parts.

- **Matching for occlusion**
  The inability of the procrustes algorithm to carry out partial matching is arguably the biggest drawback of our current HPYShape model. From the empirical results in Table 6.4, the classes where shape has the least improvement include chairs (1.7), tables (2.2), cats (3.7) and bicycles (4.3). These classes are often occluded, resulting in poor performance.

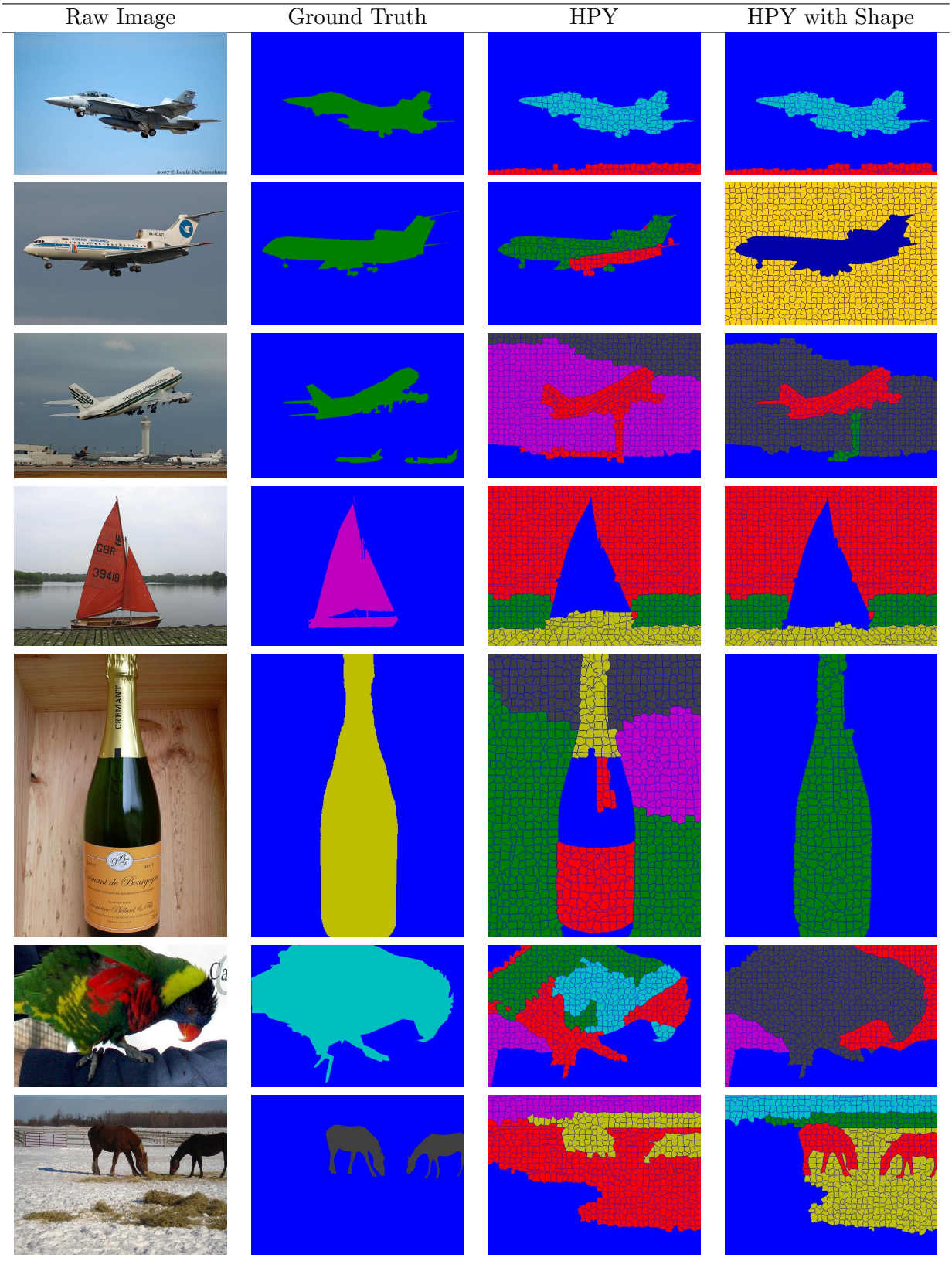


Figure 6.7. Segmentation results of the HPY model and the new HPY model with shape prior, where shape is adequate. Different colors represent different segments.

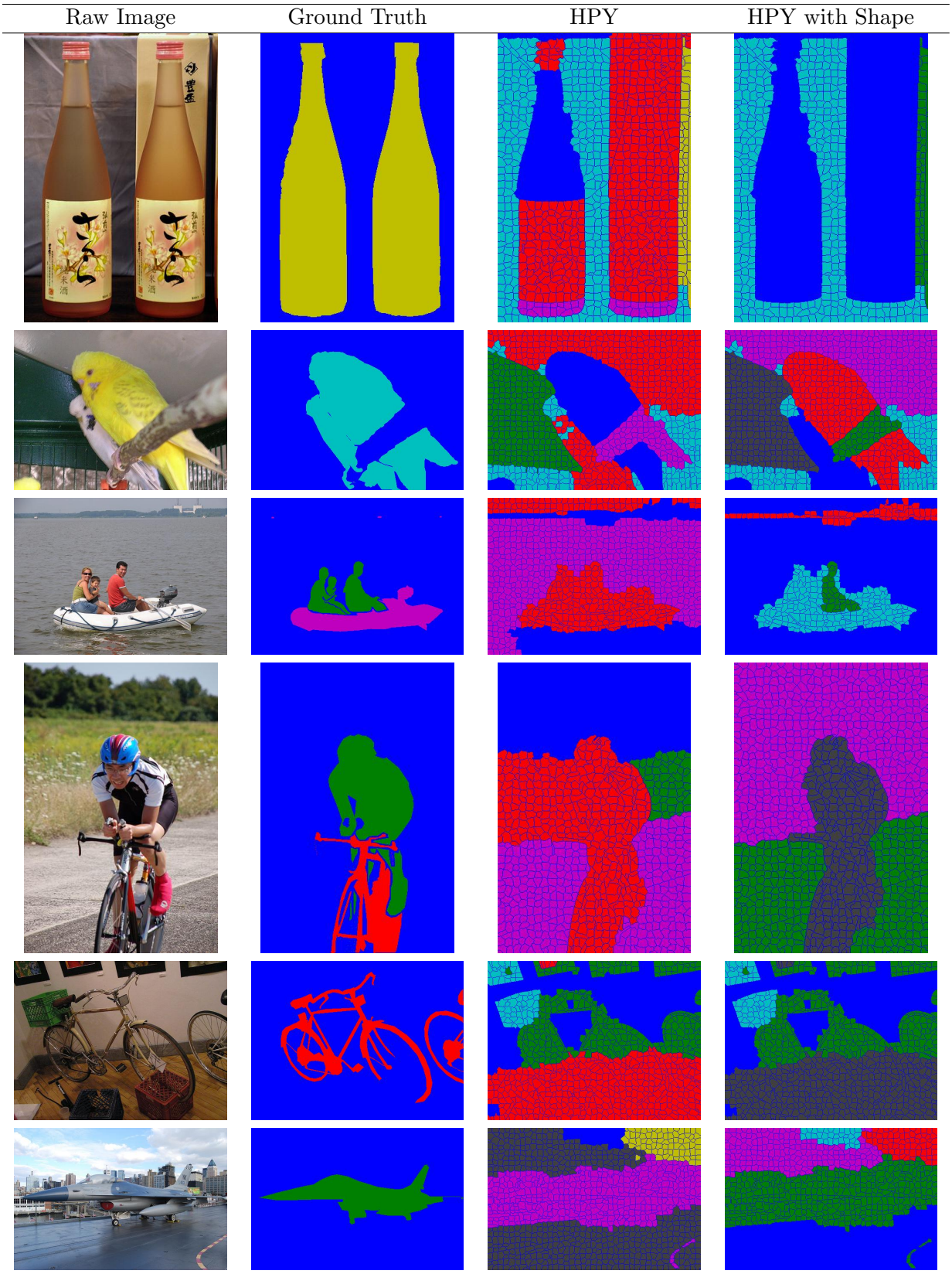


Figure 6.8. (Continued) Segmentation results of the HPY model and the new HPY model with shape prior, where shape is inadequate. Different colors represent different segments.

6.4 Conclusion

We have demonstrated that our dual representation of shape can capture the modes of intraclass variations. The extended HPY model with shape prior proves to be a significant improvement of the original HPY model; however, shortcomings of the model become apparent immediately. We will outline the various directions for future work in the next chapter as conclusion.

Chapter 7

Conclusion and Future Work

In this dissertation, we tackled the problems of sequence recognition and segmentation, and showed new ways to incorporate label data into existing models. For **sequence recognition**, we extended the kernel dimension reduction (KDR) (32) framework by designing a new dynamics kernel which can automatically estimate periodicity in the latent space, and by applying the dynamic time warping (DTW) algorithm in the form of kernel construction and regularization. Supported by experimental results, we conclude that DTW is effective in nonlinear temporal alignment.

For the task of **scene segmentation**, we gave a brief overview of the hierarchical Pitman-Yor (HPY) process (94) and improved its variational training method to make use of segment-level and class-level annotations. Our contribution lies in our discriminative training of the nonparametric Bayesian model, with a constrained variational learning framework.

For **object segmentation**, we further extended the HPY model to include class-level shape priors. Based on the Procrustes analysis, we devised a new clustering scheme which is invariant to affine transformations. Using this new clustering algorithm as the basis for our dual representation of shapes, we derived the variational learning procedure for the new HPY model with shape prior. Experimental validation is carried out on the PASCAL VOC segmentation challenge dataset, which saw significant improvements over the original HPY model. The conclusion is our shape representation works well for objects under "normal" conditions (e.g., fully visible, before an unambiguous background), but not robust for all conditions.

7.1 Future Work

We outline below a few promising venues for future research.

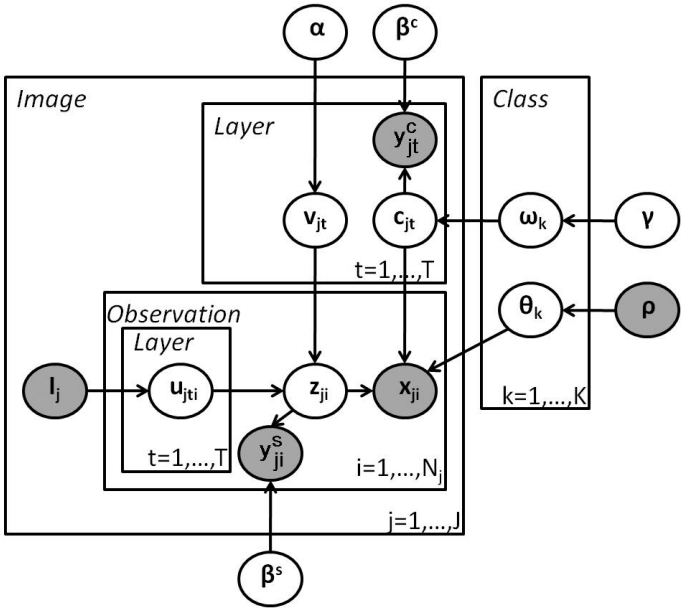


Figure 7.1. Discriminative modification #1 to the HPY model - sHPY

7.1.1 Dimensionality reduction for Alignment

Although we observed that DTW is effective when applied to KDR (or another kernel method), in our experiments the alignment is carried out in the original feature space. It is potentially better to perform warping in the projected space, as dimensionality reduction can reduce noise and capture the essence of the data. It will be interesting to formulate a framework for simultaneous subspace projection and alignment.

7.1.2 Discriminative Inference via Supervised Likelihood

Chapter 5 outlined the discriminative HPY model, where the annotations are incorporated with a constrained optimization of the variational lower bound. This approach, however, is based on optimization rather than statistical principles, since the data likelihood has been altered and a constrained minimization of the KL divergence based on unsupervised likelihood is not consistent. Alternatively, we can redefine the likelihood to include the new labels and annotations. This will also address another drawback of the DHPY model, which is its generalization power from training data to test data.

Two main approaches to discriminative training have been proposed for the *latent Dirichlet allocation (LDA)* model (8), and we have two preliminary proposals to modify our HPY graphical model in similar ways.

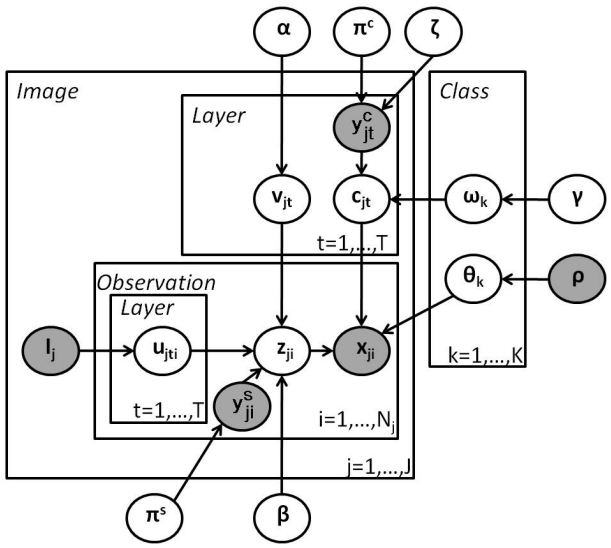


Figure 7.2. Discriminative modification #2 to the HPY model - DiscHPY

sHPY

This approach is inspired by *supervised LDA (sLDA)*, proposed by (7) and applied by (104) for image classification. It treats the labels as response variables generated from the same latent variables that generate the data. When the response variable is a categorical label, (7) suggests a generalized linear model (GLM) as the emission probability distribution of the labels. A similarly updated graphical model is shown in Fig. 7.1.

This modification, however, does not suit well for segmentation tasks. Both of (7) and (104) have response variables at the level of the document (or the image in our case). For segmentation, our response variables are the segmentation labels per superpixel; hence, we do not have a distribution of the latent states to estimate the responses.

DiscHPY

This approach is inspired by *discriminative LDA (DiscLDA)* (50). The model discriminatively transforms the latent states based on the label/class information, which can be applied at the superpixel level and is more suitable for our task of segmentation. We will present the modifications in that flavor.

The changes can be seen in Fig. 7.2. Essentially, the segment-level annotations y^s_{jt} and the class-level annotations y^c_{jt} now influence the indicator variables $\vec{z}$ and $\vec{c}$ corresponding to layer and class assignments, respectively. The modifications to the distributions are as follows:

$$
\begin{aligned}
y^s_{ji} &\in \{1,\ldots,T\} \\
y^c_{ji} &\in \{1,\ldots,K\} \\
z_{ji} &= \min\{t | u_{jti} < \beta_t^{1\{t=y^s_{ji}\}} \Phi^{-1}(v_{jt})\} \\
c_{jt}|\vec{w}, y^c_{jt}, \zeta &\sim \phi(\zeta' \tilde{\mathbf{w}}), \text{ where} \\
\zeta'_k &= \zeta_k^{1\{k \le y^c_{jt}\}}, \text{ or} \\
p(c_{jt}|\vec{w}, y^c_{jt}, \zeta) &= \prod_{k=1}^{K} (1 - w_k \zeta_k^{1\{k \le y^c_{jt}\}})^{\mathbf{1}(c_{jt}>k)} (w_k \zeta_k^{1\{k \le y^c_{jt}\}})^{\mathbf{1}(c_{jt}=k)}
\end{aligned}
$$

Note that there are many ways to define the transformation. Here we introduced a set of weights to discriminatively bias both the layer assignment decision rule and the class distribution. It can potentially correct for incorrectly specified prior hyperparameters $(\gamma_a, \gamma_b, \alpha_a, \alpha_b)$. It might, however, make the variational training more susceptible to local minima.

7.1.3 Spatial-temporal Segmentation

We have demonstrated and applied the HPY model to image segmentation. It is natural to extend the application to videos, where the segments are now spatial-temporal groupings. We can proceed in two ways:

1. Replace superpixels with spatial-temporal superpixels
2. Modify covariance function of Gaussian process in 4.4

In the first approach, one will augment the superpixelation algorithm to include temporal information. In the second, the covariance function will take the temporal information into account, similar to the dynamics kernel in section 3.2.1.

7.1.4 Shape Representation for Occlusion

In the previous chapter, we discussed a few shortcomings of our shape representation, and the primary problem stems from the inability of the contour representation and the matching mechanism to handle occlusion. Below is a set of our recommendations and remedies.

Instead of uniformly subsampling the contour coordinates, we can use corner or interest point detectors to more robustly characterize the shape. The matching algorithm itself can be made more robust as well. Currently, the k-Procrustes clustering finds the optimal alignment by searching for the best starting configuration location. It is possible to search over subsets of the shape; however, the computational cost is significantly higher. Another possibility is to use DTW in 2D to allow for nonlinear spatial distortion —in the case of occlusions, it is perceivable that the missing parts of the shape can be successfully ignored during matching.

Bibliography

[1] Y. Altun, I. Tsochantaridis, and T. Hofmann, "Hidden Markov support vector machines," in *ICML*, 2003.

[2] N. Apostoloff and A. Fitzgibbon, "Bayesian video matting using learnt image priors," in *CVPR*, 2005.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "From contours to regions: An empirical evaluation," in *CVPR*, 2009.

[4] M. S. Bartlett, "The statistical significance of canonical correlations," *Biometrika*, vol. 32, 1941.

[5] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *PAMI*, vol. 24, 2002.

[6] C. M. Bishop, "Variational learning in graphical models and neural networks," in *ICANN*, 1998.

[7] D. Blei and J. Mcauliffe, "Supervised topic models," in *NIPS*, 2007.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, 2003.

[9] G. Borgefors, "Hierarchical chamfer matching: A parametric edge matching algorithm," *PAMI*, vol. 10, 1988.

[10] A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in *CVPR*, 2007.

[11] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *PAMI*, vol. 26, 2004.

[12] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *PAMI*, vol. 23, 1999.

[13] J. Carreira and C. Sminchisescu, "Constrained parametric min-cuts for automatic object segmentation," in *CVPR*, 2010.

[14] M. Carreira-Perpinan and Z. Lu, "Dimensionality reduction by unsupervised regression," in *CVPR*, 2008.

[15] B. Catanzaro, B.-Y. Su, N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer, "Efficient, high-quality image contour detection," in *ICCV*, 2009.

[16] T. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *PAMI*, vol. 23, 2001.

[17] T. Cootes, C. Taylor, D. Cooper, and J. Graham, "Active shape models-their training and application," *CVIU*, vol. 61, 1995.

[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[19] D. Demirdjian and T. Darrell, "3-D articulated pose tracking for untethered deictic reference," in *ICMI*, 2002.

[20] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, 1959.

[21] I. Endres and D. Hoiem, "Category independent object proposals," in *ECCV*, 2010.

[22] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes (VOC) Challenge," *IJCV*, vol. 88, 2010.

[23] P. Felzenszwalb and D. Huttenlocher, "Efficient belief propagatino for early vision," *IJCV*, vol. 41, 2006.

[24] P. Felzenszwalb, D. McAllester, and D. Ramaman, "A discriminatively trained, multiscale, deformable part model," in *CVPR*, 2008.

[25] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, vol. 59, 2004.

[26] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *IJCV*, vol. 61, 2005.

[27] S. Ferson, F. J. Rohlf, and R. K. Koehn, "Measuring shape variation of two-dimensional outlines," *Syst. Zool.*, vol. 34, 1985.

[28] M. Fischler and R. Elschlager., "The representation and matching of pictorial structures," *IEEE Transactions on Computer*, vol. 22, 1973.

[29] R. Fisher, "The use of multiple measurements in taxonomic problems," *Annals Eugen.*, vol. 7, pp. 179–188, 1936.

[30] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of the ACM*, vol. 5, 1962.

[31] P.-E. Forssen and D. Lowe, "Shape descriptors for maximally stable extremal regions," in *ICCV*, 2007.

[32] K. Fukumizu, F. Bach, and M. Jordan, "Kernel dimension reduction in regression," *Annals of Statistics*, vol. 37, pp. 1871–1905, 2009.

[33] D. Gavrila and V. Philomin, "Real-time object recognition," in *ICCV*, 1999.

[34] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *PAMI*, vol. 6, 1984.

[35] J. Gonfaus, X. Boix, J. V. de Weijer, A. D. Bagdanov, J. Serrat, and J. Gonza'lez, "Harmony potentials for joint classification and segmentation," in *CVPR*, 2010.

[36] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEEE Proceedings on Radar and Signal Processing*, vol. 140, 1993.

[37] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *PAMI*, vol. 29, pp. 2247–2253, 2007.

[38] J. Gower, "Generalized procrustes analysis," *Psychometrika*, vol. 40, 1975.

[39] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, 1970.

[40] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Edu. Psych.*, vol. 24, 1933.

[41] G. B. Huang, V. Jain, and E. Learned-Miller, "Unsupervised joint alignment of complex images," in *ICCV*, 2007.

[42] A. Hyvärinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural Networks*, vol. 13, 2000.

[43] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural SVMs," *Machine Learning*, vol. 77, 2009.

[44] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Eng.*, vol. 82, 1960.

[45] A. Kanaujia, C. Sminchisescu, and D. Metaxas, "Spectral latent variable models for perceptual inference," in *ICCV*, 2007.

[46] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *IJCV*, vol. 1, 1988.

[47] P. Kohli, M. Pawan, K. Philip, and H. S. Torr, "p^3 & beyond: Solving energies with higher order cliques," in *CVPR*, 2007.

[48] H. W. Kuhn, "The Hungarian method for the assignment problem," in *50 Years of Integer Programming 1958-2008*, 2010, pp. 29–47.

[49] M. P. Kumar, P. H. S. Torr, and A. Zisserman, "OBJ CUT," in *CVPR*, 2005.

[50] S. Lacoste-julien, F. Sha, and M. I. Jordan, "DiscLDA: Discriminative learning for dimensionality reduction and classification," in *NIPS*, 2008.

[51] L. Ladicky, C. Russell, P. Kohli, and P. Torr, "Graph cut based inference with co-occurrence statistics," in *ECCV*, 2010.

[52] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001.

[53] N. Lawrence, "Probabilistic non-linear principal component analysis with Gaussian process latent variable models," *Journal of Machine Learning Research*, vol. 6, pp. 1783–1816, 2005.

[54] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006.

[55] V. Lempitsky, C. Rother, S. Roth, and A. Blake, "Fusion moves for markov random field optimization," *PAMI*, vol. 32, 2010.

[56] A. Levinstein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi, "TurboPixels: Fast superpixels using geometric flows," *PAMI*, vol. 31, 2009.

[57] K.-C. Li, "Sliced inverse regression for dimension reduction," *J. Amer. Statist. Assoc.*, vol. 86, pp. 316–327, 1991.

[58] T. Lindeberg, "Feature detection with automatic scale selection," *IJCV*, vol. 30, 1998.

[59] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein, "Graphlab: A new parallel framework for machine learning," in *UAI*, 2010.

[60] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik, "Using contours to detect and localize junctions in natural images," in *CVPR*, 2008.

[61] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *ICCV*, 2001.

[62] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *PAMI*, vol. 26, 2004.

[63] J. Matas, O. Chum, M. Urba, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *BMVC*, 2002.

[64] G. Mcneill and S. Vijayakumar, "Hierarchical procrustes matching for shape retrival," in *CVPR*, 2006.

[65] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Mullers, "Fisher discriminant analysis with kernels," *Neural Networks for Signal Processing*, pp. 41–48, 1999.

[66] "DynTex," http://old-www.cwi.nl/projects/dyntex/database.html.

[67] "Mocap," http://mocap.cs.cmu.edu.

[68] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel lattices," in *CVPR*, 2008.

[69] L. Morency, A. Quattoni, and T. Darrell, "Latent-dynamic discriminative models for continuous gesture recognition," in *CVPR*, 2007.

[70] L. Morency, A. Rahimi, and T. Darrell, "Adaptive view-based appearance model," in *CVPR*, 2003.

[71] G. Mori, X. Ren, A. Efros, and J. Malik, "Recovering human body configurations: Combining segmentation and recognition," in *CVPR*, 2004.

[72] R. M. Neal, "Slice sampling," *Annals of Statistics*, vol. 31, 2003.

[73] J. Nilsson, F. Sha, and M. Jordan, "Regression on manifolds using kernel dimension reduction," in *ICML*, 2007.

[74] P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, pp. 1373–1396, 2003.

[75] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *IJCV*, vol. 42, 2001.

[76] B. Packer, S. Gould, and D. Koller, "A unified contour-pixel model for segmentation," in *ECCV*, 2010.

[77] J. Pitman and M. Yor, "The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator," *Annals of Probability*, vol. 25, 1997.

[78] A. Quattoni, S. Wang, L. Morency, M. Collins, and T. Darrell, "Hidden conditional random fields," *PAMI*, vol. 29, no. 10, pp. 1848–1852, 2007.

[79] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[80] N. Ratliff, A. Bagnell, and M. Zinkevich, "Subgradient methods for maximum margin structured learning," in *Workshop on Learning in Structured Outputs Spaces at ICML*, 2006.

[81] X. Ren and J. Malik, "Learning a classification model for segmentation," in *ICCV*, 2003.

[82] S. Roth and M. J. Black, "On the spatial statistics of optical flow," in *ICCV*, 2005.

[83] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000. [Online]. Available: http://www.sciencemag.org/cgi/content/abstract/290/5500/2323

[84] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "LabelMe: a database and web-based tool for image annotation," *IJCV*, vol. 77, 2008.

[85] U. Schmidt, Q. Gao, and S. Roth, "A generative perspective on MRFs in low-level vision," in *CVPR*, 2010.

[86] H. Shimodaira, K.-I. Noma, M. Nakai, and S. Sagayama, "Dynamic time-alignment kernel in Support Vector Machine," in *NIPS*, 2001.

[87] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *ECCV*, 2006.

[88] A. Shyr, T. Darrell, M. I. Jordan, and R. Urtasun, "Supervised hierarchical pitman-yor process for natural scene segmentation," in *CVPR*, 2011.

[89] A. Shyr, R. Urtasun, and M. I. Jordan, "Sufficient dimensionality reduction for visual sequence classification," in *CVPR*, 2010.

[90] H. Sidenbladh, M. J. Black, and D. Fleet, "Stochastic tracking of 3d human figures using 2d image motion," in *ECCV*, 2000, pp. 702–718.

[91] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla, "Filtering using a tree-based estimator," in *ICCV*, 2003.

[92] E. Sudderth, A. Torralba, W. Freeman, and A. Willsky, "Describing visual scenes using transformed Dirichlet processes," in *NIPS*, 2005.

[93] E. Sudderth, "Graphical models for visual object recognition and tracking," Ph.D. dissertation, Massachusetts Institute of Technology, 2006.

[94] E. Sudderth and M. Jordan, "Shared segmentation of natural scenes using dependent Pitman-Yor processes," in *NIPS*, 2008.

[95] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin, "Learning structured prediction models: a large margin approach," in *ICML*, 2005.

[96] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical Dirichlet processes," *Journal of the American Statistical Association*, vol. 101, 2004.

[97] J. Tenenbaum, V. Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000. [Online]. Available: http://www.sciencemag.org/cgi/content/abstract/290/5500/2319

[98] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Stat. Soc.*, vol. 61, 1999.

[99] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *ICML*, 2004.

[100] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, pp. 71–86, 1991.

[101] R. Unnikrishnan, C. Pantofaru, and M. Hebert, "Toward objective evaluation of image segmentation algorithms," *PAMI*, vol. 29, 2007.

[102] R. Urtasun and T. Darrell, "Discriminative Gaussian process latent variable models for classification," in *ICML*, 2007.

[103] R. Urtasun, D. Fleet, A. Geiger, J. Popovic, T. Darrell, and N. D. Lawrence, "Topologically-constrained latent variable models," in *ICML*, 2008.

[104] C. Wang, D. Blei, and L. Fei-Fei, "Simultaneous image classification and annotation," in *CVPR*, 2009.

[105] J. M. Wang, D. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 283–298, 2008.

[106] S. Wang, A. Quattoni, L. Morency, D. Demirdjian, and T. Darrell, "Hidden conditional random fields for gesture recognition," in *CVPR*, 2006.

[107] S. Warshall, "A theorem on boolean matrices," *Journal of the ACM*, vol. 9, 1962.

[108] Y. Weiss, "Deriving intrinsic images from image sequences," in *ICCV*, 2001.

[109] Y. Zhao, S. C. Zhu, and S. Luo, "Co3 for ultra-fast and accurate interactive segmentation," in *ACM Multimedea*, 2010.

Appendix A

Derivations of the variational update rules for the HPY model

The following detailed derivations of the update rules can serve as the technical report for (94).

A.1 Model Assumptions

For completeness, the distributions assumed by the model in Fig.4.2 are explicitly restated here:

$$
\begin{aligned}
w_k|\gamma_a,\gamma_b &\sim Beta(1-\gamma_a,\gamma_b+k\gamma_a)\\
\theta_k|\rho &= (\theta_k^t,\theta_k^c)\\
\theta_k^t|\rho^t &\sim Dir(\rho^t)\\
\theta_k^c|\rho^c &\sim Dir(\rho^c)\\
v_{jt}|\alpha_a,\alpha_b &\sim Beta(1-\alpha_a,\alpha_b+t\alpha_a), \text{or}\\
\bar{v}_{jt}|\alpha_a,\alpha_b &\sim \mathcal{N}(v_{jt}|0,1)\cdot Beta(\bar{v}_{jt}|1-\alpha_a,\alpha_b+t\alpha_a)\\
c_{jt}|\vec{w} &\sim \phi\\
\mathbf{u_{jt}} &\sim \mathcal{N}(\mathbf{0},\mathbf{W_j})\\
z_{ji} &= \min\{t|u_{jti}<\Phi^{-1}(v_{jt})\}\\
x_{ji}^t|\vec{u_{ji}},\vec{v_j},\vec{c_j},\vec{\theta^t} &\sim Mult(x_{ji}^t|\theta^t_{c_{jz_{ji}}}),
\end{aligned}
$$

or, explicitly,

$$
\begin{aligned}
p(w_k|\gamma) &= w_k^{-\gamma_a}(1-w_k)^{\gamma_b+k\gamma_a-1} \\
p(\theta_k|\rho) &\propto \prod_{i=1}^{M}\theta_{ki}^{\rho_i-1} \\
p(v_{jt}|\alpha) &= v_{jt}^{-\alpha_a}(1-v_{jt})^{\alpha_b+t\alpha_a-1} \\
p(c_{jt}|\vec{w}) &= \prod_{k=1}^{K}(1-w_k)^{\mathbf{1}(c_{jt}>k)}w_k^{\mathbf{1}(c_{jt}=k)} \\
p(z_{ji}|u_{j:i},v_j) &= \prod_{t=1}^{T}p(u_{jti}\geq\Phi^{-1}(v_{jt}))^{\mathbf{1}(z_{ji}>t)}p(u_{jti}<\Phi^{-1}(v_{jt}))^{\mathbf{1}(z_{ji}=t)} \\
p(x_{ji}|z_{ji},\vec{c_j},\vec{\theta}) &\propto \prod_{k=1}^{K}(\theta_k^t)^{x_{jik}^t}(\theta_k^c)^{x_{jik}^c}.
\end{aligned}
$$

The variational posteriors are assumed to take the following forms

$$
\begin{aligned}
q(\theta_k|\eta_k) &= Dir(\eta_k) \\
q(\vec{c_j}|\vec{\kappa_j}) &= Mult(\vec{c_j}|\vec{\kappa_j}) \\
q(w_k|\omega_{k,a},\omega_{k,b}) &= Beta(w_k|\omega_{k,a},\omega_{k,b}) \\
q(\bar{v}_{jt}|\nu_{jt},\delta_{jt}) &= N(\bar{v}_{jt}|\nu_{jt},\delta_{jt}) \\
q(u_{jti}|\mu_{jti},\lambda_{jti}) &= N(u_{jti}|\mu_{jti},\lambda_{jti}).
\end{aligned}
$$

A.2 Variational lower bound

Combining all of the above, the variational lower bound becomes

$$
\begin{aligned}
\log p(\mathbf{x}|\alpha,\gamma,\rho) &\geq \mathbb{E}_q[\log p(\mathbf{x},\mathbf{z},\mathbf{u},\mathbf{v},\mathbf{c},\mathbf{w},\theta|\alpha,\gamma,\rho)] - \mathbb{E}_q[\log q(\mathbf{u},\mathbf{v},\mathbf{c},\mathbf{w},\theta)] \\
&= \sum_{j=1}^{J}(\sum_{i=1}^{N_j}\mathbb{E}_q[\log p(x_{ji}|\vec{u_{ji}},\vec{v_j},\vec{c_j},\vec{\theta})] + \sum_{t=1}^{T}\mathbb{E}_q[\log p(u_{jti})]) + \\
&\quad \sum_{t=1}^{T}\mathbb{E}_q[\log p(c_{jt}|\vec{w})] + \mathbb{E}_q[\log p(v_{jt}|\alpha)] + \sum_{k=1}^{K}\mathbb{E}_q[\log p(\theta_k|\rho)] + \mathbb{E}_q[\log p(w_k|\gamma)] - \\
&\quad \sum_{j=1}^{J}\sum_{t=1}^{T}\mathbb{E}_q[\log q(v_{jt}|\nu_{jt})] - \mathbb{E}_q[\log q(c_{jt}|\kappa_{jt})] - \sum_{i=1}^{N_j}\mathbb{E}_q[\log q(u_{jti}|\mu_{jti})] - \\
&\quad \sum_{k=1}^{K}\mathbb{E}_q[\log q(w_k|\omega_k)] - \mathbb{E}_q[\log q(\theta_k|\eta_k)]
\end{aligned}
$$

Drilling down to each expression,

$$
\begin{aligned}
\mathbb{E}_q[\log p(x_{ji}|\vec{u_{ji}},\vec{v_j},\vec{c_j},\vec{\theta})] &= \mathbb{E}_q[\log \prod_l \theta_{c_{jz_{ji}},l}^{x_{ji,l}}] \\
&= \sum_l x_{ji,l}\mathbb{E}_q \log \theta_{c_{jz_{ji}},l} \\
&= \sum_l x_{ji,l}\sum_{t=1}^{T}\mathbb{E}_q[\log \theta_{c_{jt},l}\mathbf{1}(z_{ji}=t)] \\
&= \sum_l x_{ji,l}\sum_{t=1}^{T}\sum_{k=1}^{K}\mathbb{E}_q[\log \theta_{k,l}\mathbf{1}(z_{ji}=t)\mathbf{1}(c_{jt}=k)] \\
&= \sum_l x_{ji,l}\sum_{t=1}^{T}\sum_{k=1}^{K}(\mathbb{E}_q \log \theta_{k,l})\mathbb{P}_{\shortparallel}(z_{ji}=t)\kappa_{jtk} \\
&= \sum_l x_{ji,l}\sum_{t=1}^{T}\sum_{k=1}^{K}(\Psi(\eta_{k,l})-\Psi(\sum_l \eta_{k,l}))\mathbb{P}_{\shortparallel}(u_{j\tau i}\geq \bar{v_{j\tau}}\forall\tau<t, u_{jti}<\bar{v_{jt}})\kappa_{jtk} \\
&= \sum_l x_{ji,l}\sum_{t=1}^{T}\sum_{k=1}^{K}(\Psi(\eta_{k,l})-\Psi(\sum_l \eta_{k,l}))\kappa_{jtk}\Phi(\frac{\nu_{jt}-\mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}})\prod_{\tau=1}^{t-1}(1-\Phi(\frac{\nu_{j\tau}-\mu_{j\tau i}}{\sqrt{\delta_{j\tau}+\lambda_{j\tau i}}})), \\
\mathbb{E}_q[\log p(\vec{u_{jt}})] &= \mathbb{E}_q[\log \frac{1}{(2\pi)^{\frac{D}{2}}|W_j|^{\frac{1}{2}}}e^{-\frac{1}{2}\vec{u_{jt}}^T W_j^{-1}\vec{u_{jt}}}] \\
&= -\frac{1}{2}\mathbb{E}_q[\vec{u_{jt}}^T W_j^{-1}\vec{u_{jt}}] \\
&= -\frac{1}{2}Tr[W_j^{-1}diag(\vec{\lambda_{jt}})]-\frac{1}{2}\vec{\mu_{jt}}^T W_j^{-1}\vec{\mu_{jt}} \\
\mathbb{E}_q[\log p(c_{jt}|\vec{w})] &= \mathbb{E}_q[\log \prod_{k=1}^{K}(1-w_k)^{\mathbf{1}(k<c_{jt})}w_k^{\mathbf{1}(k=c_{jt})}] \\
&= \sum_{k=1}^{K}\mathbb{E}_q[\mathbf{1}(k<c_{jt})\log(1-w_k)]+\mathbb{E}_q[\mathbf{1}(k=c_{jt})\log(w_k)] \\
&= \sum_{k=1}^{K}\kappa_{jtk}(\Psi(\omega_{k,a})-\Psi(\omega_{k,a}+\omega_{k,b}))+\sum_{k'=k+1}^{K}\kappa_{jtk'}(\Psi(\omega_{k,b})-\Psi(\omega_{k,a}+\omega_{k,b})) \\
\mathbb{E}_q[\log p(\bar{v_{jt}}|\alpha)] &= \mathbb{E}_q[\log \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}\bar{v_{jt}}^2}\frac{\Gamma(1-\alpha_a+\alpha_b+t\alpha_a)}{\Gamma(1-\alpha_a)\Gamma(\alpha_b+t\alpha_a)}\Phi(\bar{v_{jt}})^{-\alpha_a}(1-\Phi(\bar{v_{jt}}))^{\alpha_b+t\alpha_a-1} \\
&= -\frac{1}{2}\mathbb{E}_q[\bar{v_{jt}}^2]+\log\Gamma(1-\alpha_a+\alpha_b+t\alpha_a)-\log\Gamma(1-\alpha_a)-\log\Gamma(\alpha_b+t\alpha_a) \\
&\quad -\alpha_a\mathbb{E}_q[\log\Phi((\bar{v_{jt}}))]+(\alpha_b+t\alpha_a-1)\mathbb{E}_q[\log\Phi((1-\bar{v_{jt}}))] \\
&\geq -\frac{1}{2}(\delta_{jt}+\nu_{jt}^2)+\log\Gamma(1-\alpha_a+\alpha_b+t\alpha_a)-\log\Gamma(1-\alpha_a)-\log\Gamma(\alpha_b+t\alpha_a) \\
&\quad -\alpha_a(\log\Phi(\frac{\nu_{jt}}{\sqrt{1+\delta_{jt}}}))+(\alpha_b+t\alpha_a-1)(\log(1-\Phi(\frac{\nu_{jt}}{\sqrt{1+\delta_{jt}}}))) \\
&\geq -\frac{1}{2}(\delta_{jt}+\nu_{jt}^2)+\log\Gamma(1-\alpha_a+\alpha_b+t\alpha_a)-\log\Gamma(1-\alpha_a)-\log\Gamma(\alpha_b+t\alpha_a) \\
&\quad -\alpha_a(\log\Phi(\frac{\nu_{jt}}{\sqrt{1+\delta_{jt}}}))+(\alpha_b+t\alpha_a-1)(\log\Phi(\frac{-\nu_{jt}}{\sqrt{1+\delta_{jt}}}))
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}_q[\log p(\theta_k|\rho)] &= \mathbb{E}_q[\log \frac{\Gamma(\sum_l \rho_l)}{\prod_l \Gamma(\rho_l)} \prod_l \theta_{k,l}^{\rho_l - 1}] \\
&= \log\Gamma(\sum_l \rho_l) - \sum_l \log\Gamma(\rho_l) + \sum_l (\rho_l - 1)\mathbb{E}_q[\log \theta_{k,l}] \\
&= \log\Gamma(\sum_l \rho_l) - \sum_l \log\Gamma(\rho_l) + \sum_l (\rho_l - 1)(\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l})) \\
\mathbb{E}_q[\log p(w_k|\gamma)] &= \mathbb{E}_q[\log \frac{\Gamma(1 - \gamma_a + \gamma_b + k\gamma_a)}{\Gamma(1-\gamma_a)\Gamma(\gamma_b + k\gamma_a)} w_k^{-\gamma_a}(1 - w_k)^{\gamma_b + k\gamma_a - 1}] \\
&= \log\Gamma(1 - \gamma_a + \gamma_b + k\gamma_a) - \log\Gamma(1-\gamma_a) - \log\Gamma(\gamma_b + k\gamma_a) \\
&\quad -\gamma_a \mathbb{E}_q[\log w_k] + (\gamma_b + k\gamma_a - 1)\mathbb{E}_q[\log(1 - w_k)] \\
&= \log\Gamma(1 - \gamma_a + \gamma_b + k\gamma_a) - \log\Gamma(1-\gamma_a) - \log\Gamma(\gamma_b + k\gamma_a) \\
&\quad -\gamma_a(\Psi(\omega_{k,a}) - \Psi(\omega_{k,a} + \omega_{k,b})) + (\gamma_b + k\gamma_a - 1)(\Psi(\omega_{k,b}) - \Psi(\omega_{k,a} + \omega_{k,b})) \\
\mathbb{E}_q[\log q(\bar{v_{jt}}|\nu_{jt}, \delta_{jt})] &= \mathbb{E}_q[\log \frac{1}{\sqrt{2\pi\delta_{jt}}} e^{-\frac{(\bar{v_{jt}} - \nu_{jt})^2}{2\delta_{jt}}}] \\
&= -\frac{1}{2}\log\delta_{jt} - \frac{1}{2}\mathbb{E}_q[\frac{(\bar{v_{jt}} - \nu_{jt})^2}{\delta_{jt}}] \\
&= -\frac{1}{2}\log\delta_{jt} - \frac{1}{2} \\
\mathbb{E}_q[\log q(c_{jt}|\kappa_{jt})] &= \mathbb{E}_q[\log \prod_{k=1}^K \kappa_{jtk}^{c_{jtk}}] \\
&= \sum_{k=1}^K \mathbb{E}_q[c_{jtk}] \log \kappa_{jtk} \\
&= \sum_{k=1}^K \kappa_{jtk} \log \kappa_{jtk} \\
\mathbb{E}_q[\log q(u_{jti}|\mu_{jti}, \lambda_{jti})] &= \mathbb{E}_q[\log \frac{1}{\sqrt{2\pi\lambda_{jti}}} e^{-\frac{(u_{jti} - \mu_{jti})^2}{2\lambda_{jti}}}] \\
&= -\frac{1}{2}\log(\lambda_{jti}) - \frac{1}{2}\mathbb{E}_q[\frac{(u_{jti} - \mu_{jti})^2}{2\lambda_{jti}}] + C \\
&= -\frac{1}{2}\log(\lambda_{jti}) - \frac{1}{2} + C, \\
\mathbb{E}_q[\log q(w_k|\omega_k)] &= \mathbb{E}_q[\log \frac{\Gamma(\omega_{k,a} + \omega_{k,b})}{\Gamma(\omega_{k,a})\Gamma(\omega_{k,b})} w_k^{\omega_a - 1}(1 - w_k)^{\omega_b - 1}] \\
&= \log\Gamma(\omega_{k,a} + \omega_{k,b}) - \log\Gamma(\omega_{k,a}) - \log\Gamma(\omega_{k,b}) + (\omega_a - 1)\mathbb{E}_q[\log w_k] + (\omega_b - 1)\mathbb{E}_q[\log(1 - \\
&= \log\Gamma(\omega_{k,a} + \omega_{k,b}) - \log\Gamma(\omega_{k,a}) - \log\Gamma(\omega_{k,b}) \\
&\quad +(\omega_{k,a} - 1)(\Psi(\omega_{k,a}) - \Psi(\omega_{k,a} + \omega_{k,b})) + (\omega_{k,b} - 1)(\Psi(\omega_{k,b}) - \Psi(\omega_{k,a} + \omega_{k,b})) \\
\mathbb{E}_q[\log q(\theta_k|\eta_k)] &= \mathbb{E}_q[\log \frac{\Gamma(\sum_l \eta_{k,l})}{\prod_l \Gamma(\eta_{k,l})} \prod_l \theta_{k,l}^{\eta_{k,l} - 1}] \\
&= \log\Gamma(\sum_l \eta_{k,l}) - \sum_l \log\Gamma(\eta_{k,l}) + \sum_l (\eta_{k,l} - 1)\mathbb{E}_q[\log \theta_{k,l}] \\
&= \log\Gamma(\sum_l \eta_{k,l}) - \sum_l \log\Gamma(\eta_{k,l}) + \sum_l (\eta_{k,l} - 1)(\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l}))
\end{aligned}
$$

A.3 Optimization

For each of the variational prior parameters $\eta_{kl}, \kappa_{jtk}, \omega_{k,a|b}, \nu_{jt}, \delta_{jt}, \mu_{jti}, \lambda_{jti}$, we derive the updates here. Denote the lower bound $\mathbf{L} \equiv \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{v}, \mathbf{c}, \mathbf{w}, \theta|\alpha, \gamma, \rho)] - \mathbb{E}_q[\log q(\mathbf{u}, \mathbf{v}, \mathbf{c}, \mathbf{w}, \theta)]$. Note that $\frac{\partial \Phi(x)}{\partial x} = \frac{1}{\sqrt{2\pi}} e^{-x^2}$.

$$
\begin{aligned}
\frac{\partial \mathbf{L}}{\partial \eta_{kl}} &= \frac{\partial \sum_j \sum_i \mathbb{E}_q[\log p(x_{ji}|\vec{u_{ji}}, \vec{v_j}, \vec{c_j}, \vec{\theta})] + \mathbb{E}_q[\log p(\theta_k|\rho)] - \mathbb{E}_q[\log q(\theta_k|\eta_k)]}{\partial \eta_{kl}} \\
&= \sum_j \sum_i x_{ji,l} \sum_{t=1}^{T} \kappa_{jtk} \Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt} + \lambda_{jti}}}) \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau} + \lambda_{j\tau i}}})) \frac{\partial(\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l}))}{\partial \eta_{kl}} + (\rho_l - 1) \frac{\partial} \\
&\quad -\Psi(\sum_l \eta_{k,l}) + \Psi(\eta_{k,l}) - \Psi(\eta_{k,l}) + \Psi(\sum_l \eta_{k,l}) - (\eta_{k,l} - 1) \frac{\partial(\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l}))}{\partial \eta_{kl}} \\
&= \sum_j \sum_i x_{ji,l} \sum_{t=1}^{T} \kappa_{jtk} \Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt} + \lambda_{jti}}}) \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau} + \lambda_{j\tau i}}})) \frac{\partial(\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l}))}{\partial \eta_{kl}} \\
&\quad +(\rho_l - \eta_{k,l}) \frac{\partial(\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l}))}{\partial \eta_{kl}} \\
&= 0 \qquad \text{(optimality condition)} \\
\Rightarrow \quad \eta^*_{kl} &= \rho_l + \sum_j \sum_i x_{ji,l} \sum_{t=1}^{T} \kappa_{jtk} \Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt} + \lambda_{jti}}}) \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau} + \lambda_{j\tau i}}})) \\
\frac{\partial \mathbf{L}}{\partial \kappa_{jtk}} &= \frac{\partial \sum_i \mathbb{E}_q[\log p(x_{ji}|\vec{u_{ji}}, \vec{v_j}, \vec{c_j}, \vec{\theta})] + \mathbb{E}_q[\log p(c_{jt}|\vec{w})] - \mathbb{E}_q[\log q(c_{jt}|\kappa_{jt})]}{\partial \kappa_{jtk}} \\
&= \sum_i \sum_l x_{ji,l} (\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l})) \Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt} + \lambda_{jti}}}) \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau} + \lambda_{j\tau i}}})) \\
&\quad +\Psi(\omega_{k,a}) - \Psi(\omega_{k,a} + \omega_{k,b}) + \sum_{k'=1}^{K} \mathbf{1}(k' < k)(\Psi(\omega_{k',b}) - \Psi(\omega_{k',a} + \omega_{k',b})) - \log \kappa_{jtk} - 1 \\
&= \sum_i \sum_l x_{ji,l} (\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l})) \Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt} + \lambda_{jti}}}) \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau} + \lambda_{j\tau i}}})) \\
&\quad +\Psi(\omega_{k,a}) - \Psi(\omega_{k,a} + \omega_{k,b}) + \sum_{k'=1}^{k-1} (\Psi(\omega_{k',b}) - \Psi(\omega_{k',a} + \omega_{k',b})) - \log \kappa_{jtk} - 1 \\
&= 0 \qquad \text{(optimality condition)} \\
\Rightarrow \quad \kappa^*_{jtk} &\propto \exp\{\sum_i \sum_l x_{ji,l} (\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l})) \Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt} + \lambda_{jti}}}) \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau} + \lambda_{j\tau i}}})) \\
&\quad +\Psi(\omega_{k,a}) - \Psi(\omega_{k,a} + \omega_{k,b}) + \sum_{k'=1}^{k-1} (\Psi(\omega_{k',b}) - \Psi(\omega_{k',a} + \omega_{k',b}))\}
\end{aligned}
$$

$$\frac{\partial \mathbf{L}}{\partial \omega_{ka}} = \frac{\partial \sum_j \sum_t \mathbb{E}_q[\log p(c_{jt}|\vec{w})] + \mathbb{E}_q[\log p(w_k|\gamma)] - \mathbb{E}_q[\log q(w_k|\omega_k)]}{\partial \omega_{ka}}$$
$$=$$

$$\frac{\partial \mathbf{L}}{\partial \nu_{jt}} = \frac{\partial \sum_i \mathbb{E}_q[\log p(x_{ji}|\vec{u_{ji}}, \vec{v_j}, \vec{c_j}, \vec{\theta})] + \mathbb{E}_q[\log p(\vec{v_{jt}}|\alpha)] - \mathbb{E}_q[\log q(\vec{v_{jt}}|\nu_{jt}, \delta_{jt})]}{\partial \nu_{jt}}$$
$$= \sum_i \sum_l x_{ji,l} \sum_{k=1}^{K} (\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l})) \kappa_{jtk} \{ \frac{\exp\{-0.5(\frac{\nu_{jt}-\mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}})^2\}}{\sqrt{2\pi(\delta_{jt}+\lambda_{jti})}} \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau}+\lambda_{j\tau i}}})) -$$
$$\frac{\exp\{-0.5(\frac{\nu_{jt}-\mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}})^2\}}{\sqrt{2\pi(\delta_{jt}+\lambda_{jti})}} \sum_{t'=t+1}^{T} \Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}}) \prod_{\tau=1,\tau\neq t}^{t'-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau}+\lambda_{j\tau i}}}))\} -$$
$$\nu_{jt} - \frac{\alpha_a}{\Phi(\frac{\nu_{jt}}{\sqrt{1+\delta_{jt}}})} \frac{\exp\{-0.5(\frac{\nu_{jt}}{\sqrt{1+\delta_{jt}}})^2\}}{\sqrt{1+\delta_{jt}}} - \frac{\alpha_b + t\alpha_a - 1}{\Phi(\frac{-\nu_{jt}}{\sqrt{1+\delta_{jt}}})} \frac{\exp\{-0.5(\frac{\nu_{jt}}{\sqrt{1+\delta_{jt}}})^2\}}{\sqrt{1+\delta_{jt}}}$$

$$\frac{\partial \mathbf{L}}{\partial \delta_{jt}} = \frac{\partial \sum_i \mathbb{E}_q[\log p(x_{ji}|\vec{u_{ji}}, \vec{v_j}, \vec{c_j}, \vec{\theta})] + \mathbb{E}_q[\log p(\vec{v_{jt}}|\alpha)] - \mathbb{E}_q[\log q(\vec{v_{jt}}|\nu_{jt}, \delta_{jt})]}{\partial \delta_{jt}}$$
$$= \sum_i \sum_l x_{ji,l} \sum_{k=1}^{K} (\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l})) \kappa_{jtk} \{ \frac{-(\nu_{jt} - \mu_{jti}) \exp\{-0.5(\frac{\nu_{jt}-\mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}})^2\}}{2\sqrt{2\pi}(\delta_{jt}+\lambda_{jti})^{\frac{3}{2}}} \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau}+\lambda_{j\tau i}}})$$
$$\frac{-(\nu_{jt} - \mu_{jti}) \exp\{-0.5(\frac{\nu_{jt}-\mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}})^2\}}{2\sqrt{2\pi}(\delta_{jt}+\lambda_{jti})^{\frac{3}{2}}} \sum_{t'=t+1}^{T} \Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}}) \prod_{\tau=1,\tau\neq t}^{t'-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau}+\lambda_{j\tau i}}}))\} -$$
$$\frac{1}{2} + \frac{\alpha_a}{\Phi(\frac{\nu_{jt}}{\sqrt{1+\delta_{jt}}})} \frac{\nu_{jt} \exp\{-(\frac{\nu_{jt}}{\sqrt{1+\delta_{jt}}})^2\}}{2(1+\delta_{jt})^{\frac{3}{2}}} + \frac{\alpha_b + t\alpha_a - 1}{\Phi(\frac{-\nu_{jt}}{\sqrt{1+\delta_{jt}}})} \frac{\nu_{jt} \exp\{-(\frac{\nu_{jt}}{\sqrt{1+\delta_{jt}}})^2\}}{2(1+\delta_{jt})^{\frac{3}{2}}} + \frac{1}{2\delta_{jt}}$$

$$\frac{\partial \mathbf{L}}{\partial \mu_{jti}} = \frac{\partial \mathbb{E}_q[\log p(x_{ji}|\vec{u_{ji}}, \vec{v_j}, \vec{c_j}, \vec{\theta})] + \mathbb{E}_q[\log p(\vec{u_{jt}})] - \mathbb{E}_q[\log q(u_{jti}|\mu_{jti}, \lambda_{jti})]}{\partial \mu_{jti}}$$
$$= \sum_l x_{ji,l} \sum_{k=1}^{K} (\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l})) \kappa_{jtk} \{ \frac{-\exp\{-0.5(\frac{\nu_{jt}-\mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}})^2\}}{\sqrt{2\pi(\delta_{jt}+\lambda_{jti})}} \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau}+\lambda_{j\tau i}}})) -$$
$$\frac{-\exp\{-0.5(\frac{\nu_{jt}-\mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}})^2\}}{\sqrt{2\pi(\delta_{jt}+\lambda_{jti})}} \sum_{t'=t+1}^{T} \Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}}) \prod_{\tau=1,\tau\neq t}^{t'-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau}+\lambda_{j\tau i}}}))\} - (W_j^{-1}\vec{\mu_{jt}})_i$$

$$\frac{\partial \mathbf{L}}{\partial \lambda_{jti}} = \frac{\partial \mathbb{E}_q[\log p(x_{ji}|\vec{u_{ji}}, \vec{v_j}, \vec{c_j}, \vec{\theta})] + \mathbb{E}_q[\log p(\vec{u_{jt}})] - \mathbb{E}_q[\log q(u_{jti}|\mu_{jti}, \lambda_{jti})]}{\partial \lambda_{jti}}$$
$$= \sum_i \sum_l x_{ji,l} \sum_{k=1}^{K} (\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l})) \kappa_{jtk} \{ \frac{-(\nu_{jt} - \mu_{jti}) \exp\{-0.5(\frac{\nu_{jt}-\mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}})^2\}}{2\sqrt{2\pi}(\delta_{jt}+\lambda_{jti})^{\frac{3}{2}}} \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau}+\lambda_{j\tau i}}})$$
$$\frac{-(\nu_{jt} - \mu_{jti}) \exp\{-0.5(\frac{\nu_{jt}-\mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}})^2\}}{2\sqrt{2\pi}(\delta_{jt}+\lambda_{jti})^{\frac{3}{2}}} \sum_{t'=t+1}^{T} \Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt}+\lambda_{jti}}}) \prod_{\tau=1,\tau\neq t}^{t'-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau}+\lambda_{j\tau i}}}))\} -$$
$$\frac{1}{2} diag(W_j^{-1}) + \frac{1}{2\lambda_{jt}}$$

Appendix B

Derivations of the update rules for the HPY model with Shape prior

We now derive the variational lower bound and the update rules, as before. Note that the parts of the expression involving variational posterior have not changed (ie, those that include $q(\cdot|\cdot)$).

$$
\begin{aligned}
\log p(\mathbf{x}|\alpha,\gamma,\rho) &\geq \mathbb{E}_q[\log p(\mathbf{x},\mathbf{z},\mathbf{u},\mathbf{v},\mathbf{c},\mathbf{w},\theta|\alpha,\gamma,\rho)] - \mathbb{E}_q[\log q(\mathbf{u},\mathbf{v},\mathbf{c},\mathbf{w},\theta)] \\
&= \sum_{j=1}^{J}(\sum_{i=1}^{N_j}\mathbb{E}_q[\log p(x_{ji}|\vec{u_{ji}},\vec{v_j},\vec{c_j},\vec{\theta})] + \sum_{t=1}^{T}\mathbb{E}_q[\log p(u_{jti}|I_j,a_{jt},c_{jt},\vec{S_k})]) + \\
&\quad \sum_{t=1}^{T}\mathbb{E}_q[\log p(c_{jt}|\vec{w})] + \mathbb{E}_q[\log p(v_{jt}|\alpha)] + \sum_{k=1}^{K}\mathbb{E}_q[\log p(\theta_k|\rho)] + \mathbb{E}_q[\log p(w_k|\gamma)] - \\
&\quad \sum_{j=1}^{J}\sum_{t=1}^{T}\mathbb{E}_q[\log q(v_{jt}|\nu_{jt})] - \mathbb{E}_q[\log q(c_{jt}|\kappa_{jt})] - \sum_{i=1}^{N_j}\mathbb{E}_q[\log q(u_{jti}|\mu_{jti})] - \\
&\quad \sum_{k=1}^{K}\mathbb{E}_q[\log q(w_k|\omega_k)] - \mathbb{E}_q[\log q(\theta_k|\eta_k)]
\end{aligned}
$$

The only expressions that are different than before are $\mathbb{E}_q[\log p(x_{ji}|\vec{u_{ji}},\vec{v_j},\vec{c_j},\vec{\theta})]$ and $\mathbb{E}_q[\log p(u_{jti}|I_j,a_{jt},c_{jt},\vec{S_k})]$.

$$
\begin{aligned}
& \mathbb{E}_q[\log p(x_{ji}|\vec{u_{ji}}, \vec{v_j}, \vec{c_j}, \vec{\theta})] \\
= \;& \mathbb{E}_q[\log \prod_l \theta_{c_{jz_{ji}},l}^{x_{ji,l}}] \\
= \;& \sum_l x_{ji,l} \mathbb{E}_q \log \theta_{c_{jz_{ji}},l} \\
= \;& \sum_l x_{ji,l} \sum_{t=1}^{T} \mathbb{E}_q[\log \theta_{c_{jt},l} \mathbf{1}(z_{ji} = t)] \\
= \;& \sum_l x_{ji,l} \sum_{t=1}^{T} \sum_{k=1}^{K} \mathbb{E}_q[\log \theta_{k,l} \mathbf{1}(z_{ji} = t)\mathbf{1}(c_{jt} = k)] \\
= \;& \sum_l x_{ji,l} \sum_{t=1}^{T} \sum_{k=1}^{K} (\mathbb{E}_q \log \theta_{k,l}) \mathbb{P}_{\shortparallel}(z_{ji} = t) \kappa_{jtk} \\
= \;& \sum_l x_{ji,l} \sum_{t=1}^{T} \sum_{k=1}^{K} (\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l})) \mathbb{P}_{\shortparallel}(u_{j\tau i} - F(\mathbf{S_k}; a_{j\tau})_i \geq \bar{v_{j\tau}} \forall \tau < t, u_{jti} - F(\mathbf{S_k}; a_{jt}) < \bar{v_{jt}}) \kappa_{jtk} \\
= \;& \sum_l x_{ji,l} \sum_{t=1}^{T} \sum_{k=1}^{K} (\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l})) \kappa_{jtk} \Phi(\frac{\nu_{jt} - \mu_{jti} + F(\mathbf{S_k}; a_{jt})_i}{\sqrt{\delta_{jt} + \lambda_{jti}}}) \prod_{\tau=1}^{t-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i} + F(\mathbf{S_k}; a_{j\tau})_i}{\sqrt{\delta_{j\tau} + \lambda_{j\tau i}}})), \\
& \mathbb{E}_q[\log p(u_{jti}|I_j, a_{jt}, c_{jt}, \vec{S_k})] \\
= \;& \mathbb{E}_q[\log \frac{1}{(2\pi)^{\frac{D}{2}} |W_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{u_{jt}} - F(\mathbf{S_{c_{jt}}}; a_{jt}))^T W_j^{-1} (\vec{u_{jt}} - F(\mathbf{S_{c_{jt}}}; a_{jt}))}] \\
= \;& \mathbb{E}_q[-\frac{1}{2} \vec{u_{jt}}^T W_j^{-1} \vec{u_{jt}} + F(\mathbf{S_{c_{jt}}}; a_{jt})^T W_j^{-1} \vec{u_{jt}} - \frac{1}{2} F(\mathbf{S_{c_{jt}}}; a_{jt})^T W_j^{-1} F(\mathbf{S_{c_{jt}}}; a_{jt})] \\
= \;& \mathbb{E}_q[-\frac{1}{2} \vec{u_{jt}}^T W_j^{-1} \vec{u_{jt}} + \sum_{k=1}^{K} \mathbf{1}(c_{jt} = k) F(\mathbf{S_k}; a_{jt})^T W_j^{-1} \vec{u_{jt}} - \frac{1}{2} \mathbf{1}(c_{jt} = k) F(\mathbf{S_k}; a_{jt})^T W_j^{-1} F(\mathbf{S_k}; a_{jt})] \\
= \;& -\frac{1}{2} Tr[W_j^{-1} diag(\vec{\lambda_{jt}})] - \frac{1}{2} \vec{\mu_{jt}}^T W_j^{-1} \vec{\mu_{jt}} + \sum_{k=1}^{K} \kappa_{jtk} (F(\mathbf{S_k}; a_{jt})^T W_j^{-1} \vec{\mu_{jt}} - \frac{1}{2} F(\mathbf{S_k}; a_{jt})^T W_j^{-1} F(\mathbf{S_k}, a_{jt}))
\end{aligned}
$$

Notice that $\mathbb{E}_q[\log p(u_{jti}|I_j, a_{jt}, c_{jt}, \vec{S_k})]$ now depends on the class assignment for a layer (κ). Most of the update rules can be easily modified by replacing $\Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt} + \lambda_{jti}}})$ with $\Phi(\frac{\nu_{jt} - \mu_{jti} + F(\mathbf{S_{c_{jt}}}; a_{jt})_i}{\sqrt{\delta_{jt} + \lambda_{jti}}})$. The only ones with more modification are the rules regarding μ and κ:

$$
\begin{aligned}
\kappa^*_{jtk} \quad \propto \quad & \exp\{\sum_i \sum_l x_{ji,l}(\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l}))\Phi(\frac{\nu_{jt} - \mu_{jti}}{\sqrt{\delta_{jt} + \lambda_{jti}}}) \prod_{\tau=1}^{t-1}(1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau} + \lambda_{j\tau i}}})) \\
& + \Psi(\omega_{k,a}) - \Psi(\omega_{k,a} + \omega_{k,b}) + F(\mathbf{S_k}, a_{jt})^T W_j^{-1} \vec{\mu_{jt}} - \frac{1}{2} F(\mathbf{S_k}, a_{jt})^T W_j^{-1} F(\mathbf{S_k}, a_{jt}) \\
& + \sum_{k'=1}^{k-1} (\Psi(\omega_{k',b}) - \Psi(\omega_{k',a} + \omega_{k',b}))\}, \text{and} \\
\frac{\partial \mathbf{L}}{\partial \mu_{jti}} \quad = \quad & \sum_l x_{ji,l} \sum_{k=1}^{K} (\Psi(\eta_{k,l}) - \Psi(\sum_l \eta_{k,l}))\kappa_{jtk} \frac{-\exp\{-0.5(\frac{\nu_{jt} - \mu_{jti} + F(\mathbf{S_k}, a_{jt})_i}{\sqrt{\delta_{jt} + \lambda_{jti}}})^2\}}{\sqrt{2\pi(\delta_{jt} + \lambda_{jti})}} \\
& \{\prod_{\tau=1}^{t-1}(1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i}}{\sqrt{\delta_{j\tau} + \lambda_{j\tau i}}})) - \sum_{t'=t+1}^{T} \Phi(\frac{\nu_{jt'} - \mu_{jt'i} + F(\mathbf{S_k}, a_{jt'})_i}{\sqrt{\delta_{jt'} + \lambda_{jt'i}}}) \prod_{\tau=1, \tau \neq t}^{t'-1} (1 - \Phi(\frac{\nu_{j\tau} - \mu_{j\tau i} + F(\mathbf{S_k}, a_{j\tau})_i}{\sqrt{\delta_{j\tau} + \lambda_{j\tau i}}}) \\
& (W_j^{-1} \vec{\mu_{jt}})_i + \sum_{k=1}^{K} \kappa_{jtk} W_j^{-1} F(\mathbf{S_k}, a_{jt})
\end{aligned}
$$