Learning Discriminative Models with Incomplete Data

by

Ashish Kapoor

B. Tech., Indian Institute of Technology, Delhi (2000)
S. M., Massachusetts Institute of Technology (2002)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2006

Author .....
Program in Media Arts and Sciences,
School of Architecture and Planning
January 13, 2006

Certified by .....
Rosalind W. Picard
Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by .....
Andrew B. Lippman
Chairman
Department Committee on Graduate Students

Learning Discriminative Models with Incomplete Data

by

Ashish Kapoor

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on January 13, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Media Arts and Sciences

Abstract

Many practical problems in pattern recognition require making inferences using multiple modalities, e.g. sensor data from video, audio, physiological changes etc. Often in real-world scenarios there can be incompleteness in the training data. There can be missing channels due to sensor failures in multi-sensory data and many data points in the training set might be unlabeled. Further, instead of having exact labels we might have easy to obtain coarse labels that correlate with the task. Also, there can be labeling errors, for example human annotation can lead to incorrect labels in the training data.

The discriminative paradigm of classification aims to model the classification boundary directly by conditioning on the data points; however, discriminative models cannot easily handle incompleteness since the distribution of the observations is never explicitly modeled. We present a unified Bayesian framework that extends the discriminative paradigm to handle four different kinds of incompleteness. First, a solution based on a mixture of Gaussian processes is proposed for achieving sensor fusion under the problematic conditions of missing channels. Second, the framework addresses incompleteness resulting from partially labeled data using input dependent regularization. Third, we introduce the located hidden random field (LHRF) that learns finer level labels when only some easy to obtain coarse information is available. Finally the proposed framework can handle incorrect labels, the fourth case of incompleteness. One of the advantages of the framework is that we can use different models for different kinds of label errors, providing a way to encode prior knowledge about the process.

The proposed extensions are built on top of Gaussian process classification and result in a modular framework where each component is capable of handling different kinds of incompleteness. These modules can be combined in many different ways, resulting in many different algorithms within one unified framework. We demonstrate the effectiveness of the framework on a variety of problems such as multi-sensor affect recognition, image classification and object detection and segmentation.

Thesis Supervisor: Rosalind W. Picard
Title: Professor of Media Arts and Sciences

Learning Discriminative Models with Incomplete Data

by

Ashish Kapoor

Thesis Committee:

Advisor .....

Rosalind W. Picard
Professor of Media Arts and Sciences
Massachusetts Institute of Technology

Thesis Reader .....

Eric Horvitz
Senior Researcher
Microsoft Research

Thesis Reader .....

Joshua Tenenbaum
Paul E. Newton Assistant Professor of Computational Cognitive Science
Massachusetts Institute of Technology

Acknowledgments

I would like to thank Rosalind W. Picard, my wonderful advisor and a great source of inspiration. Thank you Roz for bringing me to MIT in the first place and for being so patient and supportive all these years. I thank Eric Horvitz and Josh Tenenbaum for being readers for this thesis. Thanks to Eric for his inspiring ideas, enthusiasm and critique about the work. Thanks to Josh for his insight, motivation and support.

I wish to thank all the folks who helped with this thesis. Special thanks to John Winn, my mentor and collaborator at Microsoft Research, Cambridge. It was John's insight and guidance that led to the work on Located Hidden Random Fields for object detection and segmentation. Thanks to Yuan (Alan) Qi for his guidance and insight with much of the work on semi-supervised learning mentioned in this thesis. Thanks to Yuri Ivanov, whose work on classifier combination inspired the work on multimodal affect recognition. Also, many thanks to Tom Minka who has tirelessly answered all my stupid and not so stupid questions on machine learning. I especially wish to thank Sumit Basu and Tanzeem Choudhury, my mentors, collaborators and friends. Thank you Sumit for the best summer internship I ever had. Thank you Tanzeem for being such a great friend and a mentor. There have been many other collaborators including Hyungil Ahn and Rana El Kaliouby whom I'd like to thank. There have been many UROPs who have burnt their midnight oil: Yue-Hann Chin, Timothy Heidel, Heather Knight, David Lafferty, Christine Lin, Louis Lopez and David Lopez Mateos. I'd also like to thank Sandy Pentland and Trevor Darrell who have inspired and motivated me with their advice, guidance and council.

Thanks to Rgrads past and present: Tim Bickmore, Win Burleson, Shani Daily, Raul Fernandez, Karen Liu, Akshay Mohan, Selene Mota, Yuan Qi, Seth Raphael, Carson Reynolds, Jay Silver and Alea Teeters. It was a great group to work with. Thanks to the friends at the Media Lab and CSAIL who have enriched my graduate student life with their insight and invaluable council. I mention a few of them here: Juan Carlos Barahona, Neal Checka, Brian Clarkson, Nathan Eagle, Mary Farbood, Tony Jebara, Aisling Kelliher, Vikram Sheel Kumar, Hyun-Yeul Lee, Andrea Lockerd Thomaz, Louis-Philippe Morency, Paul Nemirovsky, Ali Rahimi, Ben Recht, Gregory Shakhnarovich and Kevin Wilson.

Thanks to all my friends who stood by me all the time. Special thanks to the Boston mafia: Vishwanath Anantraman, Aarthi Chandrasekaran, Rupa Das, Manish Deopura, Joyce George, Vipin Gupta, Mahesh Kumar, Ashish Nimgaonkar, Anand Rajgopal, Anoop Rao, Neha Soni, Kunal Surana, Kushan Surana, Sapna Tyagi, Chintan Vaishnav, Hetal Vaishnav and Charuleka Varadharajan. I cant imagine how my life for the past six years would have been without them. Also, I was lucky to meet Julian Lange, Anurag Chandra, Ayla Ergun, Kate Lesnaia, Erin Maneri and Anshul Sood, my first friends in Boston. It was they who help me adjust to the American lifestyle when I first moved and have influenced my life in many great ways. Special thanks to Lucy Wong for being a constant source of enthusiasm and inspiration (and also for letting me take 402B). Also, special thanks to Jon Kennell, my apartment mate for two years. The days with Mango Lazy were the best. Thanks to Matt Lehman and Helana, our Gwen Stefani during Mango Lazy's early days. I also

wish to thank my close friends from IIT Delhi: Ankit Bal, Rohit Dube, Nikhil Garg, Dhaval Gothi, Sachin Jain, Hatinder Madan, Saurabh Mahajan, Dheeraj Malhotra, Ankur Mani and Alok Ranjan. Thank you guys for being with me always. Its they who made a punster out of me so that I could see the funnier side of life.

A very special thanks to my aunt Maya Das and my uncle Dilip Das. They were my family and took care of my well being all these years. A visit to Bedford was like a visit to home. Thanks to Geeta jaithai, Mukut jetpeha and Bipu dada, my earliest inspirations. Thanks to my family and friends back from home in India and there are so many others I wish to thank. Finally, thanks to Ma, Baba and my little sister Aakashi for their unconditional love and support. It was their strength and prayers that always carried me through the toughest times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Classification is one of the key tasks in many domains including affective computing and machine perception. For example, consider a computerized affective learning companion, a pro-active companion of a child working on an analytic task that aims to be sensitive to the emotional and cognitive aspects of the learning experience in an unobtrusive way. The affective learning companion [33] needs to reason about the emotional state of the learner based on facial expressions, the posture and other context information. The extracted information form the face, the posture and the context are called observations, which are sensed through sensors such as a video camera, a posture sensing chair and other hardware devices or a software. One of the main goals of the pattern recognition system is to associate a class-label with different observations, where the class-labels correspond to different affective states such as interest, boredom etc. Similarly, there are activity recognition scenarios, where the aim is to recognize different activities happening in the surroundings using a variety of sensors. For example, using these sensors we can obtain observations that describe the position information using a global positioning system (GPS), status of the cellular phone (that is what cell towers are visible), pedometer readings etc. Based on these observations, the task in an activity recognition scenario can be to identify activities, such as driving to the office, sitting, walking home. The same analogy can be extended to low-level vision tasks such as object detection. Given an image, the object detection and the segmentation task can be posed as a classification problem, where the aim is to classify each individual pixel as a background or a part belonging to the foreground. The observations in all of these scenarios are multi-dimensional and each dimension is considered as a feature. Note, that different modalities, such as face, posture etc contribute to many features in the observations.

Traditional models of supervised classification aim to learn a decision boundary given a set of observations, $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_n\}$, and the corresponding class labels, $\mathbf{t} = \{t_1, ..t_n\}$. However, there are many scenarios that far exceed this simplistic model. Often the training data is plagued by incompleteness: there is only partial information available about the observations (incompleteness in $\mathbf{X}$) or about the labels (incompleteness in $\mathbf{t}$).

There are many scenarios that can result in incomplete data. For example, many applications in affect recognition, activity recognition and machine perception can

encounter incompleteness in observations $\mathbf{X}$, especially where multimodal information is used, and where information from multiple sensors needs to be fused to recover the variable of interest. Some of the hardware and algorithms associated with some of the modes might fail occasionally, leading to missing features and making the standard pattern recognition machinery unusable on that chunk of data. Similarly there are many scenarios which can result in incompleteness in the labels $\mathbf{t}$. For instance, many practical applications have the characteristic of having very little labeled training data. They usually have lots of data (e.g. video), but most of it is unlabeled (because it is tedious, costly, and error-prone to have people label it). Further, the problem becomes even more challenging when there is labeling noise; that is, some data points have incorrect labels. Note that the noisy labels can be considered as a special case of incompleteness, where we only observe a transformed version of the true label. In many applications like emotion recognition, there is usually some uncertainty about the true labels of the data; thus, a principled approach is required to handle any labeling noise in the data. Another case of incompleteness occurs when the data points are categorized into easy to label high level classes, instead of much harder to obtain labels of interest. In these scenarios the actual labels of interest might be hard or very expensive to obtain and usually it is much easier and quicker to obtain coarser or auxiliary labels that co-occur with the labels of interest. For instance, in activity recognition it is usually very tiresome to collect labels corresponding to individual low-level activities such as walking, sleeping etc. However, it is quicker and far more convenient to obtain labels that describe your location (for example, outdoors vs. indoors).

Traditionally incompleteness has been addressed in a generative framework, which aims to jointly model the observations $\mathbf{X}$ together with the labels $\mathbf{t}$ and where the incompleteness is treated as a hidden or a latent variable. These generative approaches learn by integrating or marginalizing over the incompleteness. However, the generative approaches can run into problems when the dimensionality of observations is large or when there are very few samples in the training data; thus, making it impossible to learn the underlying probability distribution of the observations.

On the other hand, the discriminative approaches model the distribution of the labels conditioned on the observations; thus, the distribution of the input data $\mathbf{X}$ is never explicitly modeled. This alleviates some of the problems encountered with the generative model; however, now it becomes non-trivial to extend the framework to handle incomplete data. Some of the recent successes of discriminative models over generative models makes a strong case to develop methods that can handle incompleteness in the discriminative setting.

1.1 Types of Incompleteness

This thesis addresses four different types of incompleteness:

- **Incomplete Features:** This kind of incompleteness results when there are observations $\mathbf{x}_i \in \mathbf{X}$ with some features unobserved or missing. One example

where this kind of incompleteness might occur is multimodal pattern classification. There are a growing number of scenarios in affective computing and activity recognition where multimodal information is used. However, the sensors might often fail and result in missing or bad data, a frequent problem in many multimodal systems.

- **Partially Labeled Data:** Often in affect and activity recognition scenarios it might be expensive or difficult to obtain the labels. This problem is addressed in semi-supervised classification, where only a part of the available data is annotated. Specifically, given a set of data points, among which few are labeled, the goal in semi-supervised learning is to predict the labels of the unlabeled points using the *complete* set of data points. By looking at both the labeled and the unlabeled data, the learning algorithm can exploit the distribution of data points to learn the classification boundary.

- **Coarse Labels:** In most activity recognition scenarios, each activity can be broken down into sub-activities, which often form a hierarchy. Also, in many cases it is easier to build robust classifiers to recognize sub-activities lower in this hierarchy. Similarly in object recognition from static images, the objects can be broken down into parts and often it is easier to build robust parts classifiers than a single object detector. Further, in affect recognition scenarios it is often easier to get high level labels for emotions such as positive/negative or excited/calm instead of labels such as interest/frustration/surprise. However, fully supervised training of these hierarchical activity/object/affect recognition system is difficult as it is very expensive to obtain training data annotated for all the sub-classes. Also, note that in many scenarios it might be unclear how to select these sub-classes. Thus, the training data might be just annotated for the higher level classes and the challenge with this kind of data is to train a pattern recognition system that exploits the hierarchical structure using only the available coarse or auxiliary labelings.

- **Noisy Labels:** Finally, the machine learning system needs labeled data and there are many scenarios where the labels provided might be incorrect. For example in the domain of affect recognition, getting the ground truth of labels in natural data is a challenging task. There is always some uncertainty about the true labels of the data. There might be labeling noise; that is, some data points might have incorrect labels. We wish to develop a principled approach to handle this type of incompleteness as well as all the other types mentioned above

1.1.1 Other Types of Incompleteness

The kinds of incompleteness are not limited to the ones described in the previous section. For example, multiple instance learning [19], solves a different kind of incompleteness where labels are provided to a collection of observations instead of each individual observation. Further, the labels in multiple instance learning just indicate

whether there is at least one observation in the collection with the label of interest. Thus, multiple instance learning is very different from the kinds of incompleteness described earlier and is mostly used in applications such as information retrieval. In our work we focus mainly on the problems arising in affective computing, activity recognition and other machine perception tasks and these applications mostly encounter the four kinds of incompleteness handled in this document.

1.2 Contributions of the Thesis

This thesis provides a Bayesian framework for learning discriminative models with incomplete data. The framework extends many different approaches to handle different kinds of incompleteness and provides insights and connections to many existing methods in semi-supervised learning, sensor-fusion and discriminative modeling. One major advantage of viewing different methods in a single Bayesian framework is that first we can use the Bayesian framework to perform model selection tasks, which were non-trivial earlier. Second, we can now derive new methods for discriminative modeling, such as semi-supervised learning and sensor-fusion by exploiting the modularity in the proposed unified framework.

The highly challenging problem addressed in this work is motivated by real-world problems in affective computing, activity recognition and machine perception and includes the scenarios described above: there is often multi-sensory data, channels are frequently missing, most of the training data is unlabeled, there might be labeling errors in the data and the available labels might be only loosely related to the desired ones. We present a Bayesian framework that extends the discriminative paradigm of classification to handle these four types of incompleteness. The framework builds upon Gaussian process classification and the learning and inference in this class of models is performed using a variety of Bayesian inference techniques, like Expectation Propagation [57], variational approximate inference and Monte-Carlo.

Figure 1-1 graphically depicts the Gaussian process classification and the different cases of incompleteness. The GP classification is a Bayesian framework for discriminative modeling and the aim is to model the distribution of the labels conditioned on the data points. The model assumes hidden real-valued random variable y that imposes a smoothness constraint on the solution via a Gaussian process with parameters Θ_{GP} (denoted by $p(y|\mathbf{x}, \Theta_{\text{GP}})$ in figure 1-1(a)). The relationship between the hidden random variable y and the observed label t is captured by the term $p(t|y, \Theta_{\text{Noise}})$ and can be used to encode the noise process to handle noisy labels. Figures 1-1(b), (c) and (d) show the rest of the cases of incompleteness. In figure 1-1(d) the distribution $p(z|t)$ models the relationship between the label t and the coarse label z. The GP classification framework has the capability to express the different scenarios of incompleteness and our aim in this thesis is to exploit the flexibility of the Bayesian framework to handle incomplete data in the discriminative setting. The following briefly describes the basic principles of the approach derived in this thesis:

- **Incomplete Features:** The incompleteness in features is addressed using a mixture of Gaussian processes [32]. The idea is to first train multiple classifiers

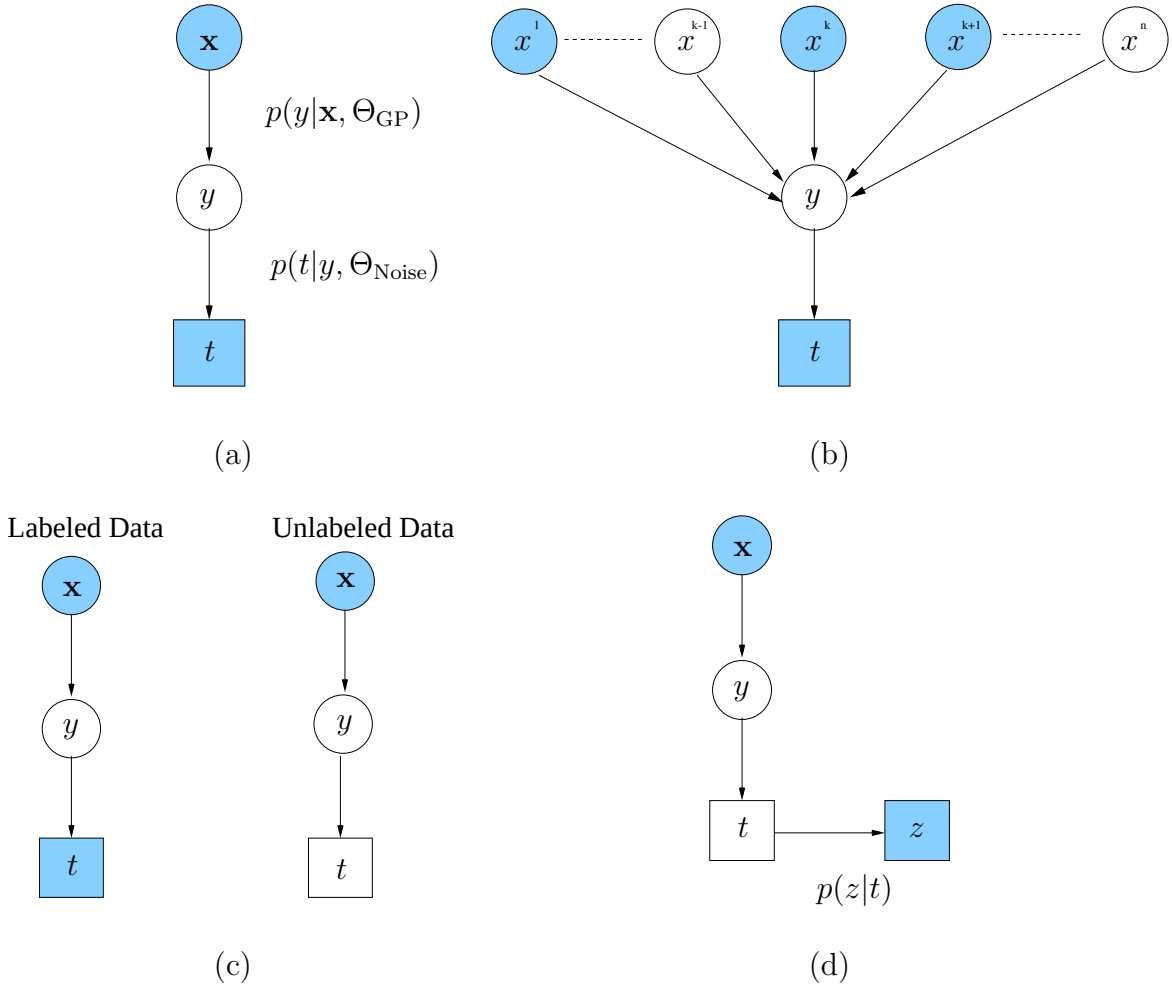


Figure 1-1: Graphical models to represent different kinds of incompleteness in Gaussian process classification. The shaded nodes in the models correspond to the random variables that are observed during the training. (a) GP classification with noise models, where $\mathbf{x}$ is the observed data point along with its label t in the training set. (b) The case of incomplete features, where we do not have all the dimensions of the data point $\mathbf{x}$. (c) Partially labeled data where we only observe labels corresponding to a few data points, whereas the rest of the points are unlabeled in the training data. (d) The case of coarse labels, where rather than observing the label t we only observe the coarse label z corresponding to the data point $\mathbf{x}$.

based on different subsets of features and then combine decisions from these multiple classifiers depending upon what channels are missing or erroneous. The framework extends the Gaussian Process classification to the mixture of Gaussian Processes, where the classification using each channel is learned via Expectation Propagation (EP), a technique for approximate Bayesian inference. The resulting posterior over each classification function is a product of Gaussians and can be updated very quickly. We evaluate the multi-sensor classification scheme on the challenging task of detecting the affective state of interest in children trying to solve a puzzle, combining sensory information from the face, the postures and the state of the puzzle task, to infer the student's state. The proposed unified approach achieves a significantly better recognition accuracy than classification based on individual channels and the standard classifier combination methods.

- **Partially Labeled Data:** The incompleteness due to partially labeled data is handled using input dependent regularization [74]. The proposed Bayesian framework is an extension of the Gaussian Process classification and connects many recent graph based semi-supervised learning work [79, 91, 92, 94]. We use Expectation Propagation that provides better approximations than previously used Laplace [94], with an additional benefit of deriving an algorithm that learns the kernel and the hyperparameters for graph based semi-supervised classification methods, while adhering to a Bayesian framework. An algorithm based on Expectation Maximization (EM) is used to maximize the evidence for simultaneously tuning the hyperparameters that define the structure of the similarity graph, the parameters that determine the transformation of the graph Laplacian, and any other parameters of the model. An additional advantage of the Bayesian framework is that we can explicitly model and estimate the different types of label noise in the data.

- **Coarse Labels:** The case of coarse label can be solved using the techniques that are used in generative modeling and have been earlier explored in analysis of hand-written diagrams using hidden random fields (HRF) [81]. Specifically, since we explicitly model the distribution of labels conditioned on $\mathbf{X}$, we can marginalize over the hidden variables that correspond to unknown but desired labels. The advantage of modeling sub-activities discriminatively is that the irrelevant sources of variability do not need to be modeled, whereas the generative alternative has to allow for all sources of variability in the data. One key contribution of the work mentioned in this thesis is the extension of HRF to model long range spatial dependencies. These models are specially effective on computer vision tasks and we use the proposed framework to address the problem of part-based object detection and recognition. By introducing the global position of the object as a latent variable, we can model the long-range spatial configuration of parts, as well as their local interactions. Experiments on benchmark datasets show that the use of discriminative parts leads to state-of-the-art detection and segmentation performance, with the additional benefit

of obtaining a labeling of the object's component parts.

- **Noisy Labels:** We propose to handle the noise in the data by using different kinds of noise models that allow different kinds of errors. The challenge here is to choose a good model that agrees with the training data. One of the advantages of this Bayesian framework is that we can use evidence maximization criteria to select the noise model and its parameters.

Figure 1-2 shows the overall framework to handle different kinds of incompleteness. The model is conditioned on different subsets of features denoted by $\{x\}^k$ in the figure. The model assumes that the hidden soft labels y^k arise due to a Gaussian process conditioned on the subset of features. The label of interest is denoted by t and z is the coarse label. Depending upon the kind of incompleteness, different random variable are observed during the training time (see figure 1-1). Further, the model is characterized using $p(y^k|\{x\}^k, \Theta_{\text{GP}})$, $p(t|y^1, .., y^P, \Theta_{\text{Noise}})$ and $p(z|t)$ corresponding to the smoothness constraint (parameterized by Θ_{GP}), the noise model (parameterized by Θ_{Noise}) and the statistical relationship between the label and the coarse label respectively. All the solutions proposed in this thesis can be combined into a single framework by considering different choices of these quantities and the different subsets of the features. Table 1.1 depicts these choices and highlights the relationship to the overall framework.

1.3 Thesis Outline

The chapters of the thesis are organized as follows:

- **Chapter 2:** Gaussian Process Classification with Noisy Labels

  Gaussian Process classification, which is the basic building block of the thesis is reviewed. Learning with noisy labels is addressed and a number of different noise models are discussed. Further, we demonstrate how various techniques for Gaussian process classification can be exploited to solve the challenging problem of affect recognition and object segmentation when only the noisy masks are available.

- **Chapter 3:** Mixture of Gaussian Processes to Combine Multiple Modalities

  The incompleteness due to missing channels and features is addressed in this chapter and we specifically focus on when there is limited labeled data available. The mixture of Gaussian Process classifiers is introduced and an approximate inference procedure for joint training is proposed, which results in a framework capable of quickly re-learning the classification given updated label associations.

- **Chapter 4:** Gaussian Process Classification with Partially Labeled Data

  This chapter discusses how the basic Gaussian Process classification framework can be extended to semi-supervised classification. The proposed framework

exploits input dependent regularization to handle the incompleteness arising due to partially labeled data and has connections to recently proposed graph based semi-supervised classification.

- **Chapter 5:** Located Hidden Random Fields to Learn Discriminative Parts

  This chapter extends the discriminative paradigm of classification to handle coarse and auxiliary labels. Rather than using the generative paradigm where the model tries to explain all the variability in the data, the focus here is on learning sub-classes discriminatively; thus, ultimately helping in explaining the provided coarse labels well.

- **Chapter 6:** Conclusion and Future Work

  This chapter summarizes the contributions and concludes the thesis by proposing extensions and future work.

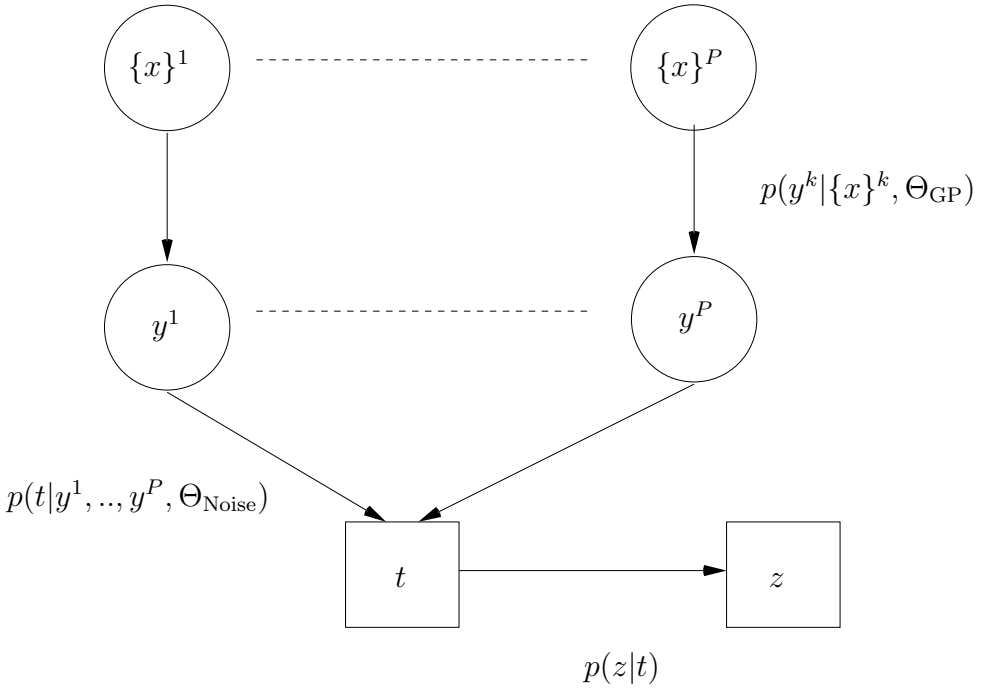


Figure 1-2: The overall framework to handle the four different kinds of incompleteness. The model is conditioned on different subsets of data features and each subset is denoted by $\{x\}^k$, where $k \in \{1, .., P\}$.

Table 1.1: Handling incompleteness with the proposed framework.

	Type of Incompleteness	**Subsets of Features**	**Prior** $p(y^k\|\{x\}^k, \Theta_{\text{GP}})$	**Likelihood** $p(t\|y^1, .., y^P, \Theta_{\text{Noise}})$	**Coarse Labels** $p(z\|t)$
Chapter 2	Noisy Labels	One Subset All features	Regular GP Prior	Different Noise Models	-NA-
Chapter 3	Incomplete Features	Multiple Subsets One Subset per Modality	Regular GP Prior	Probit Noise Model	-NA-
Chapter 4	Partially Labeled Data	One Subset All Features	Input Dependent GP Prior	Flipping Noise Model	-NA-
Chapter 5	Coarse Labels	One Subset All Features	Regular GP Prior	Logistic Noise Model	Determined by Application

Chapter 2

Gaussian Process Classification with Noisy Labels

This chapter covers the background on Gaussian Process (GP) classification and discusses the basic principles, methods for approximate inference and the framework of discriminative modeling in the Bayesian paradigm. Additionally, we discuss different models to handle noisy labels in the data and show experimental results highlighting classification in presence of the incorrect labels. Further, we empirically demonstrate how some problems in affect recognition can be effectively tackled and how modeling the noise properly can lead to significant gains. Finally, we demonstrate an application for object segmentation, which can be thought of as the discriminative version of GrabCut [70].

2.1 Gaussian Process Classification

Gaussian Process classification is related to kernel machines such as Support Vector Machines (SVMs) and has been well explored in machine learning. Under the framework, given a set of labeled data points $\mathbf{X}_L = \{\mathbf{x}_1, .., \mathbf{x}_n\}$, with class labels $\mathbf{t}_L = \{t_1, .., t_n\}$ and unlabeled points $\mathbf{X}_U = \{\mathbf{x}_{n+1}..\mathbf{x}_{n+m}\}$, we are interested in the distribution $p(\mathbf{t}_U|\mathbf{X}, \mathbf{t}_L)$. Here $\mathbf{X} = \{\mathbf{X}_L, \mathbf{X}_U\}$ and $\mathbf{t}_U$ are the random variables denoting the class labels for the unlabeled points $\mathbf{X}_U$. We limit ourselves to two-way classification, hence, the labels are, $t \in \{-1, 1\}$. Note, that there are ways to perform multi-label classification with GPs [76] and many of the techniques developed in this thesis can be extended to handle multiple classes.

Figure 2-1 shows the graphical model behind the Gaussian Process classification. Note, that we always observe $\mathbf{X}$, the data points (both training and testing) and $\mathbf{t}_L$ the labels corresponding to the data points in our training set. Intuitively, the idea behind GP classification is that the hard labels $\mathbf{t} = \{\mathbf{t}_L, \mathbf{t}_U\}$ depend upon hidden soft-labels $\mathbf{y} = \{y_1, ..., y_{n+m}\}$. These soft-hidden labels arise due to a Gaussian process which in turn imposes a smoothness constraint on the possible solutions. Given the labeled and unlabeled data points, our task is then to infer $p(\mathbf{t}_U|D)$, where $D = \{\mathbf{X}, \mathbf{t}_L\}$.

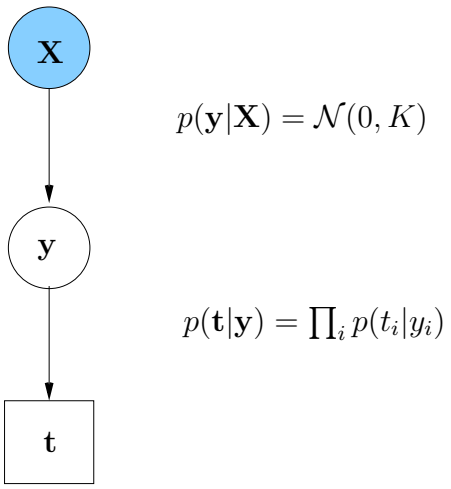


Figure 2-1: Graphical model for Gaussian process classification. Note that, $\mathbf{X} = \{\mathbf{X}_L, \mathbf{X}_U\}$ and is observed for both training and test data points. Also $\mathbf{t} = \{\mathbf{t}_L, \mathbf{t}_U\}$; however, only $\mathbf{t}_L$ corresponding to the labeled data points are observed.

Specifically:

$$p(\mathbf{t}_U|D) = p(\mathbf{t}_U|\mathbf{X}, \mathbf{t}_L) \propto \int_{\mathbf{y}} p(\mathbf{t}_U|\mathbf{y}) p(\mathbf{y}|\mathbf{X}, \mathbf{t}_L) \tag{2.1}$$

The full Bayesian treatment of GP classification requires computing the integral given in equation 2.1. However, many approaches such as the Bayes point machines [26, 58] just use the mean (or mode) of the posterior $p(\mathbf{y}|\mathbf{X}, \mathbf{t}_L)$ to provide a point classifier. As we will see in section 2.2.2, the mean of the posterior has a nice representation as a kernel classifier and can be used to classify any unseen point. The key quantity to compute is the posterior $p(\mathbf{y}|\mathbf{X}, \mathbf{t}_L)$, which if obtained in a simple approximation (such as Gaussian) can be used to compute the Bayes point or to perform Bayesian averaging as in equation 2.1. The required posterior can be written as:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{t}_L) = p(\mathbf{y}|D) \propto p(\mathbf{y}|\mathbf{X}) p(\mathbf{t}_L|\mathbf{y}) \tag{2.2}$$

The term $p(\mathbf{y}|\mathbf{X})$ in equation 2.2 is the GP prior and imposes a smoothness constraint such that the solutions that have same the labelings for similar data points are preferred. Formally we can write $p(\mathbf{y}|\mathbf{X}) \sim \mathcal{N}(0, K)$, which is the Gaussian process prior and it enforces a smoothness constraint via the covariance matrix K. The entries in the matrix K capture the notion of similarity between two points. We describe this in detail in section 2.1.1

The second term $p(\mathbf{t}_L|\mathbf{y})$ in equation 2.2 is the likelihood and incorporates information provided in the labels. The labels $\mathbf{t}$ are assumed to be conditionally independent given the soft labels $\mathbf{y}$. We discuss this in more detail in section 2.1.2.

Computing the posterior $p(\mathbf{y}|\mathbf{X}, \mathbf{t}_L)$ can be hard. There are many different kinds of likelihoods (see section 2.1.2) often used in classification that can make Bayesian inference non-trivial. One key technique to handle non-Gaussian likelihoods is to ap-

proximate $p(\mathbf{y}|D)$ as a Gaussian distribution using Assumed Density Filtering (ADF) or Expectation Propagation (EP). Section 2.2 describes some of these techniques.

2.1.1 Gaussian Process Priors

The prior $p(\mathbf{y}|\mathbf{X})$ plays a significant role in Gaussian process classification and is induced using Gaussian processes. The prior imposes a smoothness constraint such that it gives higher probability to the labelings that respect the similarity between the data points. Specifically, the intuition behind these priors is that similar data points should have the same class assignments.

This idea is captured by assuming a Gaussian process prior over the soft labels $\mathbf{y} = \{y_1, ..., y_{n+m}\}$. The similarity between the data points is defined using a kernel and examples of kernels include Gaussian, polynomial etc. Consequently, the soft labels $\mathbf{y}$ are assumed to be jointly Gaussian and the covariance between two outputs y_i and y_j specified using a kernel function applied to $\mathbf{x}_i$ and $\mathbf{x}_j$. Formally, $\mathbf{y} \sim \mathcal{N}(0, K)$ where K is a $(n+m)$-by-$(n+m)$ kernel matrix with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Note that the formulation of this prior can be extended to any finite collection of data points and this is the *process perspective* of Gaussian processes.

There is an equivalent *weight space* perspective, where the assumption is that the hidden soft-labels $\mathbf{y}$ arise due to application of a function $f(\cdot)$ directly on the input data points (i.e. $y = f(\mathbf{x})$). Further, the function takes the form of a linear combination of orthonormal basis functions. That is:

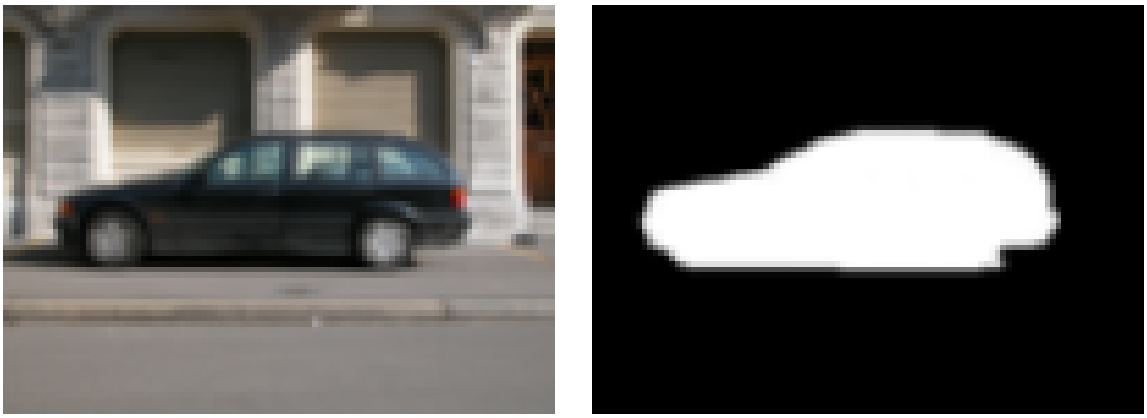
$$f(\mathbf{x}) = \sum_k w_k \nu^{1/2} \phi_k(\mathbf{x}) \tag{2.3}$$

Where ϕ_k are the eigenfunctions of the operator induced by k in the Reproducing Kernel Hilbert Space, ν_k are the corresponding eigenvalues and w_k are the weights. Note that the dimensionality of the basis can be infinite. Assuming a spherical Gaussian prior over the weights, that is $\mathbf{w} = [w_1, w_2, ..]^T \sim \mathcal{N}(0, \mathbf{I})$, it is easy to show that the hidden soft labels $\mathbf{y}$, resulting from evaluation of the function $f(\cdot)$ on the input data points $\mathbf{X}$, are jointly Gaussian with zero mean and covariance given by the kernel matrix K. In this work, we will mostly follow the process perspective, recognizing this relationship to the weight space perspective. For details readers are requested to look at [75].

2.1.2 Classification Likelihoods to Handle Noisy Labels

The likelihood models the probabilistic relation between the observed label t_i and the hidden label y_i. For Gaussian Process classification, the observed labels $\mathbf{t}_L$ are assumed to be conditionally independent given the soft labels $\mathbf{y}$. Consequently, the likelihood $p(\mathbf{t}_L|\mathbf{y})$ can be written as $p(\mathbf{t}_L|\mathbf{y}) = \prod_{i=1}^{n} p(t_i|y_i)$.

There are many real-world scenarios where there might be labeling errors in the data. For example, much of the data for applications in ubiquitous computing, affective computing and sensor network is labeled by humans, which can result in label errors due to human involvement. The erroneous observations t_i can be considered



(a) (b)

Figure 2-2: An example where the labelings provided by a human can be erroneous. (a) Input image, (b) labels provided by a human. It is far easier and less tedious to provide some incorrect labels. For example. the background visible through car windows are labeled as foreground. Note that these incorrect labels can be far from the decision boundary, consequently, the noise models that provide linear or quadratic slack might not work.

noisy and if we have any prior knowledge about the noise process, it should be encoded in the likelihood.

There are different approaches to handle errors in the labels and the errors in the labels are mostly modeled by providing a linear or a quadratic slack in the likelihood. These models work well in many of the situations, especially when the labeling error occurs because some of the training data points are too close to the decision boundary confusing the human labeling the data. While most people tend to model label errors with a linear or a quadratic slack in the likelihood, it has been noted that such an approach does not address the cases where label errors are far from the decision boundary [40].

There are many scenarios where the label error might not occur near the decision boundary and thus, the approaches that add a linear or a quadratic slack may not work well. Figure 2-2 shows an example. Here, the goal is to build a classifier that classifies foreground pixels (car) from the background pixels. We need the ground truth to train a discriminative model, however, a human might provide labels shown in figure 2-2(b), as it is tedious to provide correct label to each pixel. For example, the background visible through the windows and the ground near the wheel are both labeled as foreground. These labels are incorrect, but allowing this kind of error makes it easier for a person to label the image. Having a classification system that is robust to occurrence of some noisy labels can heavily reduce the manual effort involved in obtaining the training examples. Also, note that the incorrect labels in this example occurred due to the segmentation issues and not because the human got confused by the pixels that were near the decision boundary in the feature space.

Table 2.1 shows a few of the likelihood models which can be used in the Gaussian

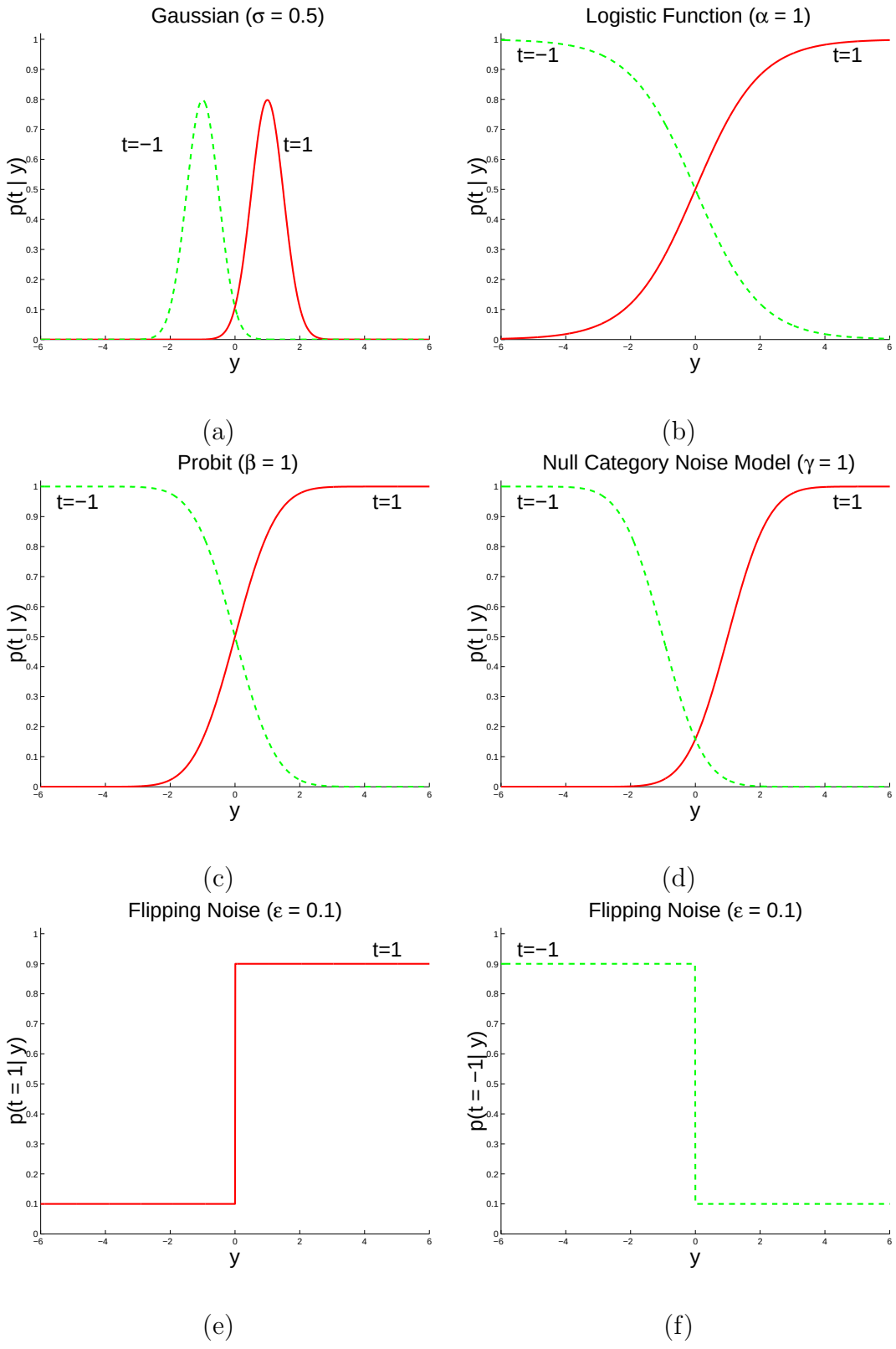


Figure 2-3: Noise models for Gaussian process classification. (a) Gaussian noise model, (b) logistic function, (c) probit noise model, (d) the null-category noise model (e) and (f) the flipping noise model plotted in two different figures for clarity.

Table 2.1: Label Noise Models for the Likelihood in Gaussian Process Classification.

Noise Model	$p(t_i\|y_i)$
Gaussian [88]	$p(t_i\|y_i) \propto \exp(-\frac{(y_i-t_i)^2}{2\sigma^2})$
Logistic Function [23, 87]	$p(t_i\|y_i) \propto \frac{1}{1+\exp(-\alpha t_i \cdot y_i)}$
Probit [66]	$p(t_i\|y_i) \propto \int_{-\infty}^{\beta t_i \cdot y_i} \mathcal{N}(z; 0, 1)$.
Null Category Noise Model [49]	$p(t_i\|y_i) = \begin{bmatrix} \phi(y_i - \gamma) & \text{if } t_i = 1 \\ \phi(y_i + \gamma) - \phi(y_i - \gamma) & \text{if } t_i = 0 \\ \phi(-(y_i + \gamma)) & \text{if } t_i = -1 \end{bmatrix}$ Here $\phi(\cdot)$ is a Normal cdf. $t_i = 0$ is a null category and never observed.
Flipping Noise [58, 63]	$p(t_i\|y_i) \propto \epsilon + (1 - 2\epsilon)\Phi(y_i \cdot t_i)$ Where, $\Phi(\cdot)$ is a step function

Process classification framework. Figure 2-3 plots these likelihoods and highlights the properties of the noise models. The Gaussian noise model 2-3(a) is used mostly in GP regression. It is also one of the popular choices for classification mostly because it is computationally easier to handle as the posterior over the hidden soft labels $\mathbf{y}$ can be written in terms of a closed form expression. The parameter σ can be thought of as a regularizer that controls the effect of the prior on the inference and controls the penalty for the noisy labels. However, as can be seen from the figure it is not a natural choice for classification and there are other noise models which are more suitable for the task.

The logistic regression likelihood (figure 2-3(b)) is a popular choice and provides a linear slack to handle noisy labels. Similarly, the probit noise model provides the quadratic slack 2-3(c) and both of these noise models inherently assume that the mistakes are near the decision boundary. The parameters α and β control the amount of slack to handle the noisy labels in these models.

The flipping noise model (figure 2-3(e) and (f)) on the other hand makes no assumption about the distribution of the incorrect labels. Here, Φ is the step function, ϵ is the labeling error rate and the model admits possibility of errors in labeling with a probability ϵ. Very similar likelihoods have been previously used for Gaussian Process

classification [63] and Bayes-point machines [57]. The above described likelihood explicitly models the labeling error rate; thus, the model is robust to label noise even if the mistakes are far from the decision boundary. However, the likelihood function is discontinuous; thus, it can cause problems when using optimization techniques such as gradient descent.

Recently, Lawrence and Jordan [49] introduced the null category noise model (figure 2-3(d)) that tries to imitate the SVM hinge loss function. The biggest advantage of this model appears to be in the semi-supervised cases (see Chapter 4). Similar to the spirit of the SVM the likelihood also assumes that the label errors occur near the decision boundary.

Except for the case of the Gaussian likelihood, computing the posterior $p(\mathbf{y}|D)$ is non-trivial. The next section discusses how to use approximate inference techniques to perform GP classification with non-Gaussian likelihoods.

2.2 Approximate Bayesian Inference

As mentioned earlier, the key step in Gaussian process classification is to evaluate the posterior $p(\mathbf{y}|D) \propto p(\mathbf{y}|\mathbf{X})p(\mathbf{t}_L|\mathbf{y})$. Although the prior $p(\mathbf{y}|\mathbf{X})$ is a Gaussian distribution, the exact posterior may not be a Gaussian and depends on the form of $p(\mathbf{t}_L|\mathbf{y})$.

When the noise model is Gaussian, the posterior is just a product of Gaussian terms and has a closed form expression. However, for all other non Gaussian likelihoods we have to resort to approximate inference techniques.

There have been a number of approaches for approximate inference for Gaussian process classification. There are sampling based techniques such as Markov Chain Monte Carlo and the billiard algorithm [72, 26], which resembles Gibbs sampling. The sampling based methods can provide very good approximations to the posteriors; however, they are computationally expensive. The deterministic approximation methods on the other hand can be faster and examples include variational approximations, Expectation Propagation, assumed density filtering and the Laplace approximation. Gibbs and Mackay [23] have described how to perform Gaussian process classification using the variational bounds and similarly, Opper and Winther have used the mean field approximation [63]. Another choice is the Laplace approximation, where the idea is to use a Taylor series expansion around the mode of the exact posterior to fit a Gaussian distribution. Assumed density filtering (ADF) and EP, which is a generalization of ADF, have been the most popular methods for deterministic approximate inference in Gaussian process classification [57, 48, 52]. Recently, Kuss and Rasmussen [46] compared the Laplace approximation with EP for Gaussian process classification and the results indicate that EP provides much better approximations of the posterior. In this thesis we use EP for approximate Bayesian inference for Gaussian process classification and the next subsection describes the main principles behind the method.

2.2.1 Expectation Propagation for Gaussian Process Classification

The posterior $p(\mathbf{y}|D)$ can be written as:

$$p(\mathbf{y}|D) \propto p(\mathbf{y}|\mathbf{X})p(\mathbf{t}_L|\mathbf{y}) \tag{2.4}$$

The idea behind the approach for GP classification is to use Expectation Propagation (EP) to first approximate $P(\mathbf{y}|D)$ as a Gaussian and then use the approximate distribution $p(y^*|D) \approx \mathcal{N}(M^*, V^*)$ to classify an unlabeled point $\mathbf{x}^*$:

$$p(t^*|D) \propto \int_{y^*} p(t^*|y^*)\mathcal{N}(M^*, V^*)$$

The prior $p(\mathbf{y}|\mathbf{X})$ is a Gaussian distribution, but the exact posterior is usually intractable when $p(\mathbf{t}_L|\mathbf{y}) = \prod_{i=1}^{n} p(t_i|y_i)$ is non Gaussian. Both ADF and EP aim to compute Gaussian approximations of each likelihood terms, that is:

$$p(t_i|y_i) \approx \tilde{t}_i = s_i \exp(-\frac{1}{2v_i}(y_i \cdot t_i - m_i)^2) \tag{2.5}$$

Hence, the approximation to the posterior is a product of Gaussian terms and can be computed easily. Conceptually, we can think of ADF/EP starting with the GP prior $\mathcal{N}(0, K)$ over the hidden soft labels $\mathbf{y}$ and incorporating all the approximate terms $\tilde{t}_i$ to approximate the posterior $p(\mathbf{y}|D) = \mathcal{N}(\mathbf{M}, V)$ as a Gaussian.

ADF and EP differ in the way they compute the Gaussian approximations of the likelihood. In ADF, the likelihood corresponding to each data point is incorporated and the resulting posterior is projected back to the exponential family (or a Gaussian). This procedure is repeated until all the data points are processed. The results of ADF are highly dependent upon the order in which the data points are processed and can often lead to poor approximation of the posterior. EP is an iterative version of ADF and usually provides better approximations. EP can be thought of as a generalization of ADF, where EP minimizes the effects of ordering of the data points by iteratively refining the projections

The basic outline of EP is shown in figure 2-4. The goal of EP is to approximate the posterior as a Gaussian and the algorithm begins by initializing the approximations of the likelihoods to have zero mean ($m_i = 0$) and infinite variance ($v_i = \inf$). Note, that this also initializes the approximate posterior: $Q = \mathcal{N}(0, K)$. Next, EP starts to refine each term approximation $\tilde{t}_i$ in an iterative manner. For each data point, the corresponding term $\tilde{t}_i$ is first removed from the posterior to compute a leave-one-out posterior (step 2(a)). Then, a refined approximation is computed by minimizing the KL-divergence between the leave-one-out posterior that uses the exact likelihood and the posterior with an approximation term limited to the Gaussian family (step 2(b)). The resulting approximation is of the form $\tilde{t}_i^{new} = s_i^{new} \exp(-\frac{1}{2v_i^{new}}(y_i \cdot t_i - m_i^{new})^2)$, which can be used to update the posterior (step 2(c)).

The refinement is repeated till convergence and the final term approximations are

1. **Initialization:**

   - Initialize $s_i = 1$, $v_i = \inf$, $m_i = 0$.
   - The approximation to the likelihoods:
     For all i : $p(t_i|y_i) \approx \tilde{t}_i = s_i \exp(-\frac{1}{2v_i}(y_i \cdot t_i - m_i)^2) = 1$
   - The approximate posterior is $Q = \mathcal{N}(0, K)$.

2. **Refinement:**
   Loop till Convergence

   For i from 1 to n

   (a) **Deletion:** Divide the approximate term $\tilde{t}_i$ from the approximate posterior Q to obtain the leave-one-out posterior $Q^{/i}$.

   (b) **Projection:** $\tilde{t}_i^{new} = \arg\min_{q \in Gaussian} KL[Q^{/i} p(t_i|y_i)||Q^{/i} q]$

   (c) **Inclusion:** Set $\tilde{t}_i = \tilde{t}_i^{new}$ and update the posterior $Q = Q^{/i} \cdot \tilde{t}_i^{new}$

3. **Output:**

   - The Gaussian approximation to the posterior:
     $Q = \mathcal{N}(0, K) \prod_{i=1}^{n} \tilde{t}_i = \mathcal{N}(\mathbf{M}, V)$.
   - The Gaussian approximations to the likelihoods:
     $p(t_i|y_i) \approx \tilde{t}_i = s_i \exp(-\frac{1}{2v_i}(y_i \cdot t_i - m_i)^2)$
   - The evidence: $p(\mathbf{t}_L|\mathbf{X}) \approx \frac{|V|^{1/2}}{|K|^{1/2}} \exp(B/2) \prod_{i=1}^{n} s_i$
     where $B = \mathbf{M}^T V^{-1} \mathbf{M} - \sum_{i=1}^{n} \frac{m_i^2}{v_i}$

Figure 2-4: The outline for approximate inference using Expectation Propagation for Gaussian process classification.

multiplied with the prior to yield the final approximate posterior. All but one step are independent of the noise model in the algorithm. Specifically, the step 2(b) in the algorithm depends upon the noise model and [56] shows how to handle different cases. Further, the convergence properties of EP are not yet well understood; however, in practice the algorithm mostly performs well for Gaussian process classification. In our experience, the algorithm converged for most of the cases. There were very few exceptions and mostly occurred when we used the flipping noise model on the cases where the label noise was very high.

There are a few other benefits of using EP which we will exploit to propose extensions in this thesis. First, one of the useful byproducts of EP is the Gaussian

approximations of the likelihoods $p(t_i|y_i)$:

$$p(t_i|y_i) \approx \tilde{t}_i = s_i \exp(-\frac{1}{2v_i}(y_i \cdot t_i - m_i)^2) \tag{2.6}$$

Note, that if we wish to exclude (include) a data point from the training set all we need to do is divide (multiply) the posterior by the Gaussian approximation term corresponding to the sample, which can be done efficiently. We will exploit this trick to train a mixture of Gaussian processes in Chapter 3. Second, EP automatically provides the normalization constant $p(\mathbf{t}_L|\mathbf{X})$ for the posterior. This constant is called the evidence or the marginalized likelihood and can be used to learn hyperparameters of the model. We will use evidence maximization for hyperparameter and kernel learning in Chapter 4 for semi-supervised scenarios.

2.2.2 Representing the Posterior Mean and the Covariance

As mentioned earlier, the full Bayesian treatment requires computing the integral given in equation 2.1 and it might seem that we need to perform Bayesian inference every time we need to classify a test point. However, there is a nice representation of the posterior mean as well as the covariance in terms of the kernel, which allows us to express the Bayes point as a kernel classifier. Details of the proof can be found in [17].

Specifically, given a prior distribution $p(\mathbf{y}|\mathbf{X}) \sim \mathcal{N}(0, K)$, for any data point $\mathbf{x}$ the corresponding mean and the covariance of the posterior $p(y|D)$, can be written as:

$$E[y_{\mathbf{x}}] = \bar{y}_{\mathbf{x}} = \sum_{i=1}^{n} \alpha_i \cdot k(\mathbf{x}_i, \mathbf{x}) \;\text{ where, } \alpha_i \in I\!R$$

$$E[(y_{\mathbf{x}} - \bar{y}_{\mathbf{x}})(y_{\mathbf{x}'} - \bar{y}_{\mathbf{x}'})] = k(\mathbf{x}, \mathbf{x}') + \sum_{i,j=1}^{n} k(\mathbf{x}, \mathbf{x}_i) \cdot \beta_{ij} \cdot k(\mathbf{x}_j, \mathbf{x}') \;\text{ where, } \beta_{ij} \in I\!R$$

We can either use the the mean as a point classifier or use equation 2.1 to classify any test points easily by using the representations mentioned above.

2.3 Hyperparameter Learning

The performance of the Gaussian process classification depends upon the kernel. The notion of similarity is captured using a kernel matrix and the performance is highly dependent upon the hyperparameters that describe the kernel. Further, there are parameters in the noise model (for example the slack penalty) and finding the right set of all these parameters can be a challenge. Many discriminative models, including SVMs often use cross-validation, which is a robust measure but can be prohibitively

expensive for real-world problems and problematic when we have few labeled data points.

Ideally we should be able to marginalize over these hyperparameters and there have been approaches based on Hybrid Monte Carlo [87]. However, these approaches are expensive and empirical Bayes provides a computationally cheaper alternative. The idea here is to maximize the marginal likelihood or the evidence, which is nothing but the constant $p(\mathbf{t}_L|\mathbf{X})$ that normalizes the posterior. This methodology of tuning the hyperparameter is often called evidence maximization and has been one of the favorite tools for performing model selection. Evidence is a numerical quantity and signifies how well a model fits the given data. By comparing the evidence corresponding to the different models (or hyperparameters that determine the model), we can choose the model and the hyperparameters suitable for the task.

Let us denote the hyperparameters of the model as Θ, which contains all the hyperparameters of the kernel as well as the parameter of the noise model. Formally, the idea behind evidence maximization is to choose a set of hyperparameters that maximize the evidence. That is, $\hat{\Theta} = \arg\max_{\Theta} \log[p(\mathbf{t}_L|\mathbf{X}, \Theta)]$.

Note that we get evidence as a byproduct when using EP (see figure 2-4). When the parameter space is small then a simple line search or the Matlab function `fminbnd`, based on golden section search and parabolic interpolation, can be used. However, in the case of a large parameter space exhaustive search is not feasible. In those cases, non-linear optimization techniques, such as gradient descent or Expectation Maximization (EM) can be used to optimize the evidence. However, the gradients of evidence are not easy to compute despite the fact that the evidence is a byproduct of EP.

In the past, there have been a few approaches that learn hyperparameters using evidence maximization. Notable among them are [18, 23], which are based on variational lower bounds and saddle point approximations. The EM-EP algorithm [40] is an alternative suitable when EP is used for approximate inference. The algorithm maximizes evidence based on the variational lower bound and similar ideas have been also outlined by Seeger [75]. In the E-step of the algorithm EP is used to infer the posterior $q(\mathbf{y})$ over the soft labels. The M-step consists of maximizing the variational lower bound:

$$\begin{aligned} F &= \int_{\mathbf{y}} q(\mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{X}, \Theta) p(\mathbf{t}_L|\mathbf{y}, \Theta)}{q(\mathbf{y})} \\ &= -\int_{\mathbf{y}} q(\mathbf{y}) \log q(\mathbf{y}) + \int_{\mathbf{y}} q(\mathbf{y}) \log \mathcal{N}(\mathbf{y}; 0, K) \\ &+ \sum_{i=1}^{n} \int_{y_i} q(y_i) \log p(t_i|y_i) \leq p(\mathbf{t}_L|\mathbf{X}, \Theta) \end{aligned}$$

The EM procedure alternates between the E-step and the M-step until convergence.

- **E-Step**: Given the current parameters Θ^i, approximate the posterior $q(\mathbf{y}) \sim \mathcal{N}(\bar{\mathbf{y}}, \Sigma_{\mathbf{y}})$ by EP.

- **M-Step**: Update $\Theta^{i+1} = \arg\max_{\Theta} \int_{\mathbf{y}} q(\mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{X}, \Theta) p(\mathbf{t}_L|\mathbf{y}, \Theta)}{q(\mathbf{y})}$

Table 2.2: Features used in the quit detection task.

Face Tracker	**Posture Sensor**	**Skin Conductivity**	**Pressure Mouse**
Presence of nods	Activity	Conductance	Activity
Presence of shakes	Postures		Skew
Probability of fidget			Mean
Probability of smile			Variance
Presence of blink			
Head velocity			
Head tilt			

In the M-step the maximization with respect to the hyperparameters Θ cannot be computed in a closed form, but can be solved using gradient descent. The EM procedure described is susceptible to local minima and one of the ways to get around this problem is to perform multiple searches with different initializations. Also, there are other ways besides this EM algorithm to maximize the evidence. For example, note that even though the gradients of the evidence are hard to compute, they can be approximated by the gradients of the variational lower bound and can be used in any gradient ascent procedure.

2.4 Experiments

The purpose of this section is two fold: first, we performed experiments to (i) demonstrate Gaussian process classification, (ii) demonstrate hyperparameter learning and (ii) compare different noise models on the classification task with noisy labels. Second, we highlight how we can exploit the techniques mentioned so far to solve the hard real-world problems in affect recognition and image classification.

We demonstrate the features of GP classification on many different tasks. The first task is to detect children in a learning environment, who are getting off-task and about to quit, based on their facial expressions, postures, physiology and mouse behavior. Further, we show the effect of different noise models on the classification accuracy by performing experiments on benchmark datasets. The final task is motivated from object detection and the goal is to classify foreground pixels from the background. Datasets for both of these tasks and the details of the experiments are discussed below.

Detecting Quitters

We look at the problem of detecting the affective state in which a child solving an analytical puzzle, such as towers of Hanoi, might quit. There are many applications of this system like intelligent tutoring systems, adaptive interfaces etc. The scenario we focus on has a child solving a puzzle and a machine that detects the affective state using the face, the postures, skin conductivity and the pressure sensitive mouse. The

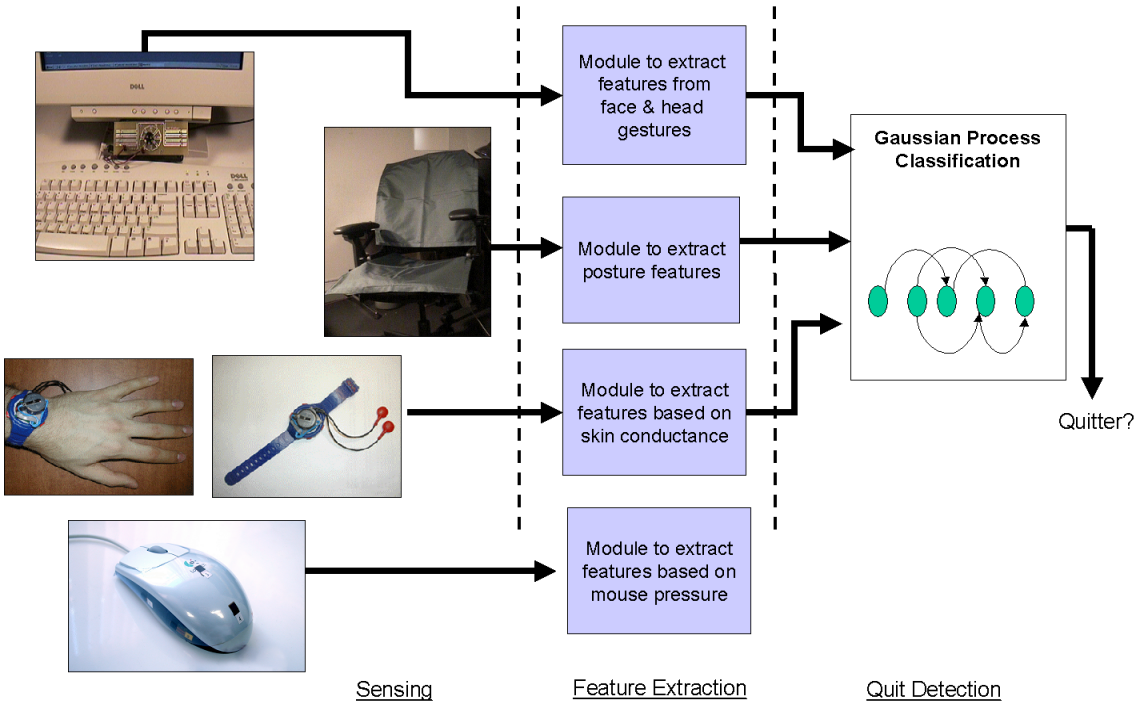


Figure 2-5: The architecture for detecting quitters in learning environments.

overall system architecture is shown in figure 2-5. The raw data from the camera, the posture sensor, the skin conductivity sensor and the pressure mouse is first analyzed to extract 14 features. These features are mentioned in table 2.2. The details about the facial feature tracking and the posture analysis can be found in appendix A. A pressure sensitive mouse [69], equipped with pressure sensors, provided some of the features for affect recognition. Further, HandWave was used to sense the skin conductance and the details of the sensor can be found in Strauss et al. [80]. For the purpose of classification we use the values averaged over 150 seconds of activity. All the features are scaled to lie between the value of zero and one before we take the mean. For the children that quit the game, we consider samples that summarize 150 seconds of activity preceding the exact time when they quit. However, for the children that did not quit, we look at a 150 seconds long window beginning 225 seconds after the start of the game. There are 24 samples in the dataset that correspond to 24 different children. Out of 24, 10 children decided to quit the game and the rest continued till the end. Thus, we have a dataset with 24 samples of dimensionality 14, where 10 samples belong to class +1 (quitters) and 14 to class -1 (non-quitters).

Note, that apriori we do not have any information regarding what features are most discriminatory and there is no reason to believe that all the features should be equally useful to provide a good classification. We need to weigh each feature individually;

Table 2.3: Recognition results for detecting quitters.

	Quit *10 Samples*		**Did Not Quit** *14 Samples*		**Accuracy**
	Correct	Misses	Correct	Misses	
Gaussian Process	8	2	11	3	79.17%
1-Nearest Neighbor	6	4	10	4	66.67%
SVM (RBF Kernel)	6	4	11	3	70.83%
SVM (GP Kernel)	8	2	11	3	79.17%

thus, for GP classification we use a Gaussian kernel shown in the equation below:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp[-\frac{1}{2} \cdot \sum_{k=1}^{K} \frac{(x_i^k - x_j^k)^2}{\sigma_k^2}] \tag{2.7}$$

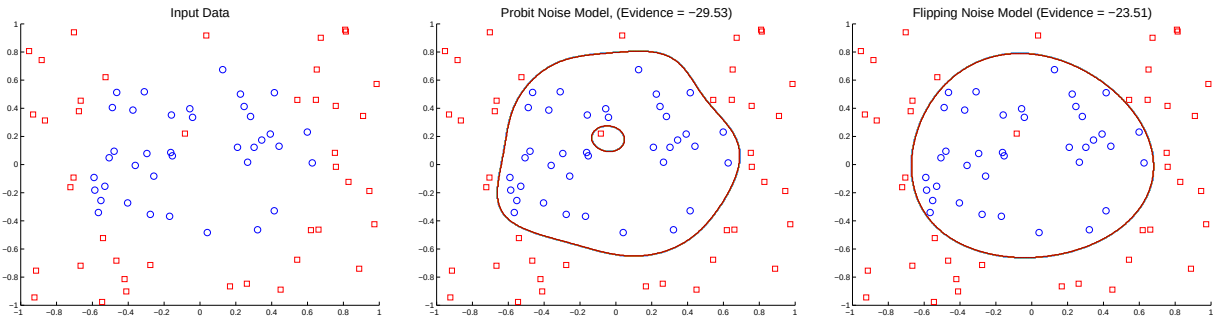
Here, x_i^k corresponds to the k^{th} feature in the i^{th} sample and note that we are weighing each dimension individually before exponentiating. The performance of the algorithm depends highly upon the kernel widths, $[\sigma_1, .., \sigma_k]$, and we use the EM-EP algorithm to tune those hyperparameters.

Each round of EM-EP take $O(n^3)$ steps, where n is the number of data points in the dataset. For the quit detection problem the whole algorithm takes around 2-3 minutes to converge. Once we have the kernel hyperparameters and the Bayes point, classifying a new point hardly takes more than a few milliseconds. The standard SVM training algorithm also takes $O(n^3)$ steps, but there are other faster alternatives. A popular trick for speeding up Gaussian process classification is to use sparsification [48, 17], which can significantly reduce the time required for the Bayesian inference.

We performed experiments with different classification techniques which include one-nearest neighbor, SVMs and GP classification. Table 2.3 shows the recognition results. The accuracies are reported using the leave-one-subject out strategy, which is to first choose all but one subject's data as training examples and test the performance on the one left out. This process is repeated for all the available samples and we report all the correct detections and misses as well as the false alarms (Table 2.3)

With Gaussian process classification we use the kernel (equation 2.7), which weighs all the features individually and we exploit the evidence maximization framework to tune those weights. Thus, GP classification allows us to use expressive models that can capture various relationships between the features in the data. The dimensionality of the observations is 14 in this example and it is non-trivial to tune these weights using cross-validation. For SVMs, we stick with the RBF kernel and use leave-one-out cross-validation to tune the kernel width. Table 2.3 demonstrates the advantage of using the Bayesian framework and we can see that the Gaussian process classification outperforms both 1-nearest neighbor strategy and the SVM with an RBF kernel.

Further, we also train an SVM that uses the maximum evidence kernel learnt using EM-EP algorithm for the Gaussian process classification and as we can see it performs the same as Gaussian process classification. Note that in this case the



(a) (b) (c)

Figure 2-6: Effect of different noise models for Gaussian process classification. (a) Training data with one noisy label. Bayes point classification boundary using (b) the probit noise model and (c) the flipping noise model.

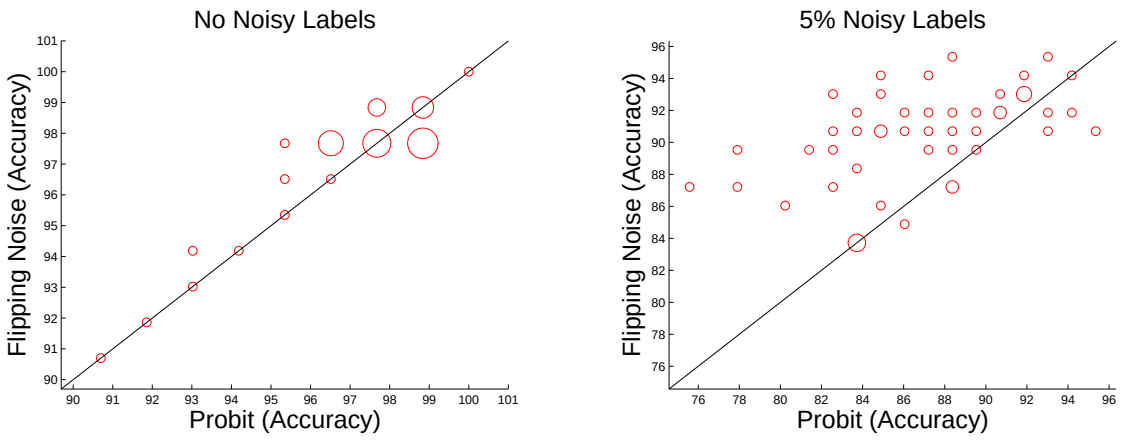
EM-EP algorithm has already found the correct kernel for classification and SVM is greatly benefiting from this computation. This experiment illustrates that GP classification provides a good framework to classify with an additional benefit of the evidence maximization for learning the hyperparameters.

Classification with Different Noise Models

We compare GP classification with the flipping noise model as well as the probit noise model (see table 2.1). The GP classification with probit noise model, which provides a quadratic slack to handle noisy labels, inherently assumes that the mistakes can only occur near the classification boundary. The flipping noise model on the other hand models the error probabilistically and does not assume that the mistakes are near the decision boundary.

Figure 2-6 illustrates the effects of two different noise models on a toy data set. Figure 2-6(a) shows the training examples, where the squares and the circles represent two different classes. Note, that the one noisy data point in this set is a square that lies right around the center and is far from the correct decision boundary. The decision boundaries (Bayes point) for the probit and the flipping noise models are shown in figure 2-6(b) and (c) respectively. We used the RBF kernel in this example and the hyperparameters for the kernel and the noise model were learnt using evidence maximization.

From the figure we can see that the probit noise model does not handle the case well when the error can be far from the decision boundary. The flipping noise model on the other hand finds a better solution. Similar observations have been earlier suggested by Kim et al. [40], however they did not directly compare the results with the probit noise model. We can also see that the evidence obtained with the flipping noise model was -23.51 and is much higher than -29.53, the evidence for the probit noise model. This suggests that the evidence maximization framework can be used to select noise models and their parameters.



(a) (b)

Figure 2-7: Gaussian process classification on the thyroid data with (a) no noisy labels, (b) labels with 5% noise. There are 50 points on each graph and each point is (accuracy probit, accuracy flipping) and corresponds to one test run. Circle width is proportional to the number of points having that coordinate. Points above the diagonal indicate where the classification using flipping noise model was more accurate. While both the noise models perform almost equally with no label noise (a) the flipping noise is particularly better when there are noisy labels (b).

We further performed tests on the thyroid dataset from the UCI machine learning repository [60]. This dataset has more than two classes, however we only look at the binary classification problem of distinguishing normal samples from the abnormal ones. The data is split into the training and the testing set with 60% samples as the training examples and the rest as test. Further, we introduce some labeling noise by randomly flipping 5% of the correct labels. All the features in the training set are scaled to have zero mean and unit variance. The Bayes point classification is performed using an RBF kernel and again we use evidence maximization to choose kernel hyperparameters and the parameters of the noise model. This testing scheme is repeated 50 times and the results are shown in figure 2-7 (a) and (b). There are 50 points in each graph and every point in the figure represents one test run. The x coordinates represent the accuracy obtained when using the probit noise model and the y coordinates indicate recognition obtained from the flipping noise model. The point lies above the diagonal whenever the flipping noise model beats the probit noise model.

Figure 2-7(a) shows that neither of the models is significantly better when there is no label noise. However, the flipping noise model beats the probit noise model significantly in the presence of noisy labels (figure 2-7(b)). Consequently, we can conclude that choosing an appropriate noise model can lead to significant gains in classification. We'd like to point out that the process by which the noise was introduced in the data favors flipping noise and there might be other scenarios where the probit noise model can be better. As discussed earlier, one big advantage of the slack based

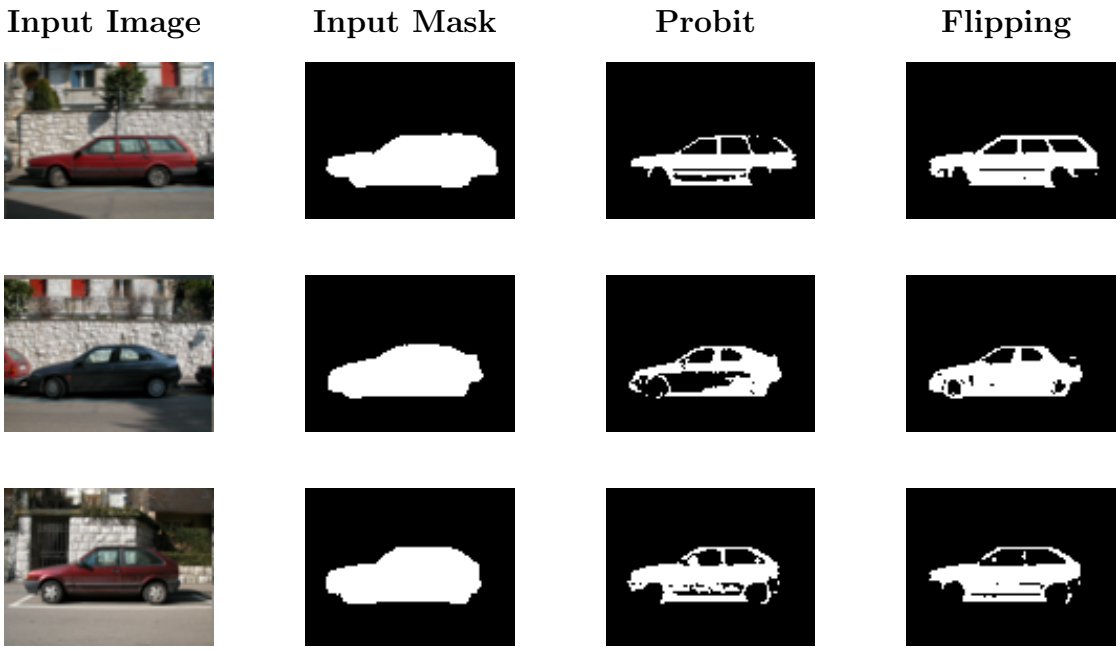


Figure 2-8: Refined segmentation obtained using different noise models. The first column shows the actual image and the noisy masks are shown in the next column. The errors in the masks are due to lazy segmentations and the mistakes do not necessarily lie near the decision boundary. The third and the fourth column show refined segmentations obtained by using GP classification with the probit and the flipping noise model respectively. The flipping noise model performs better than the slack based probit noise model.

noise models is that they are continuous; thus, are preferred whenever we are using optimizers to find a map solution. However, we can envisage a hybrid noise model by using a probit function instead of the step function in the flipping noise model; hence, it is also possible to combine the advantages of the both.

Towards Discriminative GrabCut: Pixel Classification with Noisy Masks

We now illustrate an application motivated by object detection and segmentation where flipping noise can be very useful. We use the TU Darmstadt car dataset [50], where the task is to classify foreground (car) pixels from the background. However, as described earlier it is challenging to obtain completely accurate ground truth segmentations for training the classifiers. The segmentations, as shown previously in figure 2-2, can be noisy due to the ease in labeling some background pixels as foreground.

The TU Darmstadt dataset contains images of different cars viewed from the side. The cars were all facing left and are at the same scale in all the images. All images were resized to 100×75 pixels. Our aim in this experiment was to take the available noisy segmentations and exploit the noise models to detect incorrectly labeled pixels. For each labeled image, we first trained Gaussian process classifiers

with the different noise models using the available segmentations. The pixel features used in classification were the rgb values. We use a linear kernel for all of these methods and the only difference is the noise model. Once trained, we apply the classifier to the training image itself to obtain a refined labeling. This procedure can be iterated to obtain better segmentations. Note, that this scheme can be thought of as a discriminative version of grabcut [70]. Grabcut is an object segmentation scheme that aims to classify the foreground pixels from the background given an image annotated for a region that contains most of the foreground. The grabcut uses generative models together with spatial smoothness constraints to provide the required segmentation. Note, that the scheme described in this section is a simpler version as there are no spatial constraints that impose spatial smoothness; however, these constraints can be easily introduced using a Conditional Random Field (CRF) formulation (see Chapter 5).

Figure 2-8 shows segmentation results using the GP classification with different noise models. GP classification with the flipping noise model provides the most convincing segmentation as the labels provided for training are noisy due to lazy segmentation of the training images. The GP classification with the probit model that provides quadratic slack do not do that well as it assumes that the mistakes are near the decision boundary. There are other salt and pepper noise in the final segmentations, which we remove by multiplying the classification result with the original training mask. We can also tackle this problem using spatial smoothing and imposing long range dependencies. Further, our choice of features is pretty simple in this experiment. In Chapter 5 we will see how to address many of these issues.

2.5 Conclusions and Extensions

In this chapter we introduced Gaussian process classification with different noise models, together with the basic ideas behind the approximate Bayesian inference and hyperparameter learning. We demonstrated hyperparameter learning and classification on an affect recognition task, where the goal was to detect quitters using observations recorded from face, posture, pressure and skin conductivity. Further, we demonstrated using synthetic examples, benchmark datasets and a pixel classification task that choosing a good noise model can improve recognition performance.

Gaussian Process Classification with Incomplete Data

Gaussian process classification aims to directly model the decision boundary given the training data and it is not trivial to handle all cases of incompleteness in the data with these models. Out of the four cases of incompleteness handled in this thesis, the classic GP classification has capabilities to handle only one of them. As discussed in section 2.1.2, we have shown how to use different noise models to handle the case of incorrect labels. We briefly describe difficulties associated with the rest of the cases as a roadmap for the rest of this thesis:

- **Incomplete Features:** One never models the underlying data density $p(\mathbf{x})$ in GP classification; thus, handling the case of missing features in the observations is not straight forward. The smoothness constraint in GP classification depends upon the kernel matrix K, which is hard to compute with incomplete features. In Chapter 3 we will provide extensions to GP classification using a mixture of Gaussian processes.

- **Partially Labeled Data:** As shown in this chapter, GP classifiers first assume a smoothness constraint (the GP prior) and then incorporate information from the labeled data only. The smoothness constraint only encodes the assumption that similar data points should have the same label and is independent of the underlying data density. Hence, any information from the unlabeled data is unused. In Chapter 4, we will show how to use data dependent regularization to tackle this case of incompleteness.

- **Coarse Labels:** Gaussian process classification provides a nice framework for probabilistic discrimination of the data but in the original form is just limited to the supervised classification scenarios. In Chapter 5 we will show how we can exploit the flexibility provided by the Bayesian discriminative models to encode prior knowledge about the problem using various graphical models.

Chapter 3

Mixture of Gaussian Processes for Combining Multiple Modalities

This chapter[1] describes a unified approach based on Gaussian Processes for achieving sensor fusion under the problematic conditions of missing channels and noisy labels. There are a growing number of scenarios in pattern recognition where multimodal information is used, and where information from multiple sensors needs to be fused to recover the variable of interest. Multi-sensor classification is a problem that has been addressed previously by using either data-level fusion or classifier combination schemes. In the former, a single classifier is trained on joint features; however, when the data has even one missing channel, a frequent problem, then usually all the data is ignored for that time block, resulting in a significant reduction in the total amount of data for training.

We propose to handle the incompleteness resulting from missing features using a mixture of Gaussian Processes. Note that marginalization of incomplete features requires explicit modeling of the probability density over the observations, which can be hard to model correctly when the dimensionality of the observations is big compared to the number of training examples. Rather than marginalizing over the missing data, the basic idea behind the proposed framework is to combine decisions of individual classifiers trained on different subsets of the features. Consequently, whenever there are missing features the decision level fusion ignores the contribution from the classifiers that act on the set of missing features. There are a lot of ways to combine the decisions, and the proposed framework exploits the idea behind the mixture-of-experts [30] and, as shown in the experiments section, is a better choice than most of the rule-based methods. Under the proposed approach, Gaussian Processes generate separate class labels corresponding to each individual modality. The final classification is based upon a hidden random variable, which probabilistically combines the sensors. Given both labeled and test data, the inference on unknown variables, parameters and class labels for the test data is performed using the variational bound and Expectation Propagation. There are big computational advantages

[1]The work described in this chapter appeared in the ACM Conference on Multimedia, 2005 [35] and the Workshop on Multiple Classifier Systems, 2005 [32]

of using this framework as for each *expert* the posterior over the class labels is obtained via Expectation Propagation. Hence, the approximate posterior over each classification function is a product of Gaussians and can be updated very quickly, resulting in huge computational savings.

The proposed extension can be combined easily with other extensions proposed in this thesis. For example, this model can be extended to handle the case of noisy labels and can be applied to the highly challenging problems combining many of the incompleteness scenarios described in the first chapter, such as semi-supervised cases with multi-sensory data, where channels are frequently missing and there might be labeling errors in the data. We apply this method to the challenge of detecting the affective state of interest in children trying to solve a puzzle, combining sensory information from the face, the postures and the state of the puzzle task, to infer the student's state. Classification with the proposed new approach achieves accuracy of over 86%, significantly outperforming the classification using individual modalities and other common classifier combination schemes.

3.1 Previous Work

Classification with Incomplete Data: Classifiers based on the generative paradigm explicitly model the underlying distribution of the observations and can be used easily with missing data by integrating out the incomplete features. The discriminative models on the other hand never explicitly model the observation density; thus, they cannot easily handle the incompleteness in features. One of the popular methods to get around this problem is to use imputation, where the missing data is filled with specific values, such as zeros or the mean of the distribution. There are other alternatives that require explicit models of the probability density of the observations. For example, Williams et al. [89] use a mixture of Gaussians to model the density of observations. Then, there are methods described in Rubin [71] based on sampling from the underlying observation density. As these methods explicitly model the data distribution, they can encounter problems when the dimensionality of the observation is huge as compared to the number of available training examples. Also, there are methods based on kernel completion [24, 85], where the idea is to complete the kernel matrix based on some optimization criteria or other auxiliary observations. However, these methods do not take into account the uncertainty of the missing features and it is not clear how well they perform when the kernel matrices are sparse.

The work presented in this chapter proposes an alternative way to handle the incompleteness. The idea is to train classifiers based on different subsets of features and then use a classifier combination strategy to provide a final decision. Whenever the framework encounters missing features in the data, the decision level fusion ignores those classifiers that act on missing data. Note that this framework does not ever explicitly model the underlying observation density; thus, it can be used in scenarios where the dimensionality of the observations is big compared to the number of examples.

Classifier Combination: A lot of researchers have looked into the general problem

of combining information from multiple channels. There are many methods, including Boosting [73] and Bagging [9], which generate an ensemble of classifiers by choosing different samples from the training set. These methods require a common set of training data, which is a set of joint vectors formed by stacking the features extracted from all the modalities into one big vector. As mentioned earlier, often in multi-sensor fusion problems the sensors or feature extraction might fail, resulting in data with missing channels; thus, most of the data cannot be used to form a common set of training data. Similarly, most of the data remains unused in "feature-level fusion," where a single classifier is trained on joint features.

One alternate solution is to use decision-level fusion. Kittler et al. [41] have described a common framework for combining classifiers and provided theoretical justification for using simple operators such as majority vote, sum, product, maximum and minimum. Hong and Jain [27] have used a similar framework to fuse multiple modalities for personal identification. Similarly, Han and Bhanu [25] also perform rule-based fusion for gait-based human recognition. One problem with these fixed rules is that, it is difficult to predict which rule would perform best. Then there are methods, such as layered HMMs proposed by Oliver et al. [62], which perform decision fusion and sensor selection depending upon utility functions and stacked classifiers. One main disadvantage of using stacked based classification is that these methods require a large amount of labeled training data. Alternatively, there are other mixture-of-experts [30] and critic-driven approaches [54, 29] where base-level classifiers (experts) are combined using second level classifiers (critics or gating functions) that predict how well an expert is going to perform on the current input. To make a classifier selection, the critic can either look at the current input or base its decision upon some other contextual features as well. For example, Toyama and Horvitz [83] demonstrate a head tracking system based on multiple algorithms, that uses contextual features as reliability indicators for the different tracking algorithms. In an earlier work [36] we have proposed an expert-critic system based on HMMs to combine multiple modalities. In this work we showed an alternative fusion strategy within the Bayesian framework which significantly beats the earlier method. The framework described here is also based on sensor-selection and is most similar to Tresp [84], where the mixture of Gaussian Processes is described. The key differences include classification based on Gaussian Process rather than regression; also, we use Expectation Propagation for Gaussian Process classification and our classification likelihood is robust to labeling errors and noise. Our framework is also capable of quickly re-learning the classification given updated label associations. Further, we provide a complete Bayesian treatment of the problem rather than using a maximum-likelihood training.

Affect Recognition: A lot of research has been done to develop methods for inferring affective states. Many researchers have used static methods such as questionnaires, dialogue boxes, etc., which are easy to administer but have been criticized for being static and thus not able to recognize changes in affective states. A more dynamic and objective approach for sensing affect is via sensors such as cameras, microphones, wearable devices, etc. However, most of the work on affect recognition using the sensors focuses on deliberately expressed emotions (happy /sad /angry etc.) by actors, and not on those that arise in natural situations such as classroom learn-

Table 3.1: Fixed Rule Classifier Combination Methods.

Rule	Criteria
Sum	$p(t=1\|\mathbf{x}^{(1)}..\mathbf{x}^{(P)}) \propto \sum_{p=1}^{P} p(t=1\|\mathbf{x}^{(p)})$
Product	$p(t=1\|\mathbf{x}^{(1)}..\mathbf{x}^{(P)}) \propto \prod_{p=1}^{P} p(t=1\|\mathbf{x}^{(p)})$
Max	$p(t=1\|\mathbf{x}^{(1)}..\mathbf{x}^{(P)}) \propto \max_p p(t=1\|\mathbf{x}^{(p)})$
Min	$p(t=1\|\mathbf{x}^{(1)}..\mathbf{x}^{(P)}) \propto \min_p p(t=1\|\mathbf{x}^{(p)})$
Vote	$p(t=1\|\mathbf{x}^{(1)}..\mathbf{x}^{(P)}) \propto$ $\{ \begin{array}{ll} 1 & \text{if } \sum_{p=1}^{P} \lceil p(t=1\|\mathbf{x}^{(p)}) \rceil \geq \lceil \frac{P}{2} \rceil \\ 0 & \text{otherwise} \end{array} \}$

ing. In the context of learning there have been very few approaches for the purpose of affect recognition. Notable among them is Conati's [13] work on probabilistic assessment of affect in educational games. Also Mota and Picard [59] have described a system that uses dynamic posture information to classify different levels of interest in learning environments, which we significantly extend to the multimodal scenario.

Despite the advances in machine recognition of human emotion, much of the work on machine recognition of human emotion has relied on a single modality. Exceptions include the work of Picard et al.[65], which achieved 81% classification accuracy of eight emotional states of an individual over many days of data, based on four physiological signals, and several efforts that have combined audio of the voice with video of the face, e.g. Huang et al. [28], who combined these channels to recognize six different affective states. Pantic and Rothkrantz [64] provide a survey of other audio-video combination efforts and an overview of issues in building a multimodal affect recognition system.

3.2 Our Approach

The idea behind the proposed framework is to combine decisions from *experts* that operate upon different subsets of input features. If any of the features are missing then the final decision ignores the classifiers that act based on those missing features. There are a number of ways to combine decisions and table 3.1 mentions some of the popular methods. Our method is based on a mixture-of-experts framework and figure 3-1 shows the model we follow to solve the problem. The data $\mathbf{x}^i$, where $i \in \{1,..,P\}$ correspond to P different subsets of features, which we call *channels*, and each of the subsets generates a soft-decision y^p. The variable λ determines the channel that decides the final decision t. Note, that λ is never observed and we have to marginalize over λ to compute the final decision. Intuitively, this can be thought of as weighting individual decisions y^p appropriately and combining them to infer the distribution over the class label t. The weights, which correspond to a probability distribution over λ, depend upon the data point being classified and are determined using another meta-level decision system. Note, that figure 3-1 differs from the original GP classification formulation shown in figure 1-1(b), where there

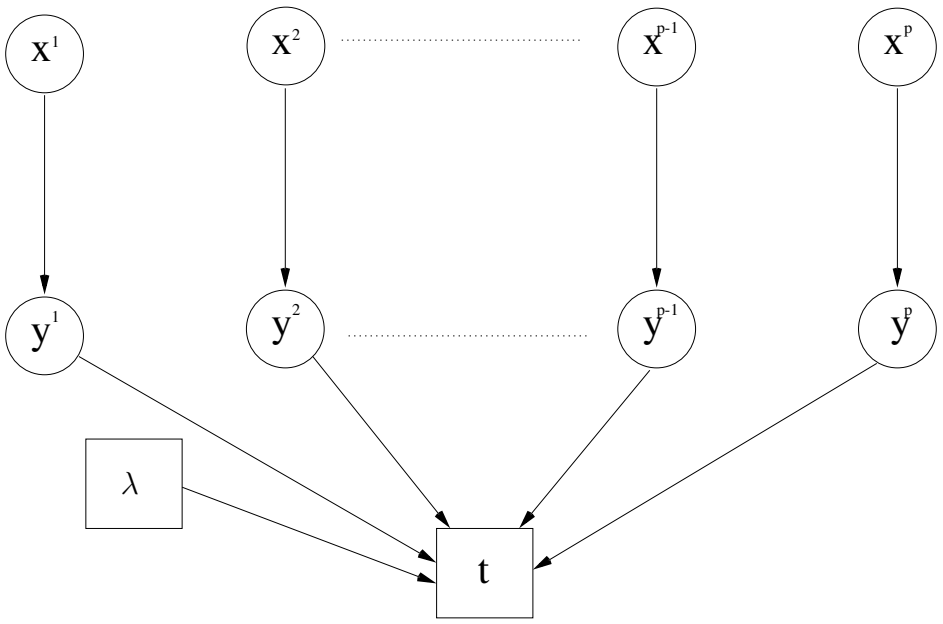


Figure 3-1: A Mixture of GPs for P channels

is a single hidden variable y conditioned on all the features. In the Mixture of GP approach we use multiple random variables y^i, where $i \in \{1, .., P\}$, which are then used to combine decisions based on different subsets.

While all the experiments on the affect dataset in this chapter assume that each channel corresponds to a single modality and is mutually exclusive from the rest of the channels (as described in section 3.3.1), it is possible more generally to have subsets of features that are based on many different modalities which are not mutually exclusive. The system described in this chapter uses Gaussian Process (GP) classification to first infer the probability distribution over y^p for all the channels. The final decision is gated through λ whose probability distribution conditioned on the test point is determined using another multi-label GP classification system.

We follow a Bayesian paradigm and the aim is to compute the posterior probability of an affective label of a test point given all the training data and the model. In section 3.2.1, we first review classification using the Gaussian Process (GP). Section 3.2.2 then extends the idea to a Mixture of Gaussian Processes and describes how to handle multiple modalities in the same Bayesian framework. Following, that we discuss the problem of detecting affective state of interest using multiple modalities and show experimental results, comparing the proposed new unified approach to other classifier combination techniques.

3.2.1 Gaussian Process Classification

GP classification is related to kernel machines such as Support Vector Machines (SVMs) and has been well explored in the machine learning community. Under the Bayesian framework, given a set of labeled data points $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_n\}$, with class

labels $\mathbf{t} = \{t_1, .., t_n\}$ and an unlabeled point $\mathbf{x}^*$, we are interested in the distribution $p(t^*|\mathbf{X}, \mathbf{t}, \mathbf{x}^*)$. Here t^* is a random variable denoting the class label for the point $\mathbf{x}^*$. Although, here we only describe how to classify one new point, all the machinery described applies as well to a set of new points without any additional computational overhead.

The idea behind GP classification is that the hard labels $\mathbf{t}$ depend upon hidden soft-labels $\mathbf{y} = \{y_1, ..., y_n\}$. These hidden soft-labels arise due to application of a function f directly on the input data points (i.e. $y_i = f(x_i)\ \forall i \in [1..n]$). Further, we assume a Gaussian Process prior on the function f; thus, the results $\mathbf{y}$ of the evaluation of the function f on any number of input data points $\mathbf{x}$ are jointly Gaussian. Further, the covariance between two outputs y_i and y_j can be specified using a kernel function applied to $\mathbf{x}_i$ and $\mathbf{x}_j$. Formally, $\{y_1, .., y_n\} \sim N(0, K)$ where K is a n-by-n kernel matrix with $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.

The observed labels $\mathbf{t}$ are assumed to be conditionally independent given the soft labels $\mathbf{y}$ and each t_i depends upon y_i through the conditional distribution:

$$p(t_i|y_i) = \Phi(\beta y_i \cdot t_i)$$

Here, $\Phi(z) = \int_{-\infty}^{z} \mathcal{N}(z; 0, 1)$ which provides a quadratic slack for labeling errors and the parameter β controls the level of slack. Note that we can use different noise models here (see chapter 2 for other noise models); however, for simplicity we only discuss the case with the probit noise model described above.

Our task is then to infer $p(t^*|D)$, where $D = \{\mathbf{X}, \mathbf{t}, \mathbf{x}^*\}$. Specifically:

$$p(t^*|D) = p(t^*|\mathbf{X}, \mathbf{t}, \mathbf{x}^*) \propto \int_{\mathbf{y}, y^*} p(t^*|\mathbf{y}, y^*) p(\mathbf{y}, y^*|\mathbf{X}, \mathbf{t}, \mathbf{x}^*) \tag{3.1}$$

Where the posterior $p(\mathbf{y}, y^*|\mathbf{X}, \mathbf{t}, \mathbf{x}^*)$ can be written as:

$$p(\mathbf{y}, y^*|\mathbf{X}, \mathbf{t}, \mathbf{x}^*) = p(\mathbf{y}, y^*|D) \propto p(\mathbf{y}, y^*|\mathbf{X}, \mathbf{x}^*) p(\mathbf{t}|\mathbf{y})$$

The term $p(\mathbf{y}, y^*|\mathbf{X}, \mathbf{x}^*) \sim N(0, K)$ is the GP prior and it enforces a smoothness constraint. The second term, $p(\mathbf{t}|\mathbf{y})$ incorporates information provided in the labels. In the framework described here, $p(\mathbf{y}, y^*|D)$ is approximated as a Gaussian distribution using Expectation Propagation (EP), a technique for approximate Bayesian inference [57]. Assuming conditional independence of labels given the soft-labels, $p(\mathbf{t}|\mathbf{y})$ can be written as:

$$p(\mathbf{t}|\mathbf{y}) = \prod_{i=1}^{n} p(t_i|y_i) = \prod_{i=1}^{n} \Phi(\beta y_i \cdot t_i)$$

The idea behind using EP is to approximate $P(\mathbf{y}, y^*|D)$ as a Gaussian. Although the prior $p(\mathbf{y}, y^*|\mathbf{X}, \mathbf{x}^*)$ is a Gaussian distribution, the exact posterior is not a Gaussian due to the form of $p(\mathbf{t}|\mathbf{y})$. Nonetheless, we can use EP to approximate the posterior

as a Gaussian. Specifically, the method approximates the terms $p(t_i|y_i)$ as:

$$p(t_i|y_i) \approx \tilde{t}_i = s_i \exp(-\frac{1}{2v_i}(y_i \cdot t_i - m_i)^2) \tag{3.2}$$

EP starts with the GP prior N(0,K) and incorporates all the approximate terms $\tilde{t}_i$ to approximate the posterior $p(\mathbf{y}, y^*|D) = N(\mathbf{M}, \mathbf{V})$ as a Gaussian. This is one key fact we will be exploiting in the mixture of Gaussian process framework, and as described in the next section, will allow us to perform Bayesian inference very efficiently.

To classify the test point $\mathbf{x}^*$, the approximate distribution $p(y^*|D) \approx N(M^*, V^*)$ can be obtained by marginalizing $p(\mathbf{y}, y^*|D)$ and then equation 3.1 can be used:

$$p(t^*|D) \propto \int_{y^*} p(t^*|y^*)N(M^*, V^*) = \Phi(\frac{M^* \cdot t^*}{\sqrt{(1+V^*)}}) \tag{3.3}$$

Here, $p(y^*|D) = N(y^*; M^*, V^*)$ and is obtained by marginalizing $p(\mathbf{y}, y^*|D)$.

3.2.2 Mixture of Gaussian Processes for Sensor Fusion

Given n data points $\bar{\mathbf{x}}_1, .., \bar{\mathbf{x}}_n$, obtained from P different sensors, our approach follows a mixture of Gaussian Processes model described in figure 3-1. Let every i^{th} data point be represented as $\bar{\mathbf{x}}_i = \{\mathbf{x}_i^{(1)}, .., \mathbf{x}_i^{(P)}\}$, and the soft labels as $\bar{\mathbf{y}}_i = \{y_i^{(1)}, .., y_i^{(P)}\}$. Given $\lambda_i \in \{1, .., P\}$, the random variable that determines the channels for the final classification, the classification likelihood can be written as:

$$P(t_i|\bar{\mathbf{y}}_i, \lambda_i = j) = P(t_i|y_i^{(j)}) = \Phi(\beta t_i \cdot y_i^{(j)})$$

Given a test point $\bar{\mathbf{x}}^*$, let $\bar{\mathbf{X}} = \{\bar{\mathbf{x}}_1, .., \bar{\mathbf{x}}_n, \bar{\mathbf{x}}^*\}$ denote all the training and the test points. Further, let $\bar{\mathbf{Y}} = \{\mathbf{y}^{(1)}, .., \mathbf{y}^{(P)}\}$, denote the hidden soft labels corresponding to each channel of all the data including the test point.

Let, $Q(\bar{\mathbf{Y}}) = \prod_{p=1}^{P} Q(\mathbf{y}^{(p)})$ and $Q(\mathbf{\Lambda}) = \prod_{i=1}^{n} Q(\lambda_i)$, denote the approximate posterior over the hidden variables $\bar{\mathbf{Y}}$ and $\mathbf{\Lambda}$, where $\mathbf{\Lambda} = \{\lambda_1, .., \lambda_n\}$ are the switches corresponding only to the n labeled data points. Let $p(\bar{\mathbf{Y}})$ and $p(\mathbf{\Lambda})$ be the priors with $p(\bar{\mathbf{Y}}) = \prod_{p=1}^{P} p(\mathbf{y}^{(p)})$, the product of GP priors and $p(\mathbf{\Lambda})$ uniform. Given $\bar{\mathbf{X}}$ and the labels $\mathbf{t}$, our algorithm aims to compute good approximations $Q(\bar{\mathbf{Y}})$ and $Q(\mathbf{\Lambda})$ to the real posteriors by iteratively optimizing the variational bound:

$$F = \int_{\bar{\mathbf{Y}}, \mathbf{\Lambda}} Q(\bar{\mathbf{Y}})Q(\mathbf{\Lambda}) \log(\frac{p(\bar{\mathbf{Y}})p(\mathbf{\Lambda})p(\mathbf{t}|\bar{\mathbf{X}}, \bar{\mathbf{Y}}, \mathbf{\Lambda})}{Q(\bar{\mathbf{Y}})Q(\mathbf{\Lambda})}) \tag{3.4}$$

The classification using EP is required only once, irrespective of the number of iterations. In each iteration to optimize the bound given in equation 3.4, the classification rules are updated using the Gaussian approximations provided by EP. The algorithm is shown in figure 3-2 and can be divided into 3 steps: initialization, optimization and

Given $\{\bar{\mathbf{X}}, \mathbf{t}\}$ and $\bar{\mathbf{x}}^*$

Step 1: Initialization

-For all the labeled points $i = 1$ to n do

· Initialize $Q(\lambda_i)$ using uniform distribution

-For all the modalities $p = 1$ to P do

· Incorporate all the labeled data points to obtain a Gaussian posterior for the soft labels:

$p^0(\mathbf{y}^{(p)}) = N(\mathbf{y}^{(p)}; \mathbf{M}_{\mathbf{y}^{(p)}}, \mathbf{V}_{\mathbf{y}^{(p)}})$

· Initialize: $Q(\mathbf{y}^{(p)}) = p^0(\mathbf{y}^{(p)})$

Step 2: Variational Updates

-Repeat until change in posteriors is less than some small threshold

· Update $Q(\mathbf{\Lambda})$ using equation 3.6.

· Update $Q(\bar{\mathbf{Y}})$ using equation 3.7.

Step 3: Classifying Test Data

-Compute $\hat{\mathbf{\Lambda}} = \arg\max_{\mathbf{\Lambda}} Q(\mathbf{\Lambda})$

-Use P-way classification to get the posterior $Q(\lambda^*)$

-Estimate $p(t^*|\bar{\mathbf{X}}, \mathbf{t})$ using equation 3.9

Figure 3-2: Summary of the algorithm to classify the test data point using a mixture of Gaussian Processes. This algorithm can be readily extended to more than one test points without any computational overhead.

classification, which are described below.

Step 1: Initialization:

In the first step, the approximate posterior $Q(\bar{\mathbf{Y}})Q(\mathbf{\Lambda}) = \prod_{p=1}^{P} Q(\mathbf{y}^{(p)}) \prod_{i=1}^{n} Q(\lambda_i)$ is initialized. Here, $Q(\lambda_i)$ are multinomial distributions and are initialized randomly using a uniform distribution. $Q(\mathbf{y}^{(p)})$ are normal distributions and to initialize them, we first use EP as described in section 3.2.1, considering all the data points irrespective of the state of the switches. EP results in the approximate Gaussian posteriors $p^0(\mathbf{y}^{(p)})$ $= N(\mathbf{y}^{(p)}; \mathbf{M}_{\mathbf{y}^{(p)}}, \mathbf{V}_{\mathbf{y}^{(p)}})$ for all $p \in \{1, .., P\}$, which are used to initialize $Q(\mathbf{y}^{(p)})$.A very useful byproduct of EP is the Gaussian approximations of the likelihoods, which would later be used to update our classification during the variational iterations in step 2.

Step 2: Optimization:

The bound in equation 3.4 is optimized by iteratively updating $Q(\bar{\mathbf{Y}})$ and $Q(\mathbf{\Lambda})$. Given the approximations $Q^k(\mathbf{\Lambda})$ and $Q^k(\bar{\mathbf{Y}})$ from the k^{th} iteration, $Q^{k+1}(\mathbf{\Lambda})$ and $Q^{k+1}(\bar{\mathbf{Y}})$ can be updated using variational updated rules [3]. Specifically, update rules for $Q(\lambda_i)$ and $Q(\mathbf{y}^{(p)})$ are as follows:

$$Q^{k+1}(\lambda_i) \propto \exp\{\int_{\bar{\mathbf{Y}}} Q^k(\bar{\mathbf{Y}}) \log p(t_i|\bar{\mathbf{Y}}, \lambda_i)\}$$

$$Q^{k+1}(\mathbf{y}^{(p)}) \propto \exp\{\int_{\mathbf{\Lambda}} Q^k(\mathbf{\Lambda}) \log p(\mathbf{y}^{(p)}) p(\mathbf{t}|\mathbf{y}^{(p)}, \mathbf{\Lambda})\}$$

The update for $Q(\lambda_i = p)$ can be written as:

$$Q^{k+1}(\lambda_i = p) \propto \exp\{\int_{y_i^{(p)}} Q^k(y_i^{(p)}) \log p(t_i|y_i^{(p)})\} \tag{3.5}$$

$$= \exp\{\int_{y_i^{(p)}} Q^k(y_i^{(p)}) \log(\Phi(t_i y_i^{(p)})\} \tag{3.6}$$

Equation 3.6 is intractable but can be computed efficiently by importance sampling using the 1-D Gaussian $Q^k(y_i^p)$ as a proposal distribution. More importantly, we have the Gaussian approximations from EP for the likelihood term:

$$p(t_i|y_i^{(p)}) \approx s_i^{(p)} \exp(-\frac{1}{2v_i^{(p)}}(y_i^{(p)} \cdot t_i - m_i^{(p)})^2)$$

It can be shown that the update rule for $Q(\mathbf{y}^{(p)})$ reduces down to:

$$Q^{k+1}(\mathbf{y}^{(p)}) \propto p(\mathbf{y}^{(p)}) \prod_{i=1}^{n} N(y_i^{(p)}; m_i^{(p)} \cdot t_i, \frac{v_i^{(p)}}{Q^k(\lambda_i)}) \tag{3.7}$$

This is just a product of Gaussians; thus, there is no need to rerun EP to estimate the new posterior over soft classifications. Note that $Q(\lambda_i)$ divides the variance, hence controlling the contribution of each labeled data point for different channels.

Step 3: Classification:

Once we have the posterior over the switches, $Q(\lambda_i)$ $\forall i \in [1..n]$, we first infer the switches for the test data $\bar{\mathbf{x}}^*$ using a meta-level GP based classification system. For this, we do a P-way classification using the GP algorithm described in 3.2.1 with $\hat{\mathbf{\Lambda}} = \arg\max_{\mathbf{\Lambda}} Q(\mathbf{\Lambda})$ as labels. Specifically, for an unlabeled point $\bar{\mathbf{x}}^*$, P different classifications are done where each classification provides us with q_r^*, the probability that channel $r \in \{1, .., P\}$ was chosen to classify $\bar{\mathbf{x}}^*$. The posterior $Q(\lambda^* = r)$ is then set to $\frac{q_r^*}{\sum_{p=1}^{P} q_p^*}$.

In our experiments, to perform this P-way classification, we clubbed all the channels together and used imputation by filling in -1 as observations for the modalities that were missing. We also tried the imputation scheme of filling in 0, but the performance was slightly worse. Note, that -1 here now represents the state of error in the observations and the meta-level decision system can take these errors into account to choose the best modalities for classification. Further, note that we are not limited to using all the channels clubbed together: various combinations of the modalities can be used including other indicator and contextual variables.

Once we have the posterior over the switch for the test data, $Q(\lambda^*)$, we can infer

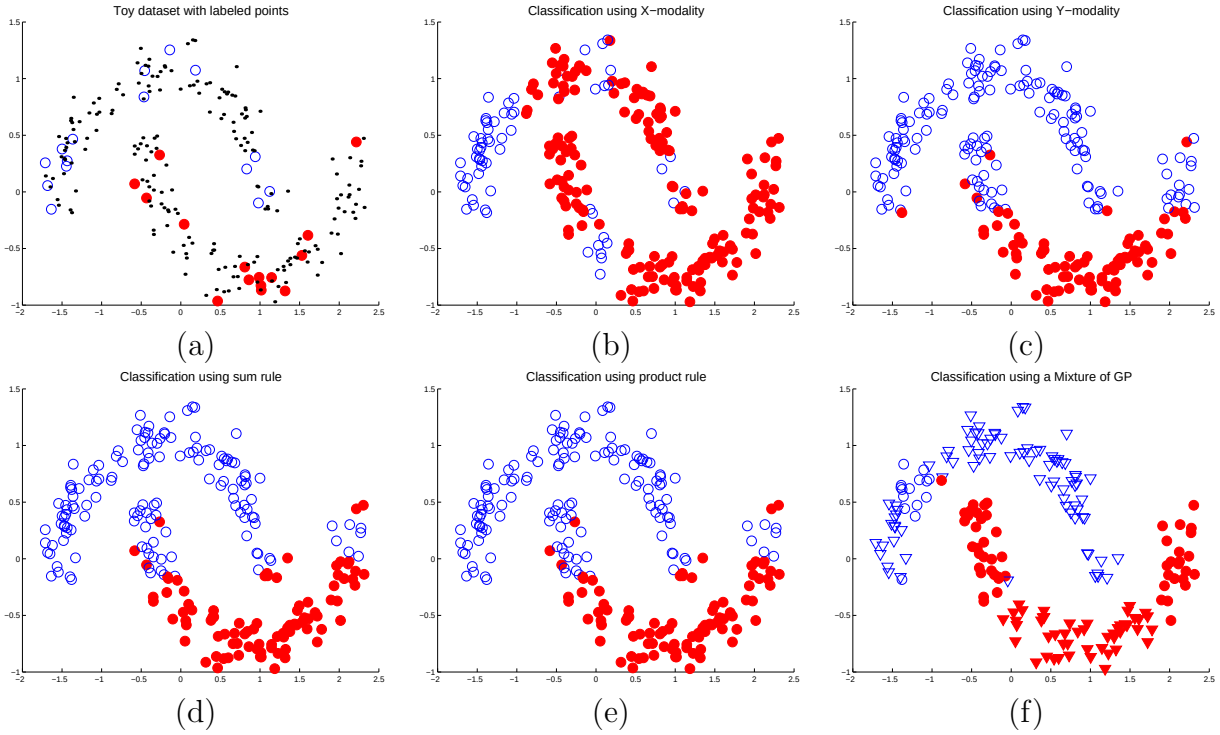


Figure 3-3: (a) Toy dataset with the labeled points highlighted, and classification results using (b) X-modality only, (c) Y-modality only, (d) sum rule, (e) product rule and (f) the Mixture of GP. The circles in (f) represent points classified with a greater weight on the X-modality and the triangles with a greater weight on the Y-modality.

the class probability of an unlabeled data $\bar{\mathbf{x}}^*$ using:

$$p(t^*|\bar{\mathbf{X}}, \mathbf{t}) = \int_{\bar{\mathbf{Y}}, \lambda^*} p(t^*|\bar{\mathbf{Y}}, \lambda^*) Q(\lambda^*) Q(\bar{\mathbf{Y}}) \tag{3.8}$$

$$= \sum_{p=1}^{P} Q(\lambda^* = p) \Phi\left(\frac{M^*_{y(p)} \cdot t^*}{\sqrt{1 + V^*_{y(p)}}}\right) \tag{3.9}$$

Here, $M^*_{y(p)}$ and $V^*_{y(p)}$ are the mean and the variance of the marginal Gaussian approximation for p^{th} channel corresponding to the hidden soft label $\bar{\mathbf{y}}^*$.

The main feature of the algorithm is that the classification using EP is required only once and the bound in equation 3.4 can be optimized very quickly using the Gaussian approximations provided by EP.

3.2.3 A Toy Example

We first demonstrate the features of the approach on a toy dataset and then apply it to the task of affect recognition using multiple modalities.

Toy Dataset: A toy dataset is shown in figure 3-3(a), which has been previously introduced by Zhou et al. [91]. The top and the bottom half moon correspond to two

different classes. The example shown in the figure has 15 labeled points from each class (30 total) and 100 test points (200 total). First, we perform two GP classifications using the method described in 3.2.1; one classifies the test points by just using the X-modality (dimension) and the other just using the Y-modality (dimension). Figures 3-3(b) & (c) show the results of these classifications using each individual modality, which is fairly poor. Figure 3-3(d) & (e) show classification using the sum and the product rule applied using the result of X and the Y classification. Finally, figure 3-3(f) shows successful classification using the Mixture of GP framework. In figure 3-3(f) the data points drawn as triangles were classified with a greater weight on the Y modality and the data points drawn as circles with a greater weight on the X-modality. We can see from the figure, that the final classification decision adapts itself according to the input space; thus, demonstrating the capability to perform sensor selection.

3.3 Multimodal Affect Recognition in Learning Environments

In this section, we present a framework to automatically extract, process and model sequences of natural occurring non-verbal behavior for recognizing affective states that occur during natural learning situations. The goal of the system is to classify affective states related to interest in children trying to solve a puzzle on a computer. The framework will be a component in computerized learning companions [10, 33] that could provide effective personalized assistance to children engaged in learning explorations and will also help in developing theoretical understanding of human behavior in learning situations.

The system uses real-time tracking of facial features and behaviors and monitors postures to extract relevant non-verbal cues. The extracted sensory information from the face, the postures and the state of the puzzle are combined using the mixture of Gaussian Process classifiers where classification using each channel is learned via Expectation Propagation [57]. The decision about the affective state is made by combining the beliefs of each individual expert using another meta-level classification system. We address pattern recognition in this multimodal scenario, which is marred by the problem of missing or bad data, as some of the algorithms and hardware that work on the individual modalities can fail, resulting in a significant reduction in the performance of the pattern recognition system.

This work tackles a number of challenging issues. While most of the prior work on emotion recognition has focused on posed emotions by actors, our emphasis is on naturally occurring non-verbal behaviors as it is crucial that the system is capable of dealing with the unconstrained nature of real data. Second, despite advances in face analysis and gesture recognition, the real-time sensing of non-verbal behaviors is still a challenging problem. In this work we demonstrate a multimodal system that can automatically extract non-verbal behaviors and features from face and postures, which can be used to detect affective states. No manual initialization is needed for

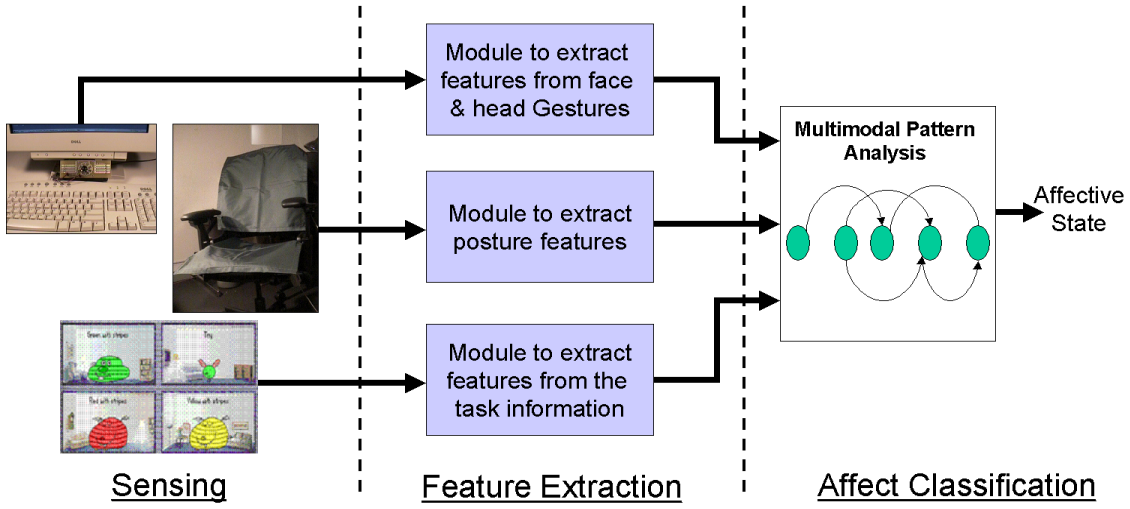


Figure 3-4: The overall architecture

this system to operate. Third, training a pattern classification system needs labeled data. Unlike the case of posed emotions, in natural data getting the ground truth of labels is a challenging task. We discuss how we can label the data reliably for different affective states.

3.3.1 System Overview

Figure 3-4 describes the architecture of the proposed system[2]. The non-verbal behaviors are sensed through a camera and a pressure sensing chair. The camera is equipped with Infrared (IR) LEDs for structured lighting that help in real-time tracking of pupils and extracting other features from the face. Similarly the data sensed through the chair is used to extract information about the postures. Features are extracted from the activity that the subject is doing on the computer as well, which are then sent to a multimodal pattern analyzer that combines all the information to predict the current affective state. In this work we focus on a scenario where children try to solve puzzles on a computer. The details of the tracking, the feature extraction and the other pre-processing steps are provided in appendix A.

The system can extract features in real time (27-29 fps for face and 8 fps for posture sensing) on a 1.8 GhZ Pentium 4 machine. The system tracks the facial features well as long as the subject is in the reasonable range of the camera. The system can detect whenever it is unable to find eyes in its field of view, which might occasionally happen due to large head and body movements. Also, sometimes the camera can only see the upper part of the face and cannot extract lower facial features, which happens if the subject leans forward. Due to these problems we often have missing information from

[2]We do not use skin conductance and the pressure mouse in the affect recognition work described in this chapter as these experiments were performed before the others sensors were integrated in the system. The other sensors can be incorporated into the framework without any difficulty.

the face; thus, we need an affect recognition system that is robust to such tracking failures.

Table 3.2 shows all the features that are extracted every one eighth of a second. We deliberately grouped them under "channels", separating for example the upper and lower face features because often the upper face features were present but not the lower. We observe that the cause of any missing feature or bad data is a sensor failure and the features tend to be missing in predictable groups; hence, for the implementation we form four different channels (table 3.2) and each channel corresponds to a group of features that can go missing simultaneously depending upon which sensor fails. Our strategy will be to fuse decisions from these channels, rather than individual features, thus, preserving some higher order statistics between different features.

Collecting the Database

Data collection for affective studies is a challenging task: The subject needs to be exposed to conditions that can elicit the emotional state in an authentic way; if we elicit affective states on demand, it is almost guaranteed not to bring out genuinely the required emotional state. Affective states associated with interest and boredom were elicited through an experiment with children aged 8 to 11 years, coming from relatively affluent areas of the state of Massachusetts in the USA. Each child was asked to solve a constraint satisfaction game called Fripples Place for approximately 20 minutes and the space where the experiment took place was a naturalistic setting allowing the subject to move freely. It was arranged with the sensor chair, one computer playing the game and recording the screen activity, having a monitor, standard mouse and keyboard, as well as two video-cameras: one capturing a side-view and one the frontal view; and finally, a Blue Eyes camera capturing the face. We made the cameras unobtrusive to encourage natural responses preserving as much as possible the original behavior. Given that we cannot directly observe the student's internal thoughts and emotions, nor can children in the age range of 8 and 11 years old reliably articulate their feelings, we chose to obtain labels of affective states from observers who are

Table 3.2: Extracted features from different modalities which are grouped into channels.

Channel 1: Upper Face
Brow Shape
Eye Shape
likelihood of nod
likelihood of shake
likelihood of blink

Channel 3: Posture
Current Posture
Level of Activity

Channel 2: Lower Face
Probability of Fidget
Probability of Smile

Channel 4: Game
Level of Difficulty
State of the Game

teachers by profession. We engaged in several iterations with teachers to ascertain a set of meaningful labels that could be reliably inferred from the data. The teachers were allowed to look at frontal video, side video, and screen activity, recognizing that people are not used to looking at chair pressure patterns. Eventually, we found that teachers could reliably label the states of high, medium and low interest, bored, and a state that we call "taking a break," which typically involved a forward-backward postural fidget and sometimes stretching. Working separately and without being aware of the final purpose of the coding task, teachers obtained an average overall agreement (Cohen's Kappa) of 78.6%. In this work, we did not use data classified as "bored" or "other" even though teachers identified them consistently. The bored state was dropped since teachers only classified very few episodes as bored, and this was not enough to develop separate training and test sets. The final database used to train and test the system included 8 different children with 61 samples of "high interest," 59 samples of "low interest" and 16 samples of "taking a break". We only look at the binary classification problem of classifying "high interest" (61 samples) from the rest (16 + 59 = 75 samples). Each of the samples is a maximum of 8 seconds long with observations recorded at 8 samples per second. Only 50 samples had features present from all the four modalities, whereas the other 86 samples had the face channel missing.

3.3.2 Empirical Evaluation

Table 3.3: Recognition rates (standard deviation in parenthesis) averaged over 100 runs.

	GP	**SVM**
Upper Face	66.81%(6.33)	69.84%(6.74)
Lower Face	53.11%(9.49)	57.06%(9.16)
Posture	81.97%(3.67)	82.52%(4.21)
Game	57.22%(4.57)	58.85%(5.99)
Mixture of GP	**86.55%(4.24)**	-

We performed experiments on the collected database using the framework to classify the state of high interest (61 samples) vs. uninterest (75 samples). The experimental methodology was to use 50% of the data for training and the rest for testing. Besides the comparison with the individual modalities, we also compare the Mixture of GP with the HMM based expert-critic scheme [36] and a naive feature level fusion with imputation. In the naive feature level fusion, the observations from all the channels are stacked to form a big vector and these vectors of fused observations are then used to train and test the classifiers. However, in our case this is not trivial as we have data with missing channels. We test a naive feature level fusion based on a simple imputation scheme where we use -1 as a value of all those observations that are missing, thus, fusing all the channels into one single vector. As previously

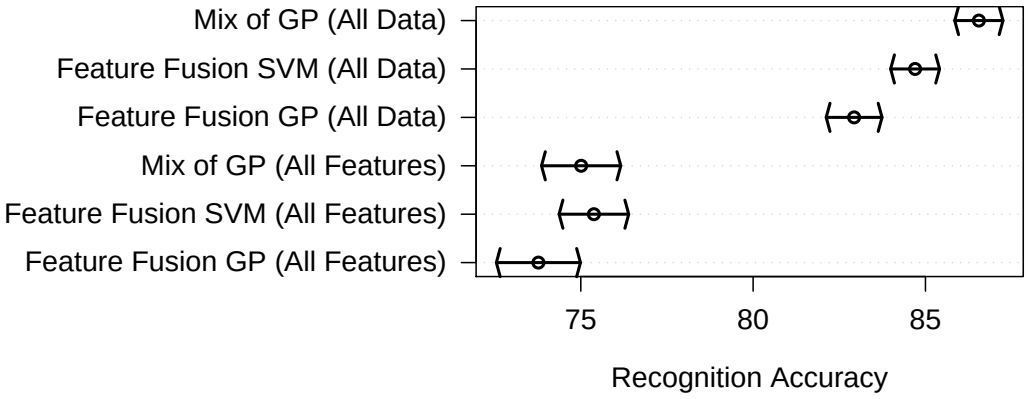


Figure 3-5: Performance comparison of the proposed Mixture of GP approach on the affect dataset with naive feature level fusions (imputations) with and without the incomplete data in the training set. Each point was generated by averaging over 100 runs. Non overlapping of error bars, the standard errors scaled by 1.64, indicates 95% significance of the performance difference. This figure suggests that it is advantageous to incorporate incomplete data in the training set. Further, mix of GP has the maximum gain as it can choose channels to classify depending upon the occurrence of incompleteness.

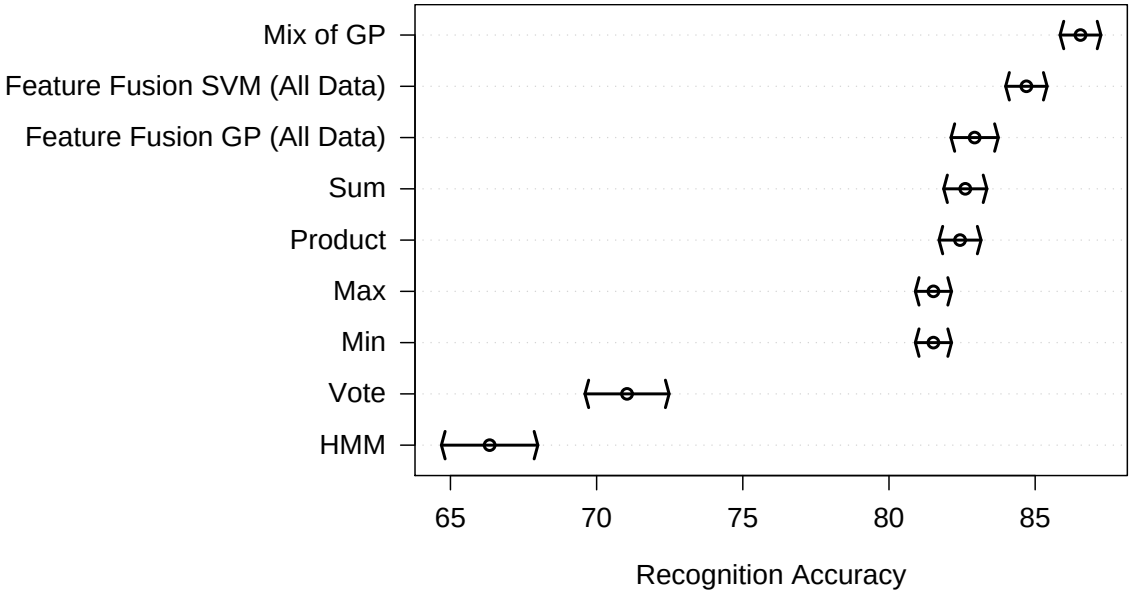


Figure 3-6: Performance comparison of the proposed Mixture of GP approach on the affect dataset with different classifier combination strategies like naive feature level fusions (imputations), fixed rules and the HMM based expert-critic framework. Each point was generated by averaging over 100 runs. Non overlapping of error bars, the standard errors scaled by 1.64, indicates 95% significance of the performance difference.

mentioned, using -1 for missing features can be thought of as observations indicating errors in sensing; hence, providing some extra information about the context.

All the experiments were done using GP classification as the base classifiers and for completeness we also perform comparisons with SVMs. Both, GP classification and SVMs used Radial Basis Function (RBF) kernels and the hyperparameters for GP classification (the kernel width σ and the noise model parameter β) were selected by evidence maximization. For SVMs we used the leave-one-out validation procedure to select the penalty parameter C and the kernel width σ. We randomly selected 50% of the points and computed the hyper-parameters for both GPs and SVMs for all individual modalities and the naive feature level fusions. This process was repeated five times and the mean values of the hyperparameters were used in our experiments. We also evaluate the performance of other classifier combination schemes by training SVMs and GP classifiers on the complete data. These standard classifier combination schemes are shown in Table 3.1.

Is it advantageous to combine multiple modalities? Table 3.3 shows the results for individual modalities using GP classification and SVMs. These numbers were generated by averaging over 100 runs and we report the mean and the standard deviation. We can see that the posture channel can classify the modalities best, followed by features from the upper face, the game and the lower face. Although, the performance obtained using GP classification is similar to SVMs and slightly worse for upper and the lower face, we find that extension of GP to a mixture boosts the performance and leads to significant gains over SVMs. The better performance of SVMs can be explained by the fact we used a leave-one-out cross validation for hyperparameter selection in SVMs, which is a robust measure than evidence maximization used for GP [66]. However, we will see in chapter 4 that in semi-supervised cases where there are very few labeled data points available evidence maximization might be preferable.

Does including incomplete data in the training set help? We compare the Mixture of GP to feature level fusion while restricting the training and testing database to the points where all the channels are available. The restriction results in a significant decrease in the available data (only 50 total data points) and the mix of GP obtained an average accuracy of 75.01% $\pm$ 1.15 over 100 runs, whereas the accuracy using feature level fusion was 73.77%$\pm$1.22. When we perform naive feature level fusion using all the data, without any restriction and using -1 for the channels that were missing, we get a significant gain in average accuracy to 82.93%$\pm$0.81. However, the Mixture of GP outperforms all of these significantly with an average accuracy of 86.55%$\pm$0.55 (see figure 3-5 for significance). Figure 3-7(a) and (b) show that the Mixture of GP not only beats feature level fusion on the subset of data where all channels are present, but also significantly beats it when incomplete data can be used. Similar results are obtained comparing to SVMs and are graphically shown in figure 3-7 (c) and (d).

Which combination scheme performs best? Figure 3-5 graphically demonstrates the performance gain obtained by the Mixture of GP approach over the feature level fusions and the HMM based expert critic framework as implemented in [36] with FACS features. The points in figure 3-5 were generated by averaging over 100 runs.

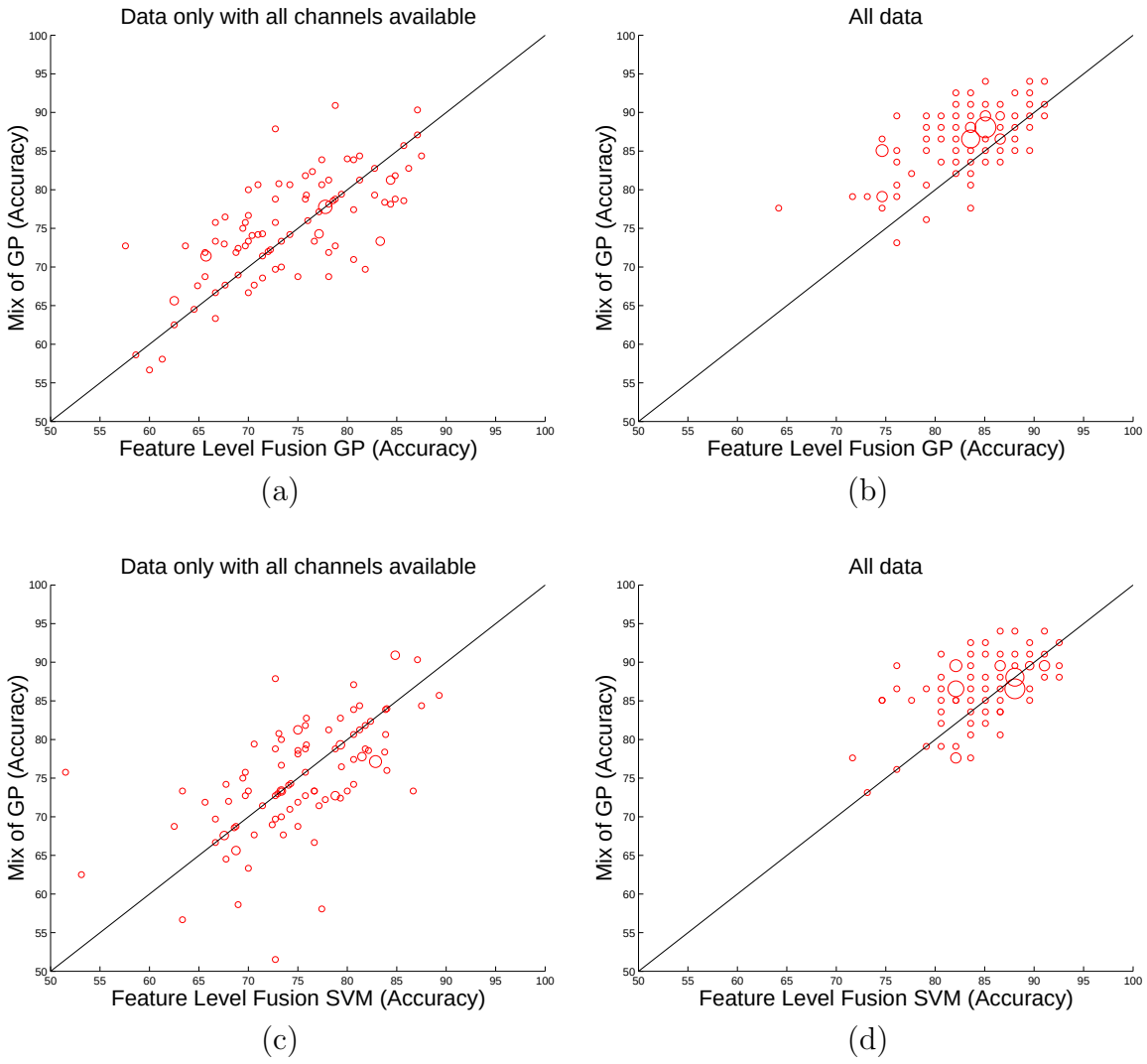
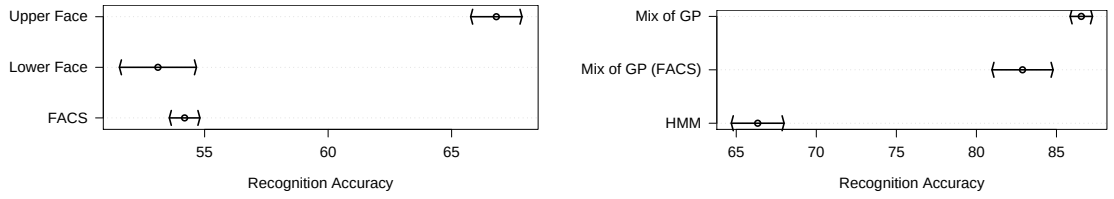


Figure 3-7: Plots comparing accuracy of Mixture of GP with GP (a)-(b) and SVM (c)-(d) where the latter methods use feature fusion. In (a) and (c) only the subset of data where all modes are intact is used. In (b) and (d) all the data is used, as described in the text. There are 100 points on each graph and each point is (accuracy SVM/GP, accuracy Mixture of GP) and corresponds to one test run. Circle width is proportional to the number of points having that coordinate. Points above the diagonal indicate where Mixture of GP was more accurate. While Mixture of GP is better (a) or comparable (c) on average when all the modes are intact, it is particularly better when there are noisy and missing channels (b) and (d).



(a) (b)

Figure 3-8: (a) Comparison of the recognition accuracy obtained while performing GP classification using the automatically extracted features from the lower and upper face with the accuracy obtained using manually coded Facial AUs. (b) Comparison of the recognition accuracy obtained by the Mixture of GP that uses automatically extracted facial features with Mixture of GP approach that uses manual FACS coding of AUs and the expert-critic HMM framework. Each point was generated by averaging over 100 runs with random 50-50 training and testing split. Non overlapping of error bars, the standard errors scaled by 1.64, indicates 95% significance of the performance difference.

The non-overlapping error bars, standard errors scaled by 1.64, signify 95% confidence in performance difference. The Mixture of GP uses all the data and can handle the missing information better than naive fusion methods; thus, it provides a significant performance boost.

FACS vs. automatically extracted features. In our earlier work [36, 32], we had used manually encoded FACS based Action Units (AU) as features extracted from the face. Figure 3-8(a) shows the improvement in performance obtained by using the features extracted from lower and upper face as described in section A.1. The accuracy of 66.81 ± 6.33 obtained while performing GP classification with automatically extracted upper facial features was significantly better than the accuracy of 54.19 ± 3.79 obtained with GP classification that used the manually coded upper AUs. Also, from figure 3-8(b) we can see that similar gains are observed when using the Mixture of GPs framework with the automatically extracted features. The difference is due to two factors. First, the set of automatically extracted upper facial features is richer than the AUs. Second, the AUs were manually encoded by just one FACS expert, thus, resulting in features prone to noise.

3.4 Conclusions and Future Work

In this chapter, we proposed a unified approach using a mixture of Gaussian Processes for achieving sensor fusion under the challenging conditions of missing channels. The incompleteness due to missing features is handled by combining the decisions of experts trained on subsets of features, where the decision fusion ignores the classifiers that act on the missing features. We provide an algorithm for Bayesian inference designed with a fast update of classification decisions based on variational and Gaussian approximations. On the task of classifying affective state of interest using information from face, postures and task information, the Mixture of GP method outperforms

feature level fusion based on imputation and several standard classifier combination schemes and obtained a recognition rate of 86%. Further, we also demonstrated that there are significant gains when the incomplete data is incorporated in the training set.

Future work includes incorporation of active learning and application of this framework to other challenging problems with limited labeled data. Note, that we preselected the subsets of features for the Mixture of GP framework and the subsets were mutually exclusive. However, the features across different modalities can be correlated and it can be advantageous to look at the subsets of features from different modalities where these subsets can even be overlapping. A direction for the future would be to discover these sets automatically. Further, there are interesting possibilities where the Mixture of GPs model is combined with different noise-models, semi-supervised classification and other extensions proposed in these thesis. Finally, there are many applications in different domains such as sensor networks, social networks and other multimodal scenarios where the proposed method can be used effectively.

Chapter 4

Gaussian Process Classification with Partially Labeled Data

Often in machine learning[1] problems there are a large number of data points of which few are labeled. This usually happens as it might be very expensive or difficult to obtain the labels. The incompleteness resulting from partially labeled data has been well researched and is often referred to as semi-supervised classification. Specifically, given a set of data points, among which few are labeled, the goal in semi-supervised learning is to predict the labels of the unlabeled points using the *complete* set of data points. By looking at both the labeled and the unlabeled data, the learning algorithm can exploit the distribution of data points to learn the classification boundary.

The original Gaussian process formulation cannot be directly used to exploit the information from the unlabeled data. This point has been elaborated earlier by Seeger [74] and more recently by Lawrence and Jordan [49]. The GP prior just encodes the assumption that similar data points should have the same label and itself does not contain any information about the data distribution. Further, the likelihood term only incorporates information from the labeled data; thus, the traditional Gaussian process classification cannot be trivially extended to the semi-supervised scenario.

Lawrence and Jordan [49] proposed to overcome this problem by introducing the null category noise model. The idea behind this approach is to use the unlabeled data to steer the decision boundary to the region of low data density. Specifically, they propose to modify the likelihood such that the model imitates the SVM loss function and favors the decision boundary with large margins. Now, the algorithm can also look at the unlabeled data points and exploit the information from the distribution of the data points to find the solution.

The work presented here explores other alternatives besides the null category noise model. Specifically, rather than focusing on the noise model, we aim to study the extensions of the standard Gaussian process prior by introducing regularization based on the unlabeled and the labeled data points. We focus mainly on graph based semi-supervised classification techniques, where the key idea is to exploit the notion of similarity to label the unlabeled points. Given a set of labeled and unlabeled points,

[1]The work described in this chapter appears in Neural Information Processing Systems, 2005 [37]

these approaches provide a way to regularize the solution using a similarity graph over the data points. Smola and Kondor [79] have provided a unifying view of many of these approaches using the theory of Reproducing Kernel Hilbert Space. Moreover, Zhu et al. [94] have also made connections between Gaussian process classification and semi-supervised learning work based on harmonic functions and Gaussian fields [92]. Similarly, there is recent work in semi-supervised learning that uses ideas behind manifold regularization in a Gaussian process framework [78]. This chapter aims to provide a Bayesian perspective to unify many graph-based semi-supervised classification methods; thus, providing a framework to analyze the similarities and the differences among most of the graph based semi-supervised methods.

One of the main problems with many of the existing graph based semi-supervised classification techniques is that of hyperparameter learning: performance depends greatly on the hyperparameters of the similarity graph, transformation of the graph Laplacian and the noise model. Figure 4 graphically shows this problem. In this example, we use an RBF kernel with the parameter σ to measure similarity between two data points. The figures show results using a recent graph based algorithm (LLGC [91]) and as we can see the results can be quite different for different values of the hyperparameter σ. Finding a correct set of hyperparameters is critical for the graph based methods to work well.

There are many advantages of viewing semi-supervised learning methods from the Bayesian perspective. Evidence maximization provides an elegant framework to do model selection and we provide a way to learn the kernel and hyperparameters for graph based semi-supervised classification, while adhering to a Bayesian framework.

We exploit the Bayesian framework for learning hyperparameters for graph-based semi-supervised classification. Given some labeled data, which can contain inaccurate labels, we pose the semi-supervised classification as an inference problem over the unknown labels. We use the evidence to simultaneously tune the hyperparameters that define the structure of the similarity graph, the parameters that determine the transformation of the graph Laplacian, and any other parameters of the model. Closest to our work is Zhu et al. [94], where they proposed a Laplace approximation for learning the edge weights. We use Expectation Propagation (EP) for approximate Bayesian inference that provides better approximations than Laplace. An additional contribution is an EM algorithm to learn the hyperparameters for the edge weights, the parameters of the transformation of the graph spectrum. More importantly, we explicitly model the level of label noise in the data, while [94] does not do. We provide what may be the first comparison of hyperparameter learning with cross-validation on state of the art algorithms (LLGC [91] and harmonic fields [92]).

This chapter is organized as follow: First, we mention some existing approaches and ideas that aim to address semi-supervised classification. Followed by that, we describe the proposed framework, the hyperparameter learning procedure and also discuss extensions to classify new unseen points that were not part of either the labeled or unlabeled data used in the Bayesian inference. After that, we describe experimental evaluation and discuss connections of the proposed framework to Gaussian processes and some of the recent work on graph-based semi-supervised classification. We end the chapter with discussion and a summary.

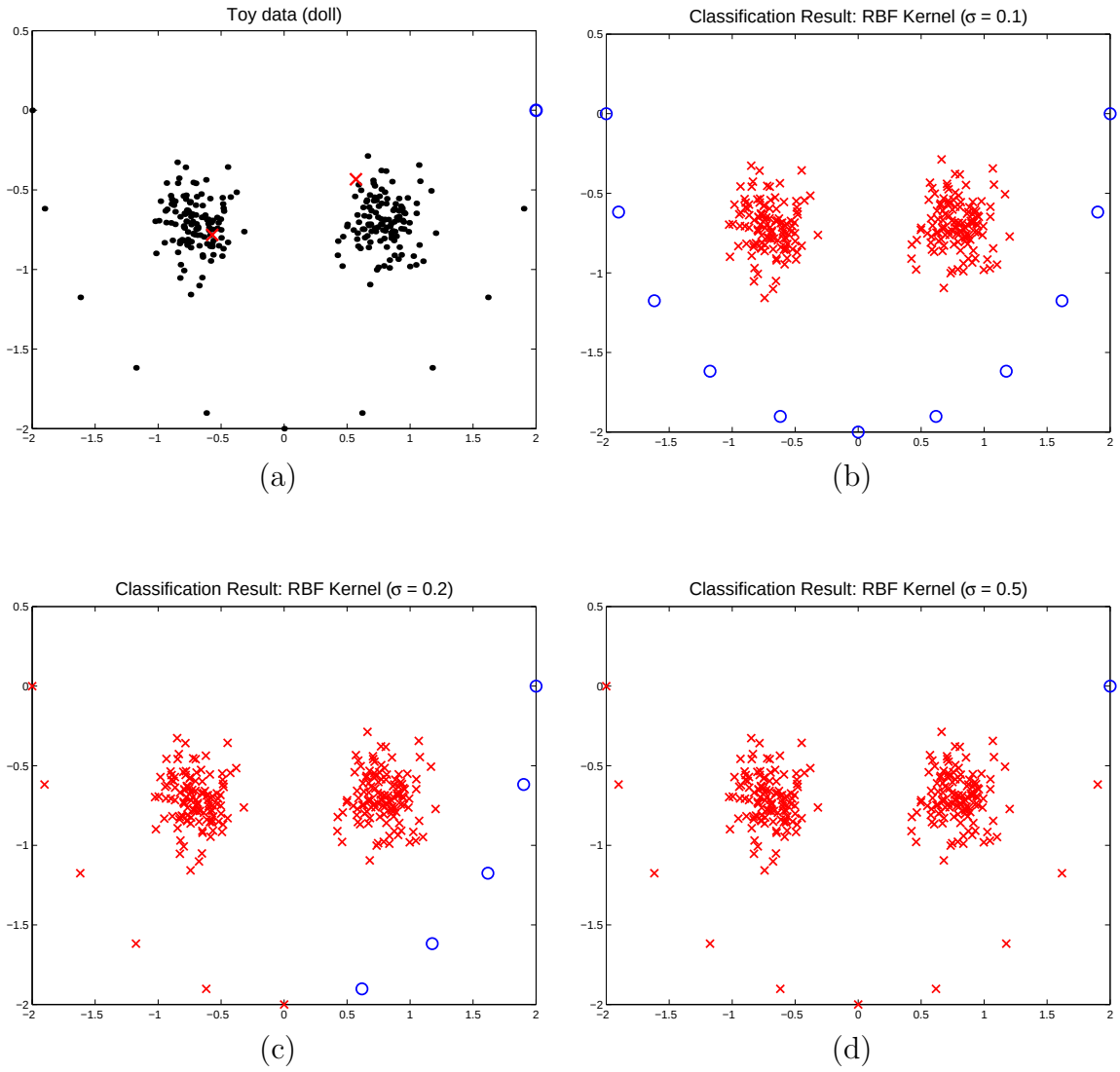


Figure 4-1: Figures demonstrating the effect of kernel width on a state of the art graph-based semi-supervised classification algorithm, LLGC [91]. (a) The toy data set: two clusters are class 1 and the half-ring is class 2. Classification results obtained by LLGC with different RBF kernel widths: (b) σ=0.1 (c) σ=0.2 and (d) σ=0.5. A bad choice of kernel width can deteriorate the classification performance.

4.1 Previous Work

4.1.1 Semi-supervised Classification

There have been a large number of approaches proposed in recent years for semi-supervised learning. The spectrum of the approaches spans generative models, support vector machines (SVMs), graph-based techniques, co-training, information regularization etc. Here, we attempt to cover a few of these approaches and the concepts behind them.

Generative Models: First, there have been a lot of work on semi-supervised learning based on a generative paradigm of classification using EM [61]. The idea behind these approaches is that the labels corresponding to the unlabeled points are considered as hidden data and the method iterates between estimating these hidden labels and refining the decision boundary. One of the main drawbacks of these approaches is that first these methods require a large database to perform well. Second, there has been evidence [12, 15] that the semi-supervised methods might perform poorly when the data deviates from the model assumptions (such as Gaussianity).

Transductive SVMs and Related Approaches: One of the more prominent approaches for semi-supervised classification from the perspective of discriminative modeling has been the transductive SVM [86]. This approach is explicitly based on the assumption that the decision boundary lies in the region where the data density is low. Transductive SVMs extend the regular SVM formulation to incorporate the information from the unlabeled data points, such that the decision boundary is steered away from the region of high density. There have been many successful applications that uses Transductive SVMs. Recently, Lawrence and Jordan [49] proposed a Bayesian approach that can be considered as a Bayesian counterpart to the Transductive SVMs. The approach suggests a null category noise model, which imitates the role of SVM hinge loss by penalizing the decision boundaries that lie in a high density region. Both of these methods utilize a noise model to achieve the semi-supervised classification. Specifically, the assumption that the decision boundary lies in a region of low density guides both of these algorithms that, as we will see, is different than the assumptions used in graph based semi-supervised classification techniques.

Regularization on Graphs: Many of the approaches proposed recently are very closely connected and are either implicitly or explicitly based on regularization on graphs [79]. The key idea behind these methods is similarity. Essentially, the idea is to put a smoothness constraint on the labels of the data points, such that the points that are similar and lie on the same structure are likely to have the same label. The smoothness constraint is obtained by creating a graph with the labeled and unlabeled data points as the vertices and with edge weights encoding the similarity between the data points. The semi-supervised classification problem is usually formulated as an optimization function where the goal is to obtain a labeling of the vertices that is both smooth over the graph and consistent with the labeled data. This optimization has been carried out using a number of different techniques such as, graph mincuts [6] and Tikhonov regularization [4, 5], and the objective to be optimized is based on analogies drawn from random walks [53, 91], electric networks [92] and spectral

methods [31]. Some of the work on semi-supervised learning has proposed kernels [5, 11, 43], which are motivated based on regularization on graphs and can be used in any kernel based learning algorithm.

Other Methods: There are approaches such as co-training [7], information regularization [82] and tree based Bayes [39]. The idea behind co-training is to iteratively train a pair of classifiers based on features which are assumed to be conditionally independent given the class label. Specifically, each classifier "co-trains" the other classifier by labeling the unlabeled points. Seeger [74] has mentioned connections of co-training to the input dependent regularization. Information regularization [14, 82] exploits the assumption that it is more likely for the data points in the regions of high data density to have the same labels. In the tree based Bayes method [39] the idea is to induce similarities based on evolutionary trees. Specifically, two data points, which are represented within a tree, are similar if they lie closer in the tree. This approach can be considered a specific case of the input-dependent regularization.

We limit the scope of our work to the graph based semi-supervised classification only. Note, that in all of the graph based methods the smoothness constraint over the labels depends highly upon the input examples (both labeled and unlabeled) leading to input dependent regularization.This is very different from the traditional transductive SVMs as well as the approach proposed by Lawrence and Jordan [49], where there is no input dependent regularization.

4.1.2 Hyperparameter and Kernel Learning

The performance of most of the graph-based algorithms depends upon the edge weights of the graph, which correspond to the similarity between the vertices. The notion of similarity is captured using a kernel matrix and the performance is highly dependent upon the hyper-parameters that describe the kernel. Further, often the smoothness constraints on the labels are imposed using a transformation of the graph Laplacian [79] and the parameters of the transformation affect the performance. Finally, there might be other parameters in the model, such as parameters to address label noise in the data. Finding a right set of parameters is a challenge, and usually the method of choice is cross-validation, which can be prohibitively expensive for real-world problems and problematic when we have few labeled data points.

The hyperparameter learning is a hard problem. Most of the methods ignore the problem of learning hyperparameters that determine the similarity graph and there are only a few approaches that address this problem. Zhu et al. [93] propose an approach based on semi-definite programming that learns a non-parametric transformation of the graph Laplacians using kernel alignment. This approach assumes that the similarity graph is already provided; thus, it does not address the learning of edge weights. Other approaches include the average label entropy [92] and the use of evidence-maximization using the Laplace approximation [94]. Our work is closely related to Zhu et al. [94]. The main differences are that we use EP to approximate the model evidence and EM to learn all the hyperparameters. The approximations using EP has been shown to be better than the Laplace approximation for the purpose of GP classification [46]; thus, providing a significant advantage over the previous

method. We describe our method next.

4.2 Bayesian Semi-Supervised Learning

We assume that we are given a set of data points $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_{n+m}\}$, of which $\mathbf{X}_L = \{\mathbf{x}_1, .., \mathbf{x}_n\}$ are labeled as $\mathbf{t}_L = \{t_1, .., t_n\}$ and $\mathbf{X}_U = \{\mathbf{x}_{n+1}, .., \mathbf{x}_{n+m}\}$ are unlabeled. Throughout this chapter we limit ourselves to two-way classification, thus $t \in \{-1, 1\}$. Our model assumes that the hard labels t_i depend upon hidden soft-labels y_i for all i. Given the dataset $D = [\{\mathbf{X}_L, \mathbf{t}_L\}, \mathbf{X}_U]$, the task of semi-supervised learning is then to infer the posterior $p(\mathbf{t}_U|D)$, where $\mathbf{t}_U = [t_{n+1}, .., t_{n+m}]$. The posterior can be written as:

$$p(\mathbf{t}_U|D) = \int_{\mathbf{y}} p(\mathbf{t}_U|\mathbf{y})p(\mathbf{y}|D) \tag{4.1}$$

In this work, we propose to first approximate the posterior $p(\mathbf{y}|D)$ and then use (4.1) to classify the unlabeled data. Using the Bayes rule we can write:

$$p(\mathbf{y}|D) = p(\mathbf{y}|\mathbf{X}, \mathbf{t}_L) \propto p(\mathbf{y}|\mathbf{X})p(\mathbf{t}_L|\mathbf{y})$$

The term, $p(\mathbf{y}|\mathbf{X})$ is the prior. It enforces a smoothness constraint and depends upon the underlying data manifold. Similar to the spirit of graph regularization [79] we use similarity graphs and their transformed Laplacian to induce priors on the soft labels $\mathbf{y}$. The second term, $p(\mathbf{t}_L|\mathbf{y})$ is the likelihood that incorporates the information provided by the labels.

In this work, $p(\mathbf{y}|D)$ is inferred using Expectation Propagation, a technique for approximate Bayesian inference [57]. Note, that similar formulation has also been proposed by Sindhwani et al. [78]. In the following subsections first we describe the prior and the likelihood in detail and then we show how evidence maximization can be used to learn hyperparameters and other parameters in the model.

4.2.1 Priors and Regularization on Graphs

The prior plays a significant role in semi-supervised learning, especially when there is only a small amount of labeled data. The prior imposes a smoothness constraint and should be such that it gives higher probability to the labelings that respect the similarity of the graph.

The prior, $p(\mathbf{y}|\mathbf{X})$, is constructed by first forming an undirected graph over the data points. The data points are the nodes of the graph and edge-weights between the nodes are based on similarity. This similarity is usually captured using a kernel. Examples of kernels include RBF, polynomial etc. Given the data points and a kernel, we can construct an $(n+m) \times (n+m)$ kernel matrix K, where $K_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$ for all $i \in \{1, .., n+m\}$.

Lets consider the matrix $\tilde{K}$, which is same as the matrix K, except that the diagonals are set to zero. Further, if G is a diagonal matrix with the diagonal entries

$G_{ii} = \sum_j \tilde{K}_{ij}$, then we can construct the combinatorial Laplacian ($\Delta = G - \tilde{K}$) or the normalized Laplacian ($\tilde{\Delta} = I - G^{-\frac{1}{2}}\tilde{K}G^{-\frac{1}{2}}$.) of the graph. For brevity, in the text we use Δ as a notation for both the Laplacians. Both the Laplacians are symmetric and positive semidefinite. Consider the eigen decomposition of Δ where $\{\mathbf{v}_i\}$ denote the eigenvectors and $\{\lambda_i\}$ the corresponding eigenvalues; thus, we can write $\Delta = \sum_{i=1}^{n+m} \lambda_i \mathbf{v}_i \mathbf{v}_i^T$. Usually, a transformation $r(\Delta) = \sum_{i=1}^{n+m} r(\lambda_i)\mathbf{v}_i\mathbf{v}_i^T$ that modifies the spectrum of Δ is used as a regularizer. Specifically, the smoothness imposed by this regularizer prefers soft labeling for which the norm $\mathbf{y}^T r(\Delta)\mathbf{y}$ is small. Equivalently, we can interpret this probabilistically as following:

$$p(\mathbf{y}|\mathbf{X}) \propto e^{-\frac{1}{2}\mathbf{y}^T r(\Delta)\mathbf{y}} = N(0, r(\Delta)^{-1}) \tag{4.2}$$

Where $r(\Delta)^{-1}$ denotes the pseudo-inverse if the inverse does not exist. Equation (4.2) suggests that the labelings with the small value of $\mathbf{y}^T r(\Delta)\mathbf{y}$ are more probable than the others. Note, that when $r(\Delta)$ is not invertible the prior is improper. The fact that the prior can be written as a Gaussian is advantageous as techniques for approximate inference can be easily applied. Also, different choices of transformation functions lead to different semi-supervised learning algorithms. For example, the approach based on Gaussian fields and harmonic functions (Harmonic) [92] can be thought of as using the transformation $r(\lambda) = \lambda$ on the combinatorial Laplacian without any noise model. Similarly, the approach based in local and global consistency (LLGC) [91] can be thought of as using the same transformation but on the normalized Laplacian and a Gaussian likelihood. Therefore, it is easy to see that most of these algorithms can exploit the proposed evidence maximization framework. In the following we focus only on the parametric linear transformation $r(\lambda) = \lambda + \delta$. Note that this transformation removes zero eigenvalues from the spectrum of Δ.

4.2.2 The Likelihood

Assuming conditional independence of the observed labels given the hidden soft labels, the likelihood $p(\mathbf{t}_L|\mathbf{y})$ can be written as $p(\mathbf{t}_L|\mathbf{y}) = \prod_{i=1}^{n} p(t_i|y_i)$. The likelihood models the probabilistic relation between the observed label t_i and the hidden label y_i. Many real-world datasets contain hand-labeled data and can often have labeling errors. While most people tend to model label errors with a linear or quadratic slack in the likelihood, it has been noted that such an approach does not address the cases where label errors are far from the decision boundary [40]. The flipping likelihood can handle errors even when they are far from the decision boundary and can be written as:

$$p(t_i|y_i) = \epsilon(1 - \Phi(y_i \cdot t_i)) + (1 - \epsilon)\Phi(y_i \cdot t_i) = \epsilon + (1 - 2\epsilon)\Phi(y_i \cdot t_i) \tag{4.3}$$

Here, Φ is the step function, ϵ is the labeling error rate and the model admits possibility of errors in labeling with a probability ϵ. This likelihood has been earlier used in the context of Gaussian process classification [40, 63]. The above described likelihood explicitly models the labeling error rate; thus, the model should be more

robust to the presence of label noise in the data. The experiments in this chapter use the flipping noise likelihood shown in (4.3).

4.2.3 Approximate Inference

In this work, we use EP to obtain a Gaussian approximation of the posterior $p(\mathbf{y}|D)$. Although, the prior derived in section 4.2.1 is a Gaussian distribution, the exact posterior is not a Gaussian due to the form of the likelihood. We use EP to approximate the posterior as a Gaussian and then equation (4.1) can be used to classify unlabeled data points. EP has been previously used [57] to train a Bayes Point Machine, where EP starts with a Gaussian prior over the classifiers and produces a Gaussian posterior. Our task is very similar and we use the same algorithm. In our case, EP starts with the prior defined in (4.2) and incorporates likelihood to approximate the posterior $p(\mathbf{y}|D) \sim N(\bar{\mathbf{y}}, \Sigma_{\mathbf{y}})$.

4.3 Hyperparameter Learning

As we established a Bayesian framework for interpreting many of the existing graph based semi-supervised learning methods, we can now utilize evidence maximization to learn the hyperparameters. Evidence maximization has been one of the favorite tools for performing model selection. Evidence is a numerical quantity and signifies how well a model fits the given data. By comparing the evidence corresponding to the different models (or hyperparameters that determine the model), we can choose the model and the hyperparameters suitable for the task.

Denote the parameters of the kernel as Θ_K and the parameters of transformation of the graph Laplacian as Θ_T. Let $\Theta = \{\Theta_K, \Theta_T, \epsilon\}$, where ϵ is the noise hyperparameter. The goal is to solve $\hat{\Theta} = \arg\max_{\Theta} \log[p(\mathbf{t_L}|\mathbf{X}, \Theta)]$.

Non-linear optimization techniques, such as gradient descent or Expectation Maximization (EM) can be used to optimize the evidence. When the parameter space is small then the Matlab function `fminbnd`, based on golden section search and parabolic interpolation, can be used. The main challenge is that the gradient of evidence is not easy to compute.

Previously, an EM algorithm for hyperparameter learning [40] has been derived for Gaussian Process classification. Using similar ideas we can derive an EM algorithm for semi-supervised learning. In the E-step EP is used to infer the posterior $q(\mathbf{y})$ over the soft labels. The M-step consists of maximizing the variational lower bound:

$$\begin{aligned} F &= \int_{\mathbf{y}} q(\mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{X}, \Theta) p(\mathbf{t}_L|\mathbf{y}, \Theta)}{q(\mathbf{y})} \\ &= -\int_{\mathbf{y}} q(\mathbf{y}) \log q(\mathbf{y}) + \int_{\mathbf{y}} q(\mathbf{y}) \log N(\mathbf{y}; 0, r(\Delta)^{-1}) \\ &+ \sum_{i=1}^{n} \int_{y_i} q(y_i) \log \left(\epsilon + (1 - 2\epsilon)\Phi(y_i \cdot t_i)\right) \leq p(\mathbf{t_L}|\mathbf{X}, \Theta) \end{aligned}$$

The EM procedure alternates between the E-step and the M-step until convergence.

- **E-Step**: Given the current parameters Θ^i, approximate the posterior $q(\mathbf{y}) \sim N(\bar{\mathbf{y}}, \Sigma_{\mathbf{y}})$ by EP.

- **M-Step**: Update $\Theta^{i+1} = \arg\max_{\Theta} \int_{\mathbf{y}} q(\mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{X},\Theta)p(\mathbf{t}_L|\mathbf{y},\Theta)}{q(\mathbf{y})}$

In the M-step the maximization with respect to the Θ cannot be computed in a closed form, but can be solved using gradient descent. For maximizing the lower bound, we used the gradient based projected BFGS method using the Armijo rule and simple line search. When using the linear transformation $r(\lambda) = \lambda + \delta$ on the Laplacian Δ, the prior $p(\mathbf{y}|\mathbf{X}, \Theta)$ can be written as $N(0, (\Delta + \delta I)^{-1})$. Define $\mathbf{Z} = \Delta + \delta I$ then, the gradients of the lower bound with respect to the parameters are as follows:

$$\frac{\partial F}{\partial \Theta_K} = \frac{1}{2} tr(\mathbf{Z}^{-1} \frac{\partial \Delta}{\partial \Theta_K}) - \frac{1}{2} \bar{\mathbf{y}}^T \frac{\partial \Delta}{\partial \Theta_K} \bar{\mathbf{y}} - \frac{1}{2} tr(\frac{\partial \Delta}{\partial \Theta_K} \Sigma_{\mathbf{y}})$$

$$\frac{\partial F}{\partial \Theta_T} = \frac{1}{2} tr(\mathbf{Z}^{-1}) - \frac{1}{2} \bar{\mathbf{y}}^T \bar{\mathbf{y}} - \frac{1}{2} tr(\Sigma_{\mathbf{y}})$$

$$\frac{\partial F}{\partial \epsilon} \approx \sum_{i=1}^{n} \frac{1 - 2\Phi(t_i \cdot \bar{y}_i)}{\epsilon + (1 - 2\epsilon)\Phi(t_i \cdot \bar{y}_i)} \text{ where: } \bar{y}_i = \int_{\mathbf{y}} y_i q(\mathbf{y})$$

It is easy to show that the provided approximation of the derivative $\frac{\partial F}{\partial \epsilon}$ equals zero, when $\epsilon = \frac{k}{n}$, where k is the number of labeled data points differing in sign from their posterior means. The EM procedure described here is susceptible to local minima and in a few cases might be too slow to converge. Especially, when the evidence curve is flat and the initial values are far from the optimum, we found that the EM algorithm provided very small steps, thus, taking a long time to converge.

Whenever we encountered this problem in the experiments, we used an approximate gradient search to find a good value of initial parameters for the EM algorithm. Essentially as the gradients of the evidence are hard to compute, they can be approximated by the gradients of the variational lower bound and can be used in any gradient ascent procedure.

4.3.1 Classifying New Points

Most of the approaches for semi-supervised classification solve the problem of predicting the labels on the unlabeled points, rather than providing a classification function. Thus, classifying new points, which were not part of the semi-supervised learning, is non-trivial. Usually an approximation based on nearest-neighbor or a linear combination of data points is used. Since we compute a posterior distribution over the soft-labels of the labeled and unlabeled data points, classifying a new point is tricky.

Note, that from the parameterization lemma for Gaussian Processes [1] it follows that given a prior distribution $p(\mathbf{y}|\mathbf{X}) \sim N(0, r(\Delta)^{-1})$, the mean of the posterior $p(\mathbf{y}|D)$ is a linear combination of the columns of $r(\Delta)^{-1}$. That is:

$$\bar{\mathbf{y}} = r(\Delta)^{-1}\mathbf{a} \quad \text{where, } \mathbf{a} \in I\!R^{(n+m)\times 1}$$

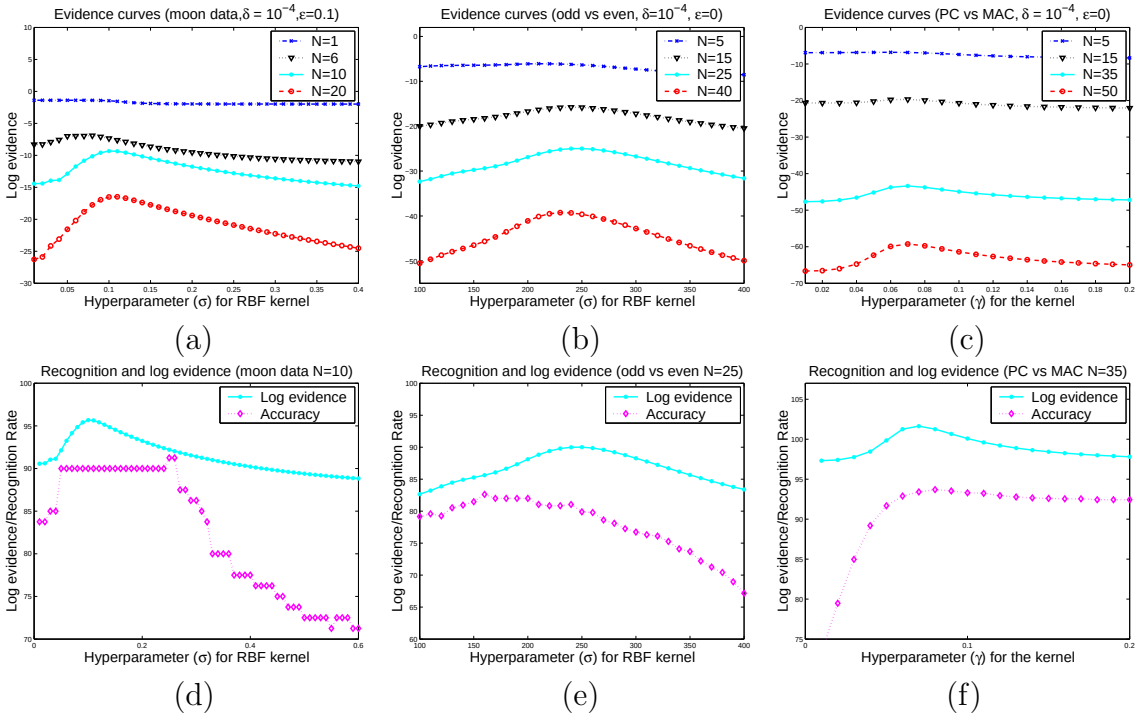


Figure 4-2: Evidence curves showing similar properties across different datasets (half-moon, odd vs even and PC vs MAC). The top row figures (a), (b) and (c) show the evidence curves for different amounts of labeled data per class. The bottom row figures (d), (e) and (f) show the recognition accuracy on unlabeled points and the log evidence, illustrating the correlation between the recognition performance and the evidence.

Further, if the similarity matrix K is a valid kernel matrix[2] then we can write the mean directly in terms of the linear combination of the columns of K:

$$\bar{\mathbf{y}} = KK^{-1}r(\Delta)^{-1}\mathbf{a} = K\mathbf{b} \tag{4.4}$$

Here, $\mathbf{b} = [b_1, .., b_{n+m}]^T$ is a column vector and is equal to $K^{-1}r(\Delta)^{-1}\mathbf{a}$. Thus, we have that $\bar{y}_i = \sum_{j=1}^{n+m} b_j \cdot K(\mathbf{x}_i, \mathbf{x}_j)$. This provides a natural extension of the framework to classify new points.

4.4 Experiments

We performed experiments to evaluate the three main contributions of this work: Bayesian hyperparameter learning, classification of unseen data points, and robustness with respect to noisy labels. For all the experiments we use the linear transformation $r(\lambda) = \lambda + \delta$ either on normalized Laplacian (EP-NL) or the combinatorial

[2]The matrix K is the adjacency matrix of the graph and depending upon the similarity criterion might not always be positive semi-definite. For example, discrete graphs induced using K-nearest neighbors might result in K that is not positive semi-definite.

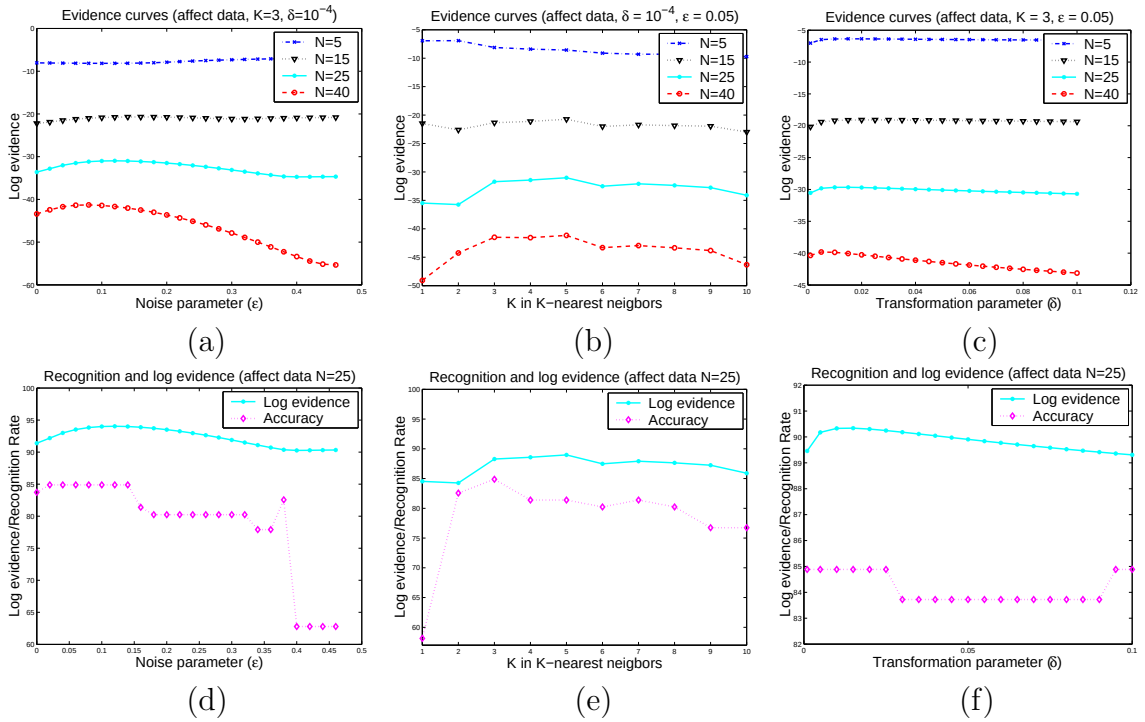


Figure 4-3: Evidence curves showing similar properties across different parameters of the model. The figures (a), (b) and (c) show the evidence curves for different amount of labeled data per class for the three different parameters in the model. The bottom row figures (d), (e) and (f) show the recognition accuracy on unlabeled points and the evidence, illustrating the correlation between the recognition performance and the evidence.

Laplacian (EP-CL). The experiments were performed on one synthetic (Figure 4-5(a)) and on three real-world datasets. Two real-world datasets were the handwritten digits and the newsgroup data from [92]. We evaluated the task of classifying odd vs even digits (15 labeled, 485 unlabeled and 1500 new (unseen) points per class) and classifying PC vs MAC (5 labeled and 895 unlabeled points per class and 61 points for PC and 82 points for MAC as new (unseen) points). An RBF kernel was used for handwritten digits, whereas the kernel $K(\mathbf{x_i}, \mathbf{x_j}) = exp[-\frac{1}{\gamma}(1 - \frac{\mathbf{x_i}^T\mathbf{x_j}}{|\mathbf{x_i}||\mathbf{x_j}|})]$ was used on a 10-NN graph to determine similarity. The third real-world dataset labels the level of interest (61 samples of high interest and 75 samples of low interest) of a child solving a puzzle on the computer. Each data point is a 19 dimensional real vector summarizing 8 seconds of activity from the face, posture and the puzzle. The labels in this database are suspected to be noisy because of human labeling. All the experiments on this data used K-nearest neighbor to determine the similarity matrix[3].

Hyperparameter learning: Figures 4-2 (a), (b) and (c) plot log evidence versus kernel parameters that determine the similarity graphs for the different datasets with

[3]The similarity matrix induced by K-nearest neighbor criterion might be not positive semi-definite. However, the graph Laplacians are always positive semi-definite; thus, we can use this construction without any difficulty

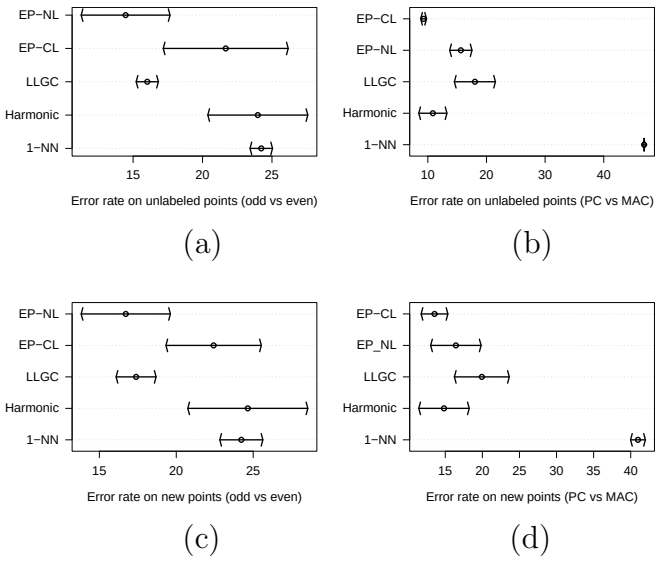


Figure 4-4: Error rates for different algorithms on digits (first column, (a) and (c)) and newsgroup dataset (second column (b) and (d)). The figures in the top row (a) and (b) show error rates on unlabeled points and the bottom row figures (c) and (d) on the new points. The results are averaged over 5 runs. Non-overlapping of error bars, the standard error scaled by 1.64, indicates 95% significance of the performance difference.

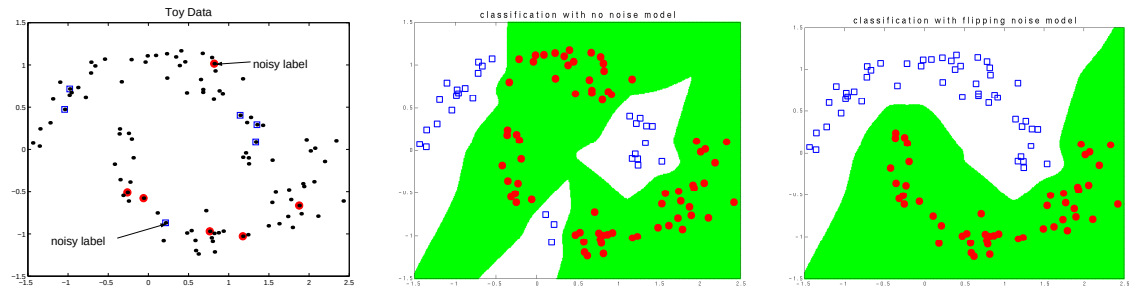


Figure 4-5: Semi-supervised classification in the presence of label noise. (a) Input data with label noise. Classification (b) without flipping noise model and with (c) flipping noise model.

varying size of the labeled set per class. The value of δ and ϵ were fixed to the values shown in the plots just for purpose of illustration. Figure 4-3 (a), (b) and (c) plots the log evidence versus the noise parameter (ϵ), the similarity matrix parameter (k in k-NN) and the transformation parameter (δ) for the affect dataset. First, we see that the evidence curves generated with very little data are flat and as the number of labeled data points increases we see the curves become peakier. When there is very little labeled data, there is not much information available for the evidence maximization framework to prefer one parameter value over the other. With more labeled data, the evidence curves become more informative. Figure 4-2 (d), (e) and (f) show the evidence curves and the recognition rate on the unlabeled data and reveal that the recognition over the unlabeled data points is highly correlated with the evidence. Note that both of these effects are observed across all the datasets as well as all the different parameters, suggesting that evidence maximization can be used for hyperparameter learning.

How good are the learnt parameters? We performed experiments on the handwritten digits and on the newsgroup data and compared with 1-NN, LLGC and the Harmonic approach. The kernel parameters for both LLGC and Harmonic were estimated using leave one out cross-validation[4]. Note that both the Harmonic field and the LLGC approach can be interpreted in terms of the new proposed Bayesian framework. (see section 4.2.1). We performed experiments with both the normalized (EP-NL) and the combinatorial Laplacian (EP-CL) with the proposed framework to classify the digits and the newsgroup data. The approximate gradient descent was first used to find an initial value of the kernel parameter for the EM algorithm. All three parameters were learnt and the top row in figure 4-4 shows the average error obtained for 5 different runs on the unlabeled points. On the task of classifying odd vs even the error rate for EP-NL was 14.46±4.4%, significantly outperforming the Harmonic (23.98±4.9%) and 1-NN (24.23±1.1%). Since the prior in EP-NL is determined using the normalized Laplacian and there is no label noise in the data, we expect the approach to at least work as well as the LLGC approach (16.02 ± 1.1%). Similarly for the newsgroup dataset EP-CL (9.28±0.7%) significantly beats LLGC (18.03±3.5%) and 1-NN (46.88±0.3%) and is better than Harmonic (10.86±2.4%). Similar, results are obtained on new points as well. The unseen points for EP-NL and EP-CL were classified using equation (4.4) and the nearest neighbor criterion was used for LLGC and Harmonic.

Handling label noise: Figure 4-5(a) shows a synthetic dataset with noisy labels. We performed semi-supervised classification both with and without the likelihood model given in (4.3) and the EM algorithm was used to tune all the parameters including the noise (ϵ). Besides modifying the spectrum of the Laplacian, the transformation parameter δ can also be considered as latent noise and provides a quadratic slack for the noisy labels [40]. The results are shown in figure 4-5 (b) and (c). The EM algorithm can correctly learn the noise parameter resulting in a perfect classification. The classification without the flipping model, even with the quadratic slack, cannot

[4]Search space for σ (odd vs even) was 100 to 400 with increments of 10 and for γ (PC vs MAC) was 0.01 to 0.2 with increments of 0.1

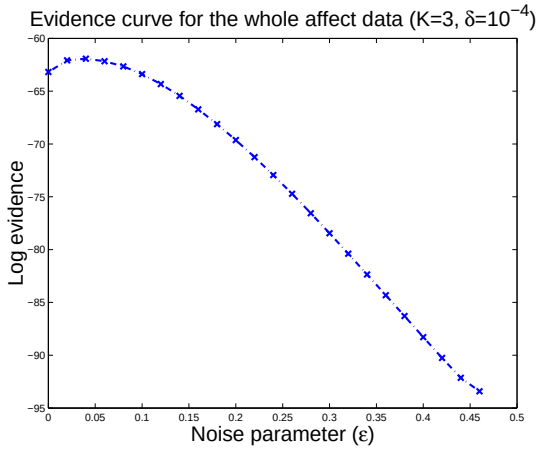


Figure 4-6: Evidence vs noise parameter plotted using all the available data in the affect dataset. The maximum at $\epsilon = 0.05$ suggests that there is around 5% label noise in the data.

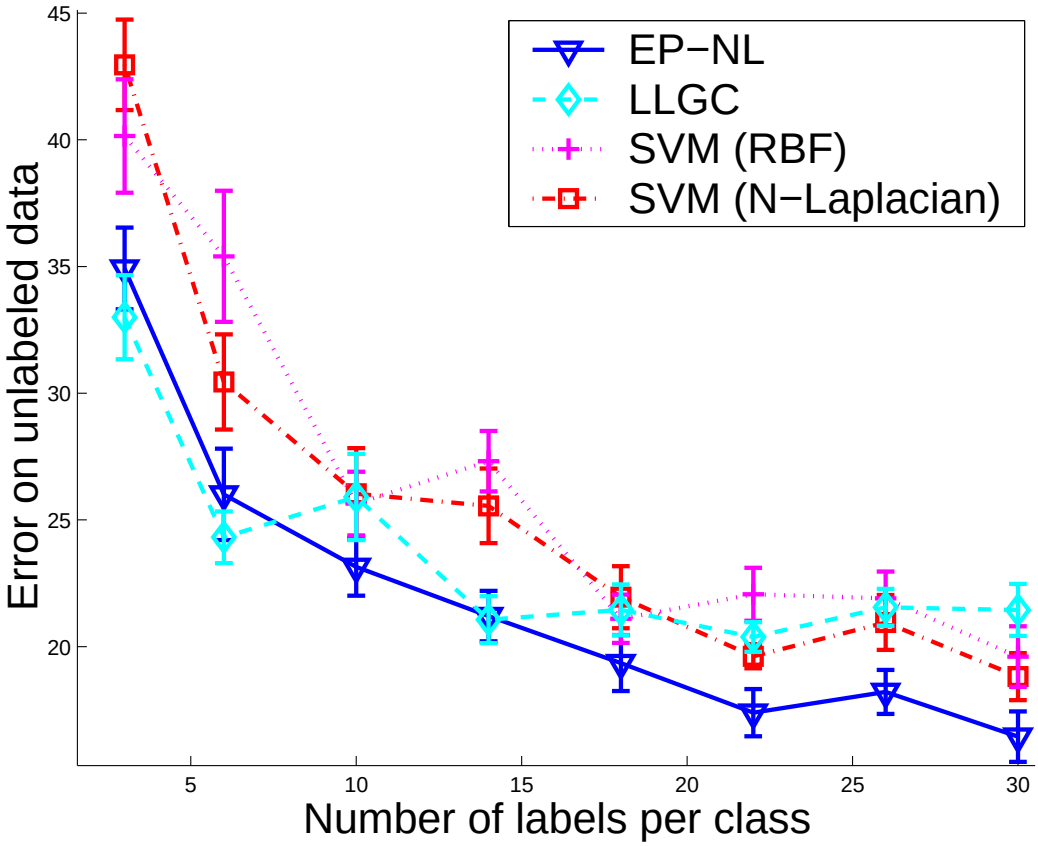


Figure 4-7: (a) Performance comparison of the proposed approach (EP-NL with the flipping noise model) with LLGC, SVM using the normalized Laplacian graph kernel and the supervised SVM (RBF kernel) on the affect dataset which has label noise. The similarity matrix K in all of the semi-supervised methods is constructed using the symmetric 3-nearest neighbor criteria. The error bars represent the standard error.

handle the noisy labels far from the decision boundary.

Is there label noise in the data? It was suspected that due to the manual labeling the affect dataset might have some label noise. To confirm this and as a sanity check, we first plotted evidence using *all* the available data. For all of the semi-supervised methods in these experiments, we use 3-NN to induce the adjacency graph. Figure 4-6 shows the plot for the evidence against the noise parameter (ϵ). From the figure, we see that the evidence peaks at $\epsilon = 0.05$ suggesting that the dataset has around 5% of labeling noise.

Figure 4-7 shows comparisons with other semi-supervised (LLGC and SVM with graph kernel) and supervised methods (SVM with RBF kernel) for different sizes of the labeled dataset. Each point in the graph is the average error on 20 random splits of the data, where the error bars represent the standard error. EM was used to tune ϵ and δ in every run. We used the same transformation $r(\lambda) = \lambda + \delta$ on the graph kernel

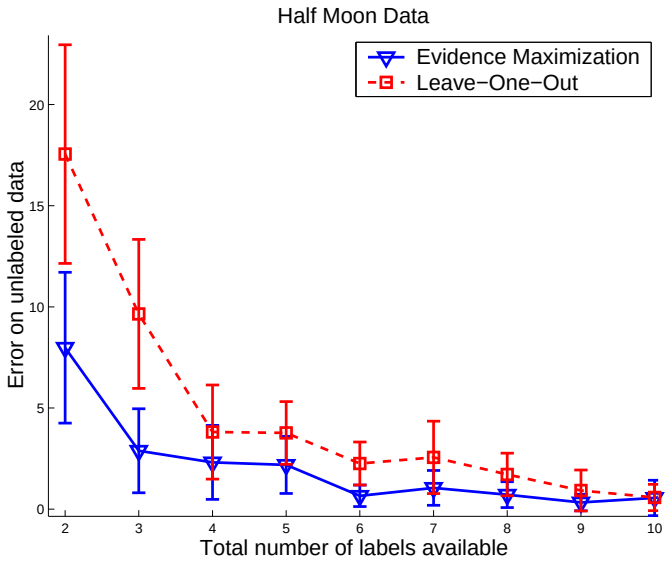


Figure 4-8: Performance comparison when choosing kernel hyperparameters (σ for RBF) using leave-one-out strategy and evidence maximization on the two half-moon dataset. Non-overlapping error bars (standard error scaled by 1.64) depict 95% confidence in performance difference. The simulation suggests that evidence might be more useful to select hyperparameters when the number of available labels are small.

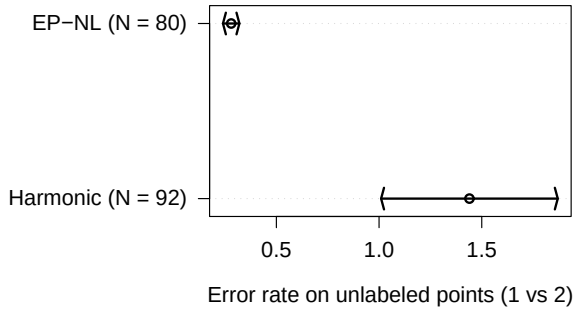


Figure 4-9: Comparison of the proposed EM method for hyperparameter learning with the result reported in [92] using label entropy minimization. The plotted error bars represent the standard deviation.

in the semi-supervised SVM. The hyperparameters in both the SVMs (including δ for the semi-supervised case) were estimated using leave one out.

When the number of labeled points is small, both LLGC and EP-NL perform similarly beating both the SVMs, but as the size of the labeled data increases we see a significant improvement of the proposed approach over the other methods. One of the reasons is when you have few labels the probability of the labeled set of points containing a noisy label is low. As the size of the labeled set increases the labeled data has more noisy labels. And, since LLGC has a Gaussian noise model, it cannot handle flipping noise well. As the number of labels increase, the evidence curve turns informative and EP-NL starts to learn the label noise correctly, outperforming the other Both the SVMs show competitive performance with more labels but still are worse than EP-NL.

We also test the method on the task of classifying "1" vs "2" in the handwritten digits dataset. With 40 labeled examples per class (80 total labels and 1800 unlabeled), EP-NL obtained an average recognition accuracy of $99.72 \pm 0.04\%$ and figure 4-9 graphically shows the gain over the accuracy of $98.56 \pm 0.43\%$ reported in [92], where the hyperparameters were learnt by minimizing label entropy with 92 labeled and 2108 unlabeled examples.

Comparing leave-one-out vs evidence maximization: Finally, we compare how the performance differs when we use the leave-one-out criterion as opposed to evidence maximization. Figure 4-8 plots recognition error for EP-NL on the half-moon dataset as the number of labeled data points is varied. The parameter ϵ is fixed to zero in the flipping noise model and we are only searching for σ, the kernel width in the RBF kernel. The results are averaged over 50 runs and again the non-overlapping error bars depict 95% confidence in the performance difference. We can see that when the number of available labels is small, evidence maximization performs much better than the leave-one-out strategy. However as the number of labels increase, both the measures converge to the same performance. Although, these experiments are performed on a toy dataset, assuming no noise in the labels, the result hints that the evidence maximization has an advantage over the leave-one-out strategy in semi-supervised scenarios where the amount of labeled data is very small.

4.5 Connections to Gaussian Processes

One key difference between the proposed semi-supervised learning framework and supervised classification using Gaussian processes is the process prior. Usually in the case of GP classification the process prior over the soft labels $\mathbf{y}$ is assumed to be $\mathcal{N}(0, K)$, where $K = [k_{ij}]$ is the kernel matrix with $k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ encoding similarity between the two data points. Equivalently, using Mercer's theorem [75] we can view the process prior in the weight space perspective by assuming $y_i = \sum_k w_k \nu_k^{1/2} \phi_k(\mathbf{x}_i)$, where $[w_1, w_2, ...]^T = \mathbf{w} \sim \mathcal{N}(0, \mathbf{I})$ and $\phi_k(\cdot)$ are the eigenfunctions of $k(\cdot, \cdot)$ and ν_k the corresponding eigenvalues.

On the other hand, in the semi-supervised learning scenario we have the following process prior: $\mathbf{y} \sim \mathcal{N}(0, r(\Delta)^{-1})$. Despite the fact that here we cannot parameterize the ij^{th} entry of the covariance matrix in terms of the i^{th} and the j^{th} data point, this is still a Gaussian process prior. To realize this, note that the eigenfunctions corresponding to the covariance matrix $r(\Delta)^{-1}$ are same as the eigenfunctions[5] corresponding to the original kernel matrix K. Thus, we can again write $y_i = \sum_k w_k \nu_k^{1/2} \phi_k(\mathbf{x}_i)$. Assuming the prior on $\mathbf{w}$ as $\mathcal{N}(0, \text{diag}[\frac{r(\lambda_1)^{1/2}}{\nu_1^{1/2}}, \frac{r(\lambda_2)^{1/2}}{\nu_2^{1/2}}, ..])$, where $r(\lambda_k)$ are the eigenvalues for $r(\Delta)^{-1}(\cdot, \cdot)$, we obtain the required process prior $\mathbf{y} \sim \mathcal{N}(0, r(\Delta)^{-1})$. Thus, the only difference between the supervised and the semi-supervised case is the prior on the weights. In the case of supervised classification using Gaussian processes we assume a spherical prior on $\mathbf{w}$, whereas the semi-supervised case assumes an elliptical prior which is determined by the given labeled and unlabeled data points. Thus, the Bayesian framework presented here nicely extends the supervised classification using Gaussian processes.

Figure 4-10 graphically shows the difference between the traditional Gaussian process classification (supervised) and the semi-supervised case. In this example we limit

[5]This is strictly true for the combinatorial Laplacian ($\Delta = G - K$) only. For the normalized Laplacian ($\tilde{\Delta} = I - G^{-\frac{1}{2}} K G^{-\frac{1}{2}}$) this holds when the underlying graph is strictly regular, that is, all the vertices have the same degree.

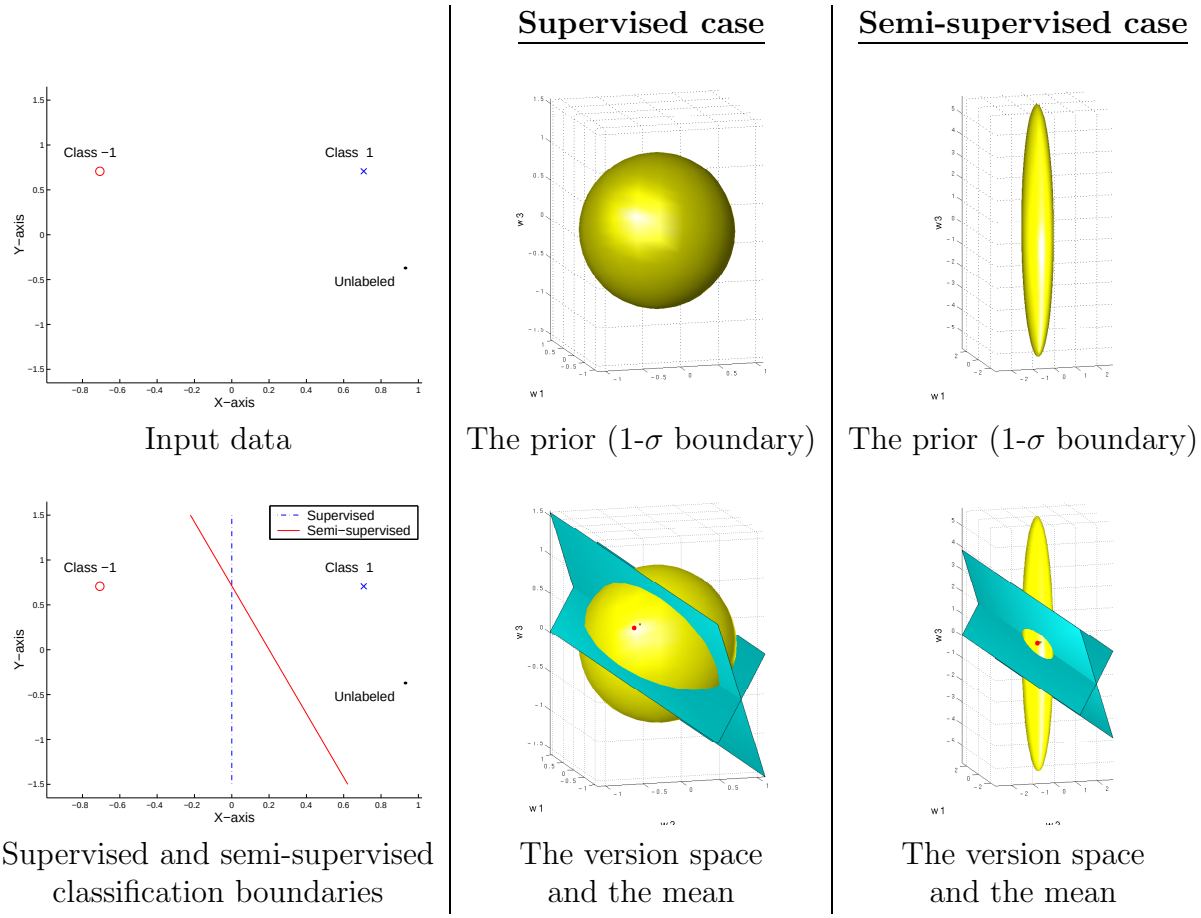


Figure 4-10: An illustrative example to relate the supervised and semi-supervised classification. The traditional Gaussian process framework does supervised classification and assumes a spherical prior over the weights. The semi-supervised case on the other hand starts with an elliptical prior due to the data dependent regularization. In both of these case the labeled data points imposes constraints (the blue planes in the second row) over the solutions. EP can be used to find the mean solution in the version space in both the scenarios (denoted by the red dot).

Table 4.1: Graph based Semi-supervised classification.

	Regularizer	Noise Model	Graph Transform
Regularized Laplacian [4]	Combinatorial Laplacian	Gaussian	$r(\lambda) = \lambda$
LLGC [91]	Normalized Laplacian	Gaussian	$r(\lambda) = \lambda$
Harmonic Fields [92]	Combinatorial Laplacian	Gaussian with $\sigma \rightarrow 0$	$r(\lambda) = \lambda$
Laplace [94]	Combinatorial Laplacian	logistic	$r(\lambda) = \lambda + \delta$
Diffusion Kernels [43]	Combinatorial Laplacian	-NA-	$r(\lambda) = \exp \frac{\sigma^2}{2\lambda}$
Manifold Regularization [5] Regularized Least Square	Combinatorial Laplacian and a regular Kernel	Gaussian	$r(\lambda) = \gamma_I \lambda$ $\gamma_I \in I\!R$
Manifold Regularization [5] Regularized Hinge Loss	Combinatorial Laplacian and a regular Kernel	SVM Loss	$r(\lambda) = \gamma_I \lambda$ $\gamma_I \in I\!R$
Cluster Kernels [11]	Normalized Laplacian	SVM Loss	Various Choices

ourselves to the linear decision boundary (same as working with the linear kernel). The first column shows the input data and the mean classification boundaries resulting in supervised and semi-supervised cases. Note that the semi-supervised boundary is significantly different than the supervised case due to the effect of unlabeled data points on the prior.

The first row illustrates the priors in the supervised and the semi-supervised case. In the supervised case, the Gaussian process assumes a spherical Gaussian prior over the weights ($\mathbf{w} = \mathcal{N}(0, \mathbf{I})$). This prior is completely independent of the underlying data distribution. On the other hand, the third column shows the prior induced over the weights using a combinatorial Laplacian (we assume $r(\lambda) = \lambda$). The prior for semi-supervised classification can be written as $\mathbf{w} = \mathcal{N}(0, [\mathbf{\Phi} r(\Delta) \mathbf{\Phi}^T]^{-1})$ and depends upon the input data distribution. Here, $\mathbf{\Phi} = [\Phi_1, ..\Phi_{n+m}]$ denotes a matrix where each column is the feature representation of the input data points: $\Phi_i = [\phi_1(\mathbf{x}_i), \phi_2(\mathbf{x}_i), ..]$. The prior is an ellipse, which assigns higher probability to those classifiers which would preserve smoothness of labels over the underlying manifold of the data.

The second row shows the resulting version space due to the constraints imposed by the labeled data. The blue planes are due to the labeled data points and they split the weight space to define a convex region corresponding to the version space. The version space is the region that contains the set of classifiers that classify all the labeled data correctly. EP returns the mean of this version space (denoted by the red dot in the figures).

Thus, we can see that the framework used in this chapter is a special case of Gaussian process classification. The only difference is that the prior over the weights $\mathbf{w}$ is not spherical as the data dependent regularization prefers those weights which would provide smooth labelings over the underlying manifold of the data.

4.6 Connections to Other Approaches

Many of the graph based semi-supervised learning algorithms can be interpreted in terms of regularization on graphs [79]. Most of the approaches aim to derive a Kernel matrix by using the graph Laplacian as a regularizer. Note that in many cases a

monotonic function might be applied to modify the spectrum of the graph Laplacian before using it as a regularizer. For different choices of Laplacian (Combinatorial or Normalized) and different choices of the transformation function, we can derive the regularizers for most of the graph based semi-supervised learning approaches. Moreover, there is a duality between the Kernels in Reproducing Kernel Hilbert Space and the Gaussian process priors which allows us to translate the regularizers in terms of prior probabilities. Thus, we can induce process priors, which are Gaussian, for semi-supervised learning motivated by the regularization on graphs.

Many of the graph based semi-supervised learning techniques were derived independently; thus, there are other differences besides just the process prior. One such difference is the noise model or the likelihood. Even though, some of the work has just focused on deriving the graph kernels [11, 43, 79], where these kernels can be used in an SVM classification formulation, there have been approaches where different noise models have been used. For example, Gaussian noise has been frequently used [91] and it has the advantage that the solutions can be often obtained as closed form expressions. Similarly, Zhu et al [94] have used a noise model based on logistic function, which is perhaps a more natural noise model than the Gaussian one. However, there is no closed form solution when using this noise model and an approximation (for example Laplace or EP) is warranted. The null category noise model [49] is another noise model that has been introduced for semi-supervised classification. We would like to point out that much of the semi-supervised learning work has focused on the smoothness constraints over the graph without paying much attention to the noise model. That is, most of the work so far has been focused on exploiting the information from the unlabeled points without really caring about different kinds of noise the training data might have. We feel that modeling noise is a very important aspect and a complete semi-supervised learning framework should focus both on the process prior and the noise model as well. The Bayesian framework described here aims to cover both of these aspects and provides us insight to many of these semi-supervised learning methods.

Table 4.1 summarizes many of the recent graph based semi-supervised learning methods and their interpretation in terms of the Bayesian framework. By choosing different kinds of Laplacians, different noise models and a different transformation of the graph spectrum, we can derive all of these methods. Most of the graph based methods aim to provide a solution that respect the labels in the training data but also consider an appropriate regularization term that depends upon the underlying graph. The Bayesian interpretation stems from the fact that the regularizers can be used to induce a prior over the possible labelings and together with the likelihood terms that favor the solution consistent with the training data, we can tie most of these methods in a single framework. Note that there is no easy Bayesian interpretation of SVM loss. There have been attempts in the past and perhaps the null category noise model introduced by Lawrence and Jordan [49] is the closest. Nonetheless, by viewing all of these methods in the Bayesian perspective, we can exploit techniques such as evidence maximization to choose hyperparameters for these methods. Further, given a new dataset we now have a tool for choosing the best performing algorithms between these graph based methods.

Also, note that even though the transductive SVMs [86] and the semi-supervised classification using the null category noise model [49] are not graph based methods, they can still be subsumed under this Bayesian perspective. Lawrence and Jordan have used null category noise model together with the Gaussian process prior; hence, besides the Gaussian process prior the only difference between their method and graph based techniques is the noise model. Further, the proposed null category noise model imitates the SVM hinge loss for classification. Thus, taking into account the duality between regularization in RKHS and the Gaussian process prior, we can relate the transductive SVM and the Gaussian process formulation.

4.7 Conclusion and Future Work

We have presented and evaluated a new method for learning hyperparameters for graph-based semi-supervised classification in a Bayesian framework. The results indicate that evidence maximization works well for learning hyperparameters, including the amount of label noise in the data.

Computational complexity: The computational complexity for the evidence maximization procedure is $O(n^3)$, where n is the number of data points, due to matrix inversion required to compute the gradient.

Convergence issues: Since the proposed algorithm is based on gradient descent and EM, it suffers from some of the disadvantages of these algorithms. The EM algorithm is guaranteed to converge at a local optimum; consequently, the algorithm may not always provide the best solution. One possible way to alleviate this problem is to provide good initializations to the parameters using any available prior knowledge. Further, we can do a very coarse search over the parameter space to find good initial estimates.

Possible improvements: A very interesting direction would be to incorporate kernel alignment measures to reduce the search space as well as to obtain good initial estimates for the parameters and the hyper parameters. Further, sparsification on the lines of [17] would help a lot to reduce the computational complexity of the evidence maximization procedure.

Chapter 5

Located Hidden Random Fields to Learn Discriminative Parts

This chapter[1] introduces the Located Hidden Random Field (LHRF), a conditional model for simultaneous part-based detection and segmentation of events or objects of a given class. Given some training data with coarse labels, the LHRF automatically learns a set of parts (the finer labels) that are both discriminative and informative about the relative position (either spatial or temporal) with respect to the coarse labels provided for training.

Parts based object detection and segmentation from images is one task where these models can be used very effectively. The idea in the discriminative framework is to train a network of classifiers that can first detect parts of objects and then combine there inferences to segment out the complete object from the image. However, it is very difficult and tedious to obtain training images labeled for each individual part. Thus, we need a model that can reliably learn parts from segmentation masks only (which are considered coarse labels). LHRF solves this problem by introducing a set of latent variables, which correspond to parts and are hidden during the training phase. By introducing the global position of the object as a latent variable, the LHRF models the long-range spatial configuration of these parts, as well as their local interactions. Experiments on benchmark datasets show that the use of discriminative parts leads to state-of-the-art detection and segmentation performance, with the additional benefit of obtaining a labeling of the object's component parts.

Although most of the discussion in this chapter focuses around the problem of object detection and segmentation from the images, we would like to point out that the proposed model can be very easily applied to activity recognition and affect recognition scenarios.

[1] All of the work described from section 5.2 to section 5.6 was done in collaboration with John Winn during the summer internship in 2005 at Microsoft Research, Cambridge, UK. This work appears in European Conference on Computer Vision, 2006 [38].

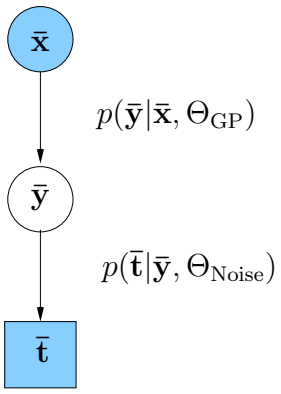


Figure 5-1: The graphical model for Gaussian process sequence classification.

5.1 Relation to Gaussian Process Classification

The model proposed in this chapter is an extension of the Conditional Random Field (CRF) [47] and can be considered as a network of classifiers. Altun et al. [2] have shown that CRFs can be thought of as a special case of the Gaussian process framework. Note, that this is different from the regular GP classification where there is one single label for every data point. However, we can extend the GP classification framework to a CRF by assuming existence of random variables $\bar{\mathbf{y}}$, which are again induced by Gaussian processes. Note, that all the labels $\bar{\mathbf{t}}$ depend upon each other and this dependence is characterized by the structure of the underlying graph. The CRF framework can be considered as a special case of Gaussian process classification [2] where we only use a linear kernel. We can generalize the CRF framework to other types of kernels as well. Figure 5-1 graphically shows the a CRF based on a network of Gaussian process classifiers.

In this chapter we mostly focus on linear classifiers (i.e. having a linear kernel). Recently, a method for Bayesian training and computing the Bayes point for these models was proposed [67]. However, for simplicity we only focus here on the MAP solution.

5.2 Learning Discriminative Parts for Object Detection and Segmentation

This chapter specifically addresses the problem of simultaneous detection and segmentation of objects belonging to a particular class. Our approach is to use a conditional model which is capable of learning discriminative parts of an object. A part is considered discriminative if it can be reliably detected by its local appearance in the image and if it is well localized on the object and hence informative as to the object's location.

The use of parts has several advantages. First, there are local spatial interactions between parts that can help with detection, for example, we expect to find the nose right above the mouth on a face. Hence, we can exploit local part interactions

to exclude invalid hypotheses at a local level. Second, knowing the location of one part highly constrains the locations of other parts. For example, knowing the locations of wheels of a car constrains the positions where the rest of the car can be detected. Thus, we can improve object detection by incorporating long range spatial constraints on the parts. Third, by inferring a part labeling for the training data, we can accurately assess the variability in the appearance of each part, giving better part detection and hence better object detection. Finally, the use of parts gives the potential for detecting objects even if they are partially occluded.

One possibility for training a parts-based system is to use supervised training with hand-labeled parts. The disadvantage of this approach is that it is very expensive to get training data annotated for parts, plus it is unclear which parts should be selected. Existing generative approaches try to address these problems by clustering visually similar image patches to build a codebook in the hope that clusters correspond to different parts of the object. However, this codebook has to allow for all sources of variability in appearance. We provide a discriminative alternative where irrelevant sources of variability do not need to be modeled.

This chapter introduces Located Hidden Random Field, a novel extension to the Conditional Random Field [47] that can learn parts discriminatively. We introduce a latent part label for each pixel which is learned simultaneously with model parameters, given the segmentation mask for the object. Further, the object's position is explicitly represented in the model, allowing long-range spatial interactions between different object parts to be learned.

5.3 Related Work

There have been a number of parts-based approaches to segmentation or detection. It is possible to pre-select which parts are used as in Crandall et al. [16] – however, this requires significant human effort for each new object class. Alternatively, parts can be learned by clustering visually similar image patches [1, 50] but this approach does not exploit the spatial layout of the parts in the training images. There has been work with generative models that do learn spatially coherent parts in an unsupervised manner. For example, the constellation models of Fergus et al. [21] learn parts which occur in a particular spatial arrangement. However, the parts correspond to sparsely detected interest points and so parts are limited in size, cannot represent untextured regions and do not provide a segmentation of the image. More recently, Winn and Jojic [90] used a dense generative model to learn a partitioning of the object into parts, along with an unsupervised segmentation of the object. Their method does not learn a model of object appearance (only of object shape) and so cannot be used for object detection in cluttered images.

As well as unsupervised methods, there are a range of supervised methods for segmentation and detection. Ullman and Borenstein [8] use a fragment-based method for segmentation, but do not provide detection results. Shotton et al. [77] use a boosting method based on image contours for detection, but this does not lead to a segmentation. There are a number of methods using Conditional Random Fields (CRFs) to

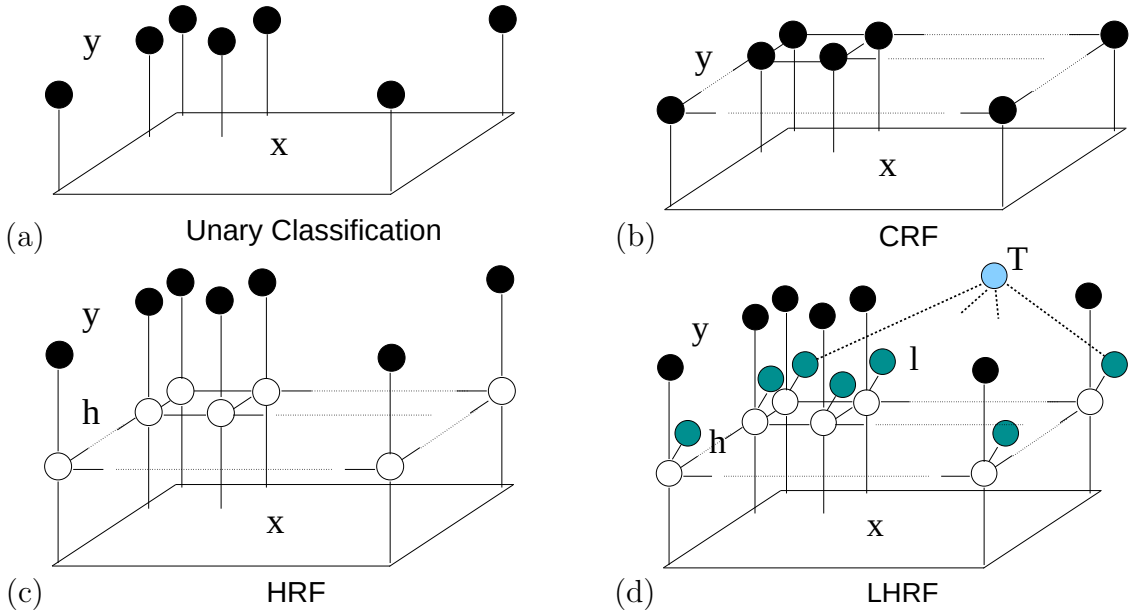


Figure 5-2: Graphical models for different discriminative models of images. The image $\mathbf{x}$ and the shaded vertices are observed during training time. The parts $\mathbf{h}$, denoted by unfilled circles, are not observed and are learnt during the training. In the LHRF model, the node corresponding to T is connected to all the locations l_i, depicted using thick dotted lines.

achieve segmentation [45] or sparse part-based detection [68]. The OBJ CUT work of Kumar et al. [44] uses a discriminative model for detection and a separate generative model for segmentation but requires that the parts are learned in advance from video. Unlike the work presented here, none of these approaches achieves part-learning, segmentation and detection in a single probabilistic framework.

Our choice of model has been motivated by Szummer's [81] Hidden Random Field (HRF) for classifying handwritten ink. The HRF automatically learns parts of diagram elements (boxes, arrows etc.) and models the local interaction between them. However, the parts learned using an HRF are not spatially localized as the relative location of the part on the object is not modeled. In this work we introduce the Located HRF, which models the spatial organization of parts and hence learns part which are spatially localized.

5.4 Discriminative Models for Object Detection

Our aim is to take an $n \times m$ image $\mathbf{x}$ and infer a label for each pixel indicating the class of object that pixel belongs to. We denote the set of all image pixels as V and for each pixel $i \in V$ define a label[2] $y_i \in \{0, 1\}$ where the background class is indicated by $y_i = 0$ and the foreground by $y_i = 1$. The simplest approach is to classify each pixel

[2]Please note the change in notation. $\mathbf{h}$ will be the hidden label of interest (parts) and $\mathbf{y}$ are the auxiliary labels (foreground vs. background).

independently of other pixels based upon some local features, corresponding to the graphical model of Fig. 5-2a. However, as we would like to model the dependencies between pixels, a conditional random field can be used.

Conditional Random Field (CRF):

A CRF consists of a network of classifiers that interact with one another such that the decision of each classifier is influenced by the decision of its neighbors. In the graphical model for a CRF, the class label corresponding to every pixel is connected to its neighbors in a 4-connected grid, as shown in Fig. 5-2b. We denote this new set of edges as E.

Given an image $\mathbf{x}$, a CRF induces a conditional probability distribution $p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})$ using the potential functions ψ_i^1 and ψ_{ij}^2. Here, ψ_i^1 encodes compatibility of the label given to the ith pixel with the observed image $\mathbf{x}$ and ψ_{ij}^2 encodes the pairwise label compatibilities for all $(i, j) \in E$ conditioned on $\mathbf{x}$. Thus, the conditional distribution $p(\mathbf{y} \mid \mathbf{x})$ induced by a CRF can be written as:

$$p(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta}, \mathbf{x})} \prod_{i \in V} \psi_i^1(y_i, \mathbf{x}; \boldsymbol{\theta}) \prod_{(i,j) \in E} \psi_{ij}^2(y_i, y_j, \mathbf{x}; \boldsymbol{\theta}) \tag{5.1}$$

where the partition function $Z(\boldsymbol{\theta}, \mathbf{x})$ depends upon the observed image $\mathbf{x}$ as well as the parameters $\boldsymbol{\theta}$ of the model. We assume that the potentials ψ_i^1 and ψ_{ij}^2 take the following form:

$$\begin{aligned} \psi_i^1(y_i, \mathbf{x}; \boldsymbol{\theta}_1) &= \exp[\boldsymbol{\theta}_1(y_i)^{\mathrm{T}} \mathbf{g}_i(\mathbf{x})] \\ \psi_{ij}^2(y_i, y_j, \mathbf{x}; \boldsymbol{\theta}_2) &= \exp[\boldsymbol{\theta}_2(y_i, y_j)^{\mathrm{T}} \mathbf{f}_{ij}(\mathbf{x})] \end{aligned}$$

Here, $\mathbf{g}_i : \mathcal{R}^{n \times m} \rightarrow \mathcal{R}^d$ is a function that computes a d-dimensional feature vector at pixel i, given the image $\mathbf{x}$. Similarly, the function $\mathbf{f}_{ij} : \mathcal{R}^{n \times m} \rightarrow \mathcal{R}^d$ computes the d-dimensional feature vector for edge ij.

Hidden Random Field:

A Hidden Random Field (HRF) [81] is an extension to a CRF which introduces a number of *parts* for each object class. Each pixel has an additional hidden variable $h_i \in \{1 \ldots H\}$ where H is the total number of parts across all classes. These hidden variables represent the assignment of pixels to parts and are not observed during training. Rather than modeling the interaction between foreground and background labels, an HRF instead models the local interaction between the parts. Fig. 5-2c shows the graphical model corresponding to an HRF showing that the local dependencies captured are now between parts rather than between class labels. There is also an additional edge from a part label h_i to the corresponding class label y_i. Similar to [81], we assume that every part is uniquely allocated to an object class and so parts are not shared. Specifically, there is deterministic mapping from parts to object-class

and we can denote it using $y(h_i)$.

Similarly to the CRF, we can define a conditional model for the label image $\mathbf{y}$ and part image $\mathbf{h}$:

$$p(\mathbf{y}, \mathbf{h} \,|\, \mathbf{x}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta}, \mathbf{x})} \prod_{i \in V} \psi_i^1(h_i, \mathbf{x}; \boldsymbol{\theta}_1)\ \phi(y_i, h_i) \prod_{(i,j) \in E} \psi_{ij}^2(h_i, h_j, \mathbf{x}; \boldsymbol{\theta}_2) \tag{5.2}$$

where the potentials are defined as:

$$\begin{aligned} \psi_i^1(h_i, \mathbf{x}; \boldsymbol{\theta}_1) &= \exp[\boldsymbol{\theta}_1(h_i)^{\mathrm{T}} \mathbf{g}_i(\mathbf{x})] \\ \psi_{ij}^2(h_i, h_j, \mathbf{x}; \boldsymbol{\theta}_2) &= \exp[\boldsymbol{\theta}_2(h_i, h_j)^{\mathrm{T}} \mathbf{f}_{ij}(\mathbf{x})] \\ \phi(y_i, h_i) &= \delta(y(h_i) = y_i) \end{aligned}$$

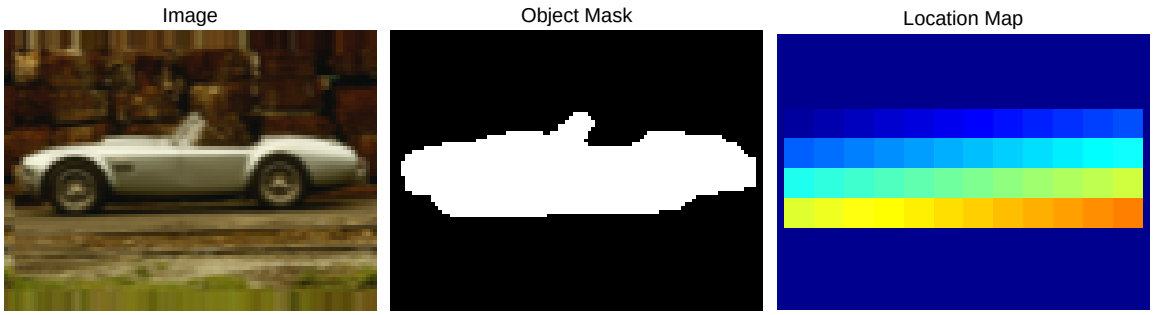
where δ is an indicator function. The hidden variables in the HRF can be used to model parts and interaction between those parts, providing a more flexible model which in turn can improve detection performance. However, there is no guarantee that the learnt parts are spatially localized. Also, as the model only contains local connections, it does not exploit the long-range dependencies between all the parts of the object.

5.4.1 Located Hidden Random Field

The Located Hidden Random Field (LHRF) is an extension to the HRF, where the parts are used to infer not only the background/foreground labels but also a position label in a coordinate system defined relative to the object. We augment the model to include the position of the object T, encoded as a discrete latent variable indexing all possible object locations in the image. We assume a fixed object size so a particular object position defines a rectangular reference frame enclosing the object. This reference frame is coarsely discretized into bins, representing different discrete locations within the reference frame. Fig. 5-3 shows an example image, the object mask and the reference frame divided into bins (shown color-coded).

We also introduce a set of location variables $l_i \in \{0, .., L\}$, where l_i takes the non-zero index of the corresponding bin, or 0 if the pixel lies outside the reference frame. For example, in Fig. 5-3 the different colors correspond to different values a location variable can take. Given a measure of compatibility between these location variables and different parts, we can impose long range dependency between the parts through the reference frame. Given a location T the location labels are uniquely defined according to the corresponding reference frame. Hence, when T is unobserved, the location variables are all tied together via their connections to T. These connections allow the long-range spatial dependencies between parts to be learned. As there is only a single location variable T, this model makes the assumption that there is a single object in the image (although it can be used recursively for detecting multiple objects – see Section 5.5).

Assume, that the potentials ψ^1, ψ^2, ϕ are defined as in the HRF, and ψ^3 denotes



(a) (b) (c)

Figure 5-3: Instantiation of different nodes in an LHRF. (a) image $\mathbf{x}$, (b) class labels $\mathbf{y}$ showing ground truth segmentation (c) color-coded location map $\mathbf{l}$. The darkest color corresponds to the background.

the potential encoding the compatibility between parts and locations:

$$\psi^3(h_i, l_i; \boldsymbol{\theta}_3) = \exp[\boldsymbol{\theta}_3(h_i, l_i)] \tag{5.3}$$

Here $\boldsymbol{\theta}_3(h_i, l_i)$ is a look-up table with an entry for each part and location index. Note, that each entry in the table is a parameter that needs to be learnt. We define a conditional model for the label image $\mathbf{y}$, the position T, the part image $\mathbf{h}$ and the locations $\mathbf{l}$ as:

$$\begin{aligned} p(\mathbf{y}, \mathbf{h}, \mathbf{l}, T \,|\, \mathbf{x}; \boldsymbol{\theta}) \;\; = \;\; & \prod_{i \in V} \psi_i^1(h_i, \mathbf{x}; \boldsymbol{\theta}_1)\; \phi(y_i, h_i)\; \psi^3(h_i, l_i; \boldsymbol{\theta}_3)\; \delta(l_i = \mathrm{loc}(i, T)) \\ & \times \prod_{(i,j) \in E} \psi_{ij}^2(h_i, h_j, \mathbf{x}; \boldsymbol{\theta}_2) \times \frac{1}{Z(\boldsymbol{\theta}, \mathbf{x})} \end{aligned} \tag{5.4}$$

Where, $\mathrm{loc}(i, T)$ is the location label of the ith pixel when the reference frame is in position T.

In the LHRF, the parts need to be compatible with the location index as well as the class label, which means that the part needs to be informative about the spatial location of the object as well as its class. Hence, unlike the HRF, the LHRF learns spatially coherent parts which occur in a consistent location on the object. The spatial layout of these parts is captured in the parameter vector $\boldsymbol{\theta}_3$, which encodes where each part lies in the co-ordinate system of the object.

Table 5.1 gives a summary of the properties of the four discriminative models which have been described in this section.

5.5 Inference and Learning

There are two key tasks that need to be solved when using the LHRF model: learning the model parameters $\boldsymbol{\theta}$ and inferring the labels for an input image $\mathbf{x}$.

Inference:

Given a novel image $\mathbf{x}$ and parameters $\boldsymbol{\theta}$, we can classify an i^{th} pixel as background or foreground by first computing the marginal $p(y_i \,|\, \mathbf{x}; \boldsymbol{\theta})$ and assigning the label that maximizes this marginal. The required marginal is computed by marginalizing out the part variables $\mathbf{h}$, the location variables $\mathbf{l}$, the position variable T and all the labels $\mathbf{y}$ except y_i.

$$p(y_i \,|\, \mathbf{x}; \boldsymbol{\theta}) = \sum_{\mathbf{y}/y_i} \sum_{\mathbf{h},\mathbf{l},T} p(\mathbf{y}, \mathbf{h}, \mathbf{l}, T \,|\, \mathbf{x}; \boldsymbol{\theta})$$

If the graph had small tree width, this marginalization could be performed exactly using the junction tree algorithm. However, even ignoring the long range connections to T, the tree width of a grid is the length of its shortest side and so exact inference is computationally prohibitive. The earlier described models, CRF and HRF, all have such a grid-like structure, which is of the same size as the input image; thus, we resort to approximate inference techniques. In particular, we considered both loopy belief propagation (LBP) and sequential tree-reweighted message passing (TRWS) [42]. Specifically, we compared the accuracy of max-product and the sum-product variants of LBP and the max-product form of TRWS (an efficient implementation of sum-product TRWS was not available – we intend to develop one for future work). We found that both max-product algorithms performed best on the CRF with TRWS outperforming LBP. However, on the HRF and LHRF models, the sum-product LBP gave significantly better performance than either max-product method. This is probably because the max-product assumption that the posterior mass is concentrated at the mode is inaccurate due to the uncertainty in the latent part variables. Hence, we used sum-product LBP for all LHRF experiments. Note, that the sum-product version of LBP is a special case of EP [58]. Further, the sum-product TRWS can also be considered a special case of power EP [55].

When applying LBP in the graph, we need to send messages from each h_i to T and update the approximate posterior $p(T)$ as the product of these; hence,

$$\log p(T) = \sum_{i \in V} \log \sum_{h_i} b(h_i)\ \psi^3(h_i, \mathrm{loc}(i, T)) \tag{5.5}$$

Table 5.1: Comparison of Different Discriminative Models.

	Parts-Based	Spatially Informative Parts	Models Local Spatial Coherence	Models Long Range Spatial Configuration
Unary Classifier	No	–	No	No
CRF	No	–	Yes	No
HRF	Yes	No	Yes	No
LHRF	Yes	Yes	Yes	Yes

where $b(h_i)$ is the product of messages into the ith node, excluding the message from T. To speed up the computation of $p(T)$, we make the following approximation:

$$\log p(T) \approx \sum_{i \in V} \sum_{h_i} b(h_i) \log \psi^3(h_i, \text{loc}(i, T)). \tag{5.6}$$

This posterior can now be computed very efficiently using convolutions.

Parameter Learning:

Given an image $\mathbf{x}$ with labels $\mathbf{y}$ and location map $\mathbf{l}$, the parameters $\boldsymbol{\theta}$ are learnt by maximizing the conditional likelihood $p(\mathbf{y}, \mathbf{l}|\mathbf{x}, \boldsymbol{\theta})$ multiplied by the Gaussian prior $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|0, \sigma^2\mathbf{I})$. Hence, we seek to maximize the objective function $\mathcal{F}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$, where $\mathcal{L}(\boldsymbol{\theta})$ is the log of the conditional likelihood.

$$\begin{aligned} \mathcal{F}(\boldsymbol{\theta}) &= \log p(\mathbf{y}, \mathbf{l}|\mathbf{x}; \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) = \log \sum_{\mathbf{h}} p(\mathbf{y}, \mathbf{h}, \mathbf{l}|\mathbf{x}; \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \\ &= -\log Z(\boldsymbol{\theta}, \mathbf{x}) + \log \sum_{\mathbf{h}} \tilde{p}(\mathbf{y}, \mathbf{h}, \mathbf{l}, \mathbf{x}; \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \end{aligned} \tag{5.7}$$

where:

$$\tilde{p}(\mathbf{y}, \mathbf{h}, \mathbf{l}, \mathbf{x}; \boldsymbol{\theta}) = \prod_i \psi_i^1(h_i, \mathbf{x}; \boldsymbol{\theta}_1) \phi(y_i, h_i) \psi^3(h_i, l_i; \boldsymbol{\theta}_3) \prod_{(i,j) \in E} \psi_{ij}^2(h_i, h_j, \mathbf{x}; \boldsymbol{\theta}_2).$$

We use gradient ascent to maximize the objective with respect to the parameters $\boldsymbol{\theta}$. The derivative of the log likelihood $\mathcal{L}(\boldsymbol{\theta})$ with respect to the model parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3\}$ can be written in terms of the features, single node marginals and pairwise marginals:

$$\begin{aligned} \frac{\delta \mathcal{L}(\boldsymbol{\theta})}{\delta \boldsymbol{\theta}_1(h')} &= \sum_{i \in V} \mathbf{g}_i(\mathbf{x}) \cdot (p(h_i = h'|\mathbf{x}, \mathbf{y}, \mathbf{l}; \boldsymbol{\theta}) - p(h_i = h'|\mathbf{x}; \boldsymbol{\theta})) \\ \frac{\delta \mathcal{L}(\boldsymbol{\theta})}{\delta \boldsymbol{\theta}_2(h', h'')} &= \sum_{(i,j) \in E} \mathbf{f}_{ij}(\mathbf{x}) \cdot (p(h_i = h', h_j = h''|\mathbf{x}, \mathbf{y}, \mathbf{l}; \boldsymbol{\theta}) - p(h_i = h', h_j = h''|\mathbf{x}; \boldsymbol{\theta})) \\ \frac{\delta \mathcal{L}(\boldsymbol{\theta})}{\delta \boldsymbol{\theta}_3(h', l')} &= \sum_{i \in V} p(h_i = h', l_i = l'|\mathbf{x}, \mathbf{y}, \mathbf{l}; \boldsymbol{\theta}) - p(h_i = h', l_i = l'|\mathbf{x}; \boldsymbol{\theta}) \end{aligned}$$

It is intractable to compute the partition function $Z(\boldsymbol{\theta}, \mathbf{x})$ and hence the objective function (5.7) cannot be computed exactly. Instead, we use the approximation to the partition function given by the LBP or TRWS inference algorithm, which is also used to provide approximations to the marginals required to compute the derivative of the objective. Notice that the location variable T comes into effect only when computing marginals for the unclamped model (where $\mathbf{y}$ and $\mathbf{l}$ are not observed), as the sum over

$\mathbf{l}$ should be restricted to those configurations consistent with a value of T. We have trained the model both with and without this restriction. Better detection results are achieved without it. This is for two reasons: including this restriction makes the model very sensitive to changes in image size and secondly, when used for detecting multiple objects, the restriction of a single object instance does not apply, and hence should not be included when training part detectors.

Image Features:

We aim to use image features which are informative about the part label but invariant to changes in illumination and small changes in pose. The features used in this work for both unary and pairwise potentials are Scale Invariant Feature Transform (SIFT) descriptors [51], except that we compute these descriptors at only one scale and do not rotate the descriptor, due to the assumption of fixed object scale and rotation. For efficiency of learning, we apply the model at a coarser resolution than the pixel resolution – the results given in this chapter use a grid whose nodes correspond to 2×2 pixel squares. For the unary potentials, SIFT descriptors are computed at the center of the each grid square. For the edge potentials, the SIFT descriptors are computed at the location half-way between two neighboring squares. To allow parameter sharing between horizontal and vertical edge potentials, the features corresponding to the vertical edges in the graphs are rotated by 90 degrees. Note, that the compatibilities between the parts are captured using these SIFT features. Rotating the features by 90 degrees for the vertical edges allows us to use the same set of parameters to encode the compatibilities of parts in both the horizontal and vertical directions.

Detecting multiple objects:

Our model assumes that a single object is present in the image. We can reject images with no objects by comparing the evidence for this model with the evidence for a background-only model. Specifically, for each given image we compute the approximation of $p(\text{model} \,|\, \mathbf{x}, \boldsymbol{\theta})$, which is the normalization constant $Z(\boldsymbol{\theta}, \mathbf{x})$ in (5.4). This model evidence is compared with the evidence for a model which labels the entire image as background $p(\text{noobject} \,|\, \mathbf{x}, \boldsymbol{\theta})$. By defining a prior on these two models, we define the threshold on the ratio of the model evidences used to determine if an object is present or absent. By varying this prior, we can obtain precision-recall curves for detection.

We can use this methodology to detect multiple objects in a single image, by applying the model recursively. Given an image, we detect whether it contains an object instance. If we detect an object, the unary potentials are set to uniform for all pixels labeled as foreground. The model is then reapplied to detect further object instances. This process is repeated until no further objects are detected.

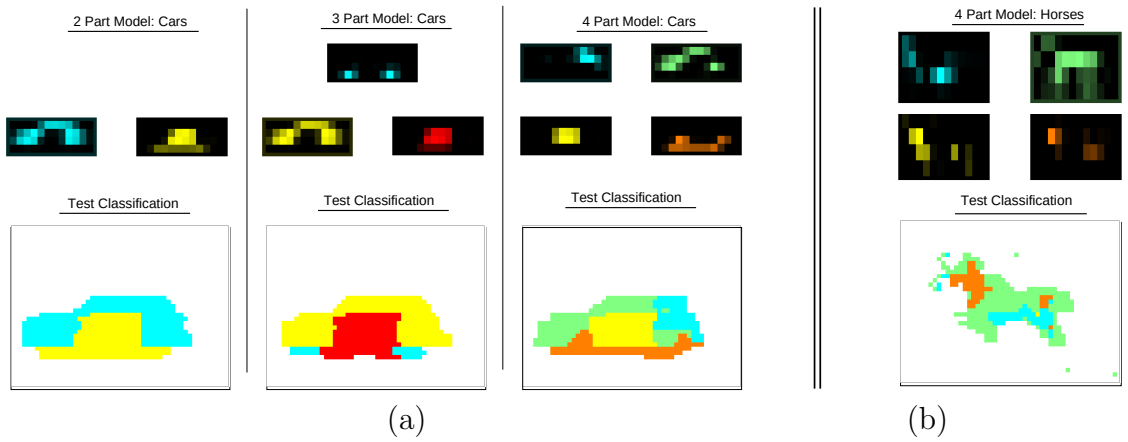


Figure 5-4: The learned discriminative parts for (a) Cars (side-view) and (b) Horses. The first row shows, for each model, the conditional probability $p(l|h)$, indicating where the parts occur within the object reference frame. Dark regions correspond to a low probability. The second row shows the part labeling of a test image for each model.

5.6 Experiments and Results

We performed experiments to (i) demonstrate the different parts learnt by the LHRF, (ii) compare different discriminative models on the task of pixelwise segmentation and (iii) demonstrate simultaneous detection and segmentation of objects in test images.

Training the models:

We trained each discriminative model on two different datasets: the TU Darmstadt car dataset [50] and the Weizmann horse dataset [8]. From the TU Darmstadt dataset, we extracted 46 images of different cars viewed from the side, of which 35 were used for training. The cars were all facing left and were at the same scale in all the images. To gain comparable results for horses, we used 46 images of horses taken from the Weizmann horse dataset, similarly partitioned into training and test sets. All images were resized to 100×75 pixels. Ground truth segmentations are available for both of these data sets, which were used either for training or for assessing segmentation accuracy. For the car images, the ground truth segmentations were modified to label car windows as foreground rather than background.

Training the LHRF on 35 images of size 100×75 took about 2.5 hours on a 3.2 GHz machine. Our implementation is in MATLAB except the loopy belief propagation, which is implemented in C. Once trained, the model can be applied to detect and segment an object in a 100×75 test image in around three seconds.

Learning discriminative parts:

Fig. 5-4 illustrates the learned conditional probability of location given parts $p(l \mid h)$ for two, three and four parts for cars and a four part model for horses. The results

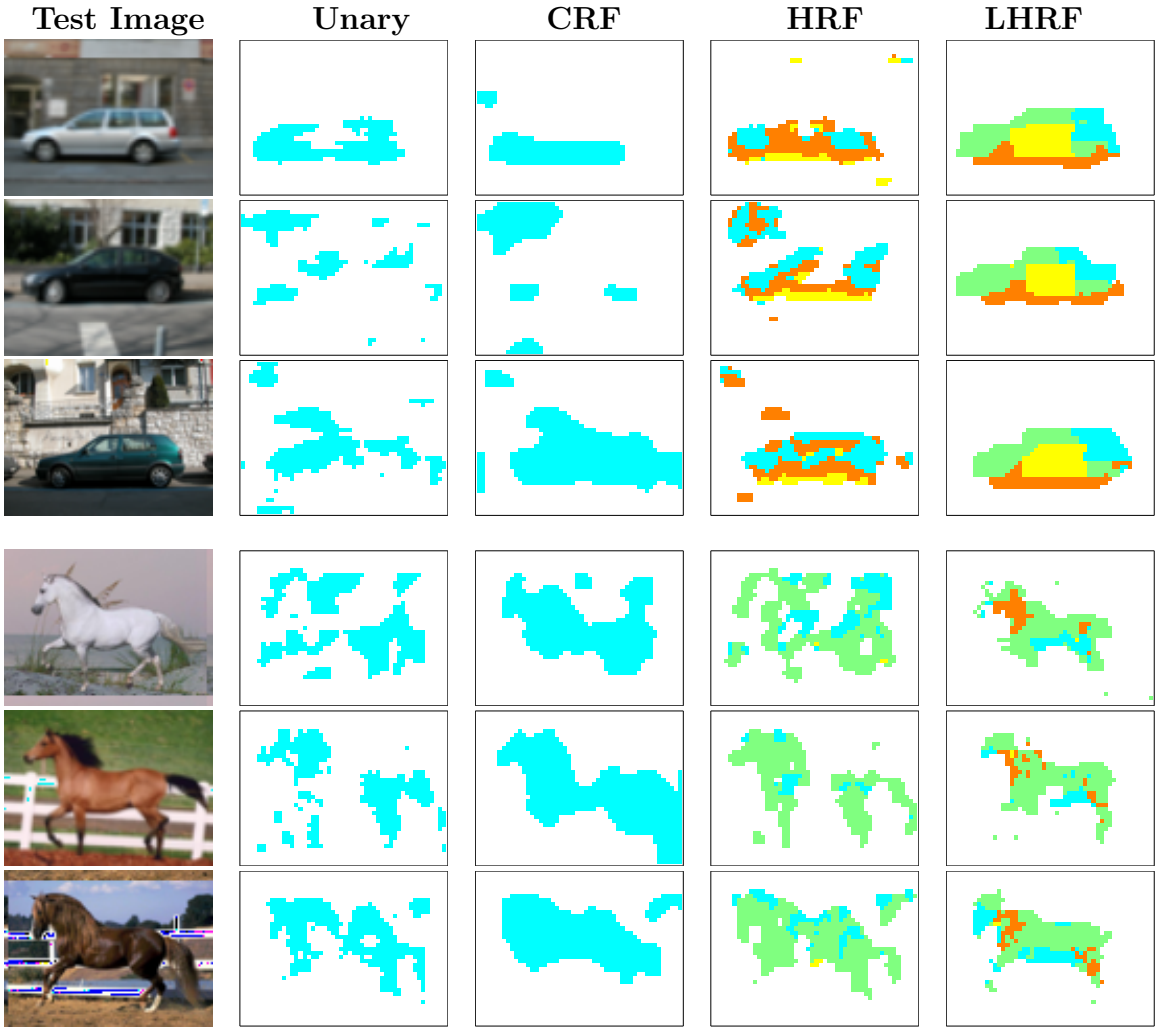


Figure 5-5: Segmentation results for car and horse images. The first column shows the test image and the second, third, fourth and fifth column correspond to different classifications obtained using unary, CRF, HRF and LHRF respectively. The colored pixels correspond to the pixels classified as foreground. The different colors for HRF and LHRF classification correspond to pixels classified as different parts.

show that spatially localized parts have been learned. For cars, the model discovers the top and the bottom parts of the cars and these parts get split into wheels, middle body and the top-part of the car as we increase the number of parts in the model. For horses, the parts are less semantically meaningful, although the learned parts are still localized within the object reference frame. One reason for this is that the images contain horses in varying poses and so semantically meaningful parts (e.g. head, tail) do not occur in the same location within a rigid reference frame.

Table 5.2: Segmentation accuracies for different models and approaches.

	Cars	Horses
Unary	86.1%	82.3%
CRF	86.5%	83.6%
HRF (4-Parts)	90.1%	85.4%
LHRF (4-Parts)	**95.2**%	**88.1**%
LOCUS [90]	94.0%	93.0%
Borenstein et al. [8]	-	93.6%

Table 5.3: Segmentation accuracies for LHRF with different numbers of parts.

Model	Cars
1-part LHRF	89.7%
2-part LHRF	93.0%
3-part LHRF	94.1%
4-part LHRF	95.2%

Segmentation accuracy:

We evaluated the segmentation accuracy for the car and horse training sets for the four different models of Fig. 5-2. As mentioned above, we selected the first 35 out of 46 images for training and used the remaining 11 to test. Segmentations for test images from the car and horse data sets are shown in Fig. 5-5. Unsurprisingly, using the unary model leads to many disconnected regions. The results using CRF and HRF have spatially coherent regions but local ambiguity in appearance means that background regions are frequently classified as foreground. Note that the parts learned by the HRF are not spatially coherent. Table 5.2 gives the relative accuracies of the four models where accuracy is given by the percentage of pixels classified correctly as foreground or background. We observe that LHRF gives a large improvement for cars and a smaller, but significant improvement for horses. Horses are deformable objects and parts occur varying positions in the location frame, reducing the advantage of the LHRF. For comparison, Table 5.2 also gives accuracies from [90] and [8] obtained for different test sets taken from the same dataset. Both of these approaches allow for deformable objects and hence gives better segmentation accuracy for horses, whereas our model gives better accuracy for cars. In Section 5.7 we propose to address this problem by using a flexible reference frame. Notice however that, unlike both [90] and [8] our model is capable of segmenting multiple objects from large images against a cluttered background (see section 5.5).

Table 5.3 shows the segmentation accuracy as we vary the number of parts in the LHRF and we observe that the accuracy improves with more parts. For models with more than four parts, we found that at most only four of the parts were used and hence the results were not improved further. It is possible that a larger training set would provide evidence to support a larger number of parts.

Simultaneous detection and segmentation:

To test detection performance, we used the UIUC car dataset [1]. This dataset includes 170 images provided for testing, containing a total of 200 cars, with some images containing multiple cars. Again, all the cars in this test set are at the same

scale.

Detection performance was evaluated for models trained on 35 images from the TU Darmstadt dataset. Fig. 5-6(a) shows detection accuracy for varying numbers of foreground parts in the LHRF model. From the figure, we can see that increasing the number of parts increases the detection performance, by exploiting both local and long-range part interactions. Fig. 5-6(b) compares the detection performance with other existing approaches, with the results summarized in Table 5.4. Our method is exceeded in accuracy only by the Liebe et al. method and then only when an additional validation step is used, based on an MDL criterion. This validation step could equally be applied in our case – without it, our method gives a 3.0% improvement in accuracy over Liebe et al. Note, that the number of examples used to train the model is less than used by all of the existing methods. Fig. 5-7 shows example detections and segmentations achieved using the 4-part LHRF.

5.7 Conclusions and Future Work

We have presented a novel discriminative method for learning object parts to achieve very competitive results for both the detection and segmentation tasks simultaneously, despite using fewer training images than competing approaches. The Located HRF has been shown to give improved performance over both the HRF and the CRF by learning parts which are informative about the location of the object, along with their spatial layout. We have also shown that increasing the number of parts leads to

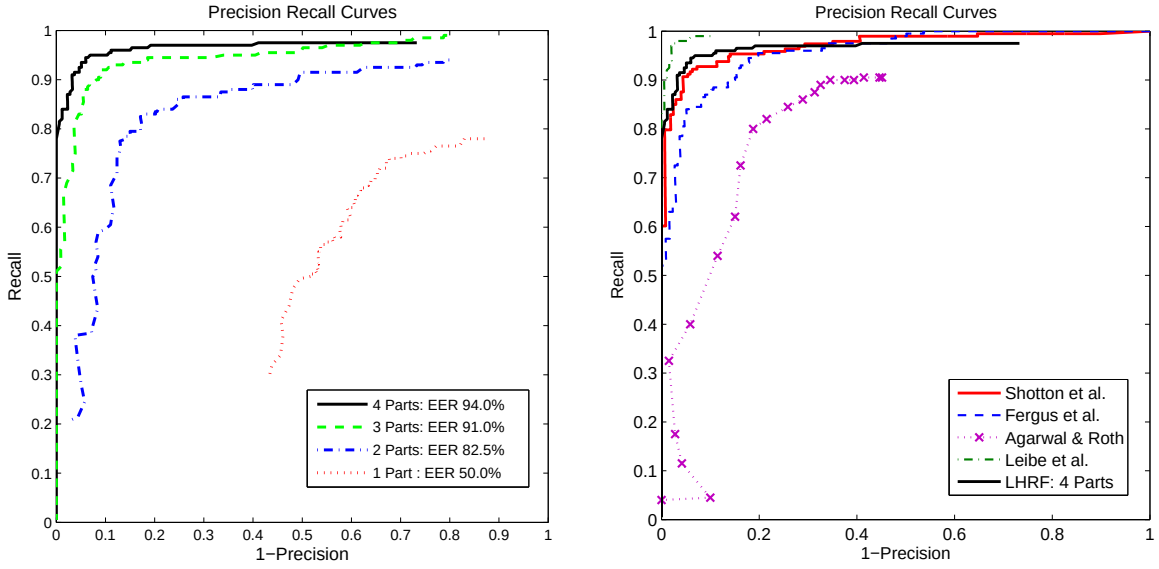


Figure 5-6: Precision-recall curves for detection on the UIUC dataset. (a) performance for different numbers of parts. Note that the performance improves as the number of parts increases. (b) relative performance for our approach against existing methods.

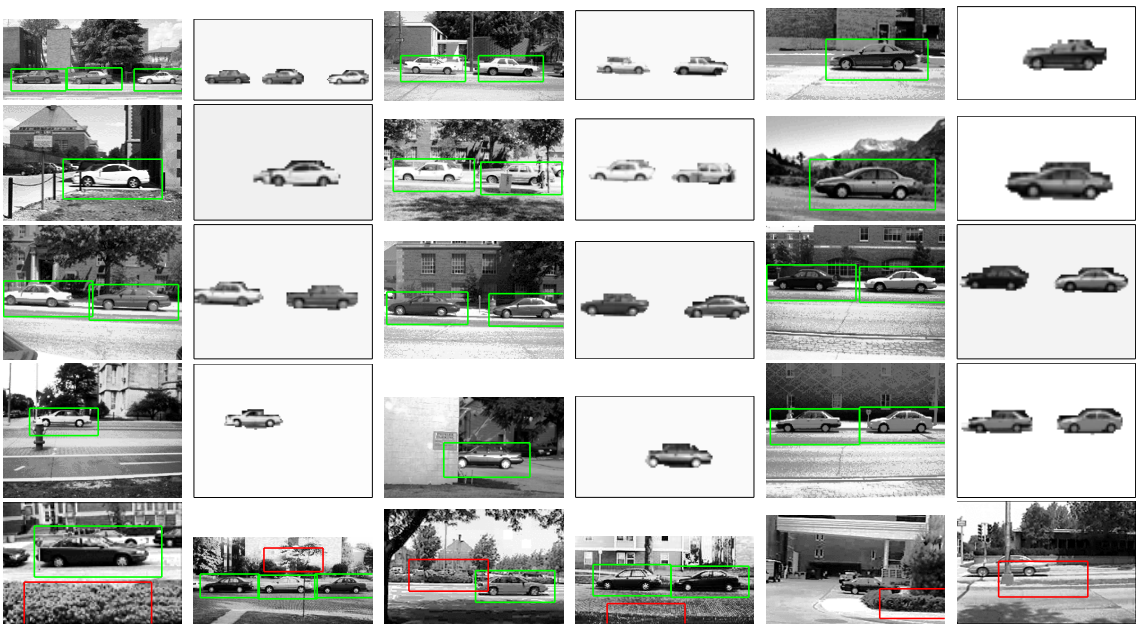


Figure 5-7: Examples of detection and segmentation on the UIUC dataset. The top four rows show correct detections (green boxes) and the corresponding segmentations. The bottom row shows example false positives (red boxes) and false negatives.

improved accuracy on both the segmentation and detections tasks. Additionally, once the model parameters are learned, our method is efficient to apply to new images.

One extension of this model that we plan to investigate is to introduce edges between the location labels. These edges would have asymmetric potentials encouraging the location labels to form into (partial) regular grids of the form of Fig. 5-3c. By avoiding the use of a rigid global template, such a model would be robust to significant partial occlusion of the object, to object deformation and would also be able to detect multiple object instances in one pass. We also plan to extend the model to multiple object classes and learn parts that can be shared between these classes.

Table 5.4: Comparison of detection performance.

	Number of Training Images	Equal Error Rate
Leibe et al.(MDL) [50]	50	97.5%
Our method	**35**	**94.0%**
Shotton et al. [77]	100	92.1%
Leibe et al. [50]	50	91.0%
Garg et al. [22]	1000	~88.5%
Agarwal & Roth [1]	1000	~79.0%

Chapter 6

Conclusion and Future Work

6.1 Summary

In this thesis we proposed various extensions to learn discriminative models with different types of incomplete data. Specifically, the proposed techniques handle four different kinds of incompleteness arising due to incorrect labels, incomplete features, partially labeled data and data annotated with coarse labels. The framework provides insights and connections to many existing methods in semi-supervised learning, sensor-fusion and discriminative modeling and one major advantage of viewing different methods in a single Bayesian framework is that we can use evidence maximization to perform model selection tasks, which were non-trivial earlier. Also, we can now exploit the modularity of the unified framework to combine different aspects of incompleteness and derive new methods for discriminative modeling.

In the following, we highlight the major contributions of the work:

- **Noisy Labels:** Section 2.1.2 discusses different noise models that can be used to handle different kinds of noise in the label. In section 2.4 Gaussian process classification is applied to the challenging problem of detecting quitters in a learning environment and we highlight kernel learning via the EM-EP algorithm. We also demonstrate on different datasets that the right choice of noise model can provide significant gains. We highlight scenarios in human-computer interaction and computer vision, where the incorrect labels might not lie near the decision boundary, thus, cannot be modeled using the linear or the quadratic slack. For these scenarios, we empirically show that the flipping noise model outperforms other slack based noise models. One of the advantages of this Bayesian framework is that we can use evidence maximization criteria to select the noise model and its parameters.

- **Incomplete Features:** The incompleteness due to missing features is handled by combining the decisions of experts trained on subsets of features, where the decision fusion ignores the classifiers that act on the missing features. In section 3.2, we proposed a unified approach using a mixture of Gaussian Processes for achieving sensor fusion under the challenging conditions of missing channels.

We provide an algorithm for Bayesian inference designed with a fast update of classification decisions based on variational and Gaussian approximations. In section 3.3.2 we demonstrate experiments on the task of classifying affective state of interest using multimodal information with missing data. The mixture of GP method outperformed feature level fusion based on imputation and several standard classifier combination schemes and obtained a recognition rate of over 86%. The experiments also show that there are significant gains when the incomplete data is incorporated in the training set. Finally, we demonstrate a multimodal affect recognition system capable of tracking facial features, postures and classifying affective states of children in a learning environment.

- **Partially Labeled Data:** We present extensions of Gaussian Process classification using data dependent regularization. The resulting framework connects many recent graph based semi-supervised learning efforts [79, 91, 92, 94]. Inference in the models with the non-Gaussian likelihoods are carried out using Expectation Propagation, which provides better approximations than the previously used Laplace approximation [94], with an additional benefit of deriving an EM-algorithm that exploits evidence maximization to learn the kernel and the hyperparameters of the model. Additionally using the same machinery, the framework can explicitly model and estimate the different types of label noise in the data. The experiments in section 4.4 show that the evidence maximization can be effectively exploited to learn hyperparameters of the model. On the half-moon dataset we demonstrate experimentally that when the number of available labeled data points are small the hyperparameter selection using evidence maximization should be preferred over the leave-one-out strategy. Further, as demonstrated on the affect dataset, we can conclude that modeling label noise appropriately in the model leads to significant gains. Finally, many different Graph-based approaches can be viewed in the Bayesian perspective, consequently suggesting that evidence maximization can be utilized to tune the hyperparameters of those methods as well.

- **Coarse Labels:** In section 5.4.1 the located hidden random field (LHRF) was introduced, which extends the traditional discriminative frameworks like conditional random fields and the hidden random random fields. These models have been shown to be specially effective on computer vision tasks and we use the proposed framework to address the problem of part-based object detection and recognition. Experiments in section 5.6 shows how LHRF learns object parts to achieve very competitive results for both the detection and segmentation tasks simultaneously, despite using fewer training images than competing approaches. The Located HRF has been shown to give improved performance over both the HRF and the CRF by learning parts which are informative about the location of the object, along with their spatial layout. We have also shown that increasing the number of parts leads to improved accuracy on both the segmentation and detection tasks. Additionally, once the model parameters are learned, our method is efficient to apply to new images.

6.2 Application Scenarios

The highly challenging problems addressed in this work are motivated by problems in affective computing, activity recognition and machine perception. Specifically, the proposed framework in the thesis can be used in any scenario that needs analysis and machine learning from data that arises from multiple channels, having very few labels and annotations, some of which might be incorrect or coarsely annotated. There are several research application scenarios that encounter incompleteness due to multimodal data, or the challenge of annotating the enormous amount of corpus with the labels of interest. Below, we briefly describe some of the possible applications where the techniques proposed in this thesis can be applied:

- **Affective Computing:** Pattern recognition for affective computing can very effectively use some of the methods described in this thesis. Besides, handling multimodal data, this thesis provides a rigorous framework to handle uncertainty and resolution of the labels which often plagues affect recognition problems. A lot of the methods described in this thesis can significantly reduce the pre-processing time by exploiting semi-supervised classification and also by allowing incorrectly labeled examples. Further, there is always a question of right set of emotions to detect in an affective computing scenario. The located hidden random field offers a solution where the exact emotions are discovered automatically based upon some coarser level categorization such as mood.

- **Sensor Networks and Context Aware Applications:** Sensor networks is an area where a lot of applications require fusion of information from multiple sources. Further, there might be other energy or environmental constraints that prohibit the sensors from sending complete observations. Then there are other applications that involve analyzing multi-sensory data collected through cellular phones and PDAs to learn human and social networks. Often, these scenarios result in a huge amount of data which can be very hard to annotate and semi-supervised learning and HRF based models can be applied to these applications as well.

- **Man-Machine Interaction:** The proposed framework can be an important component in a machine learning system that aims to interact naturally with people. Applications that need to learn to be adaptive to the users internal state would find this system useful. Similarly, the robotic characters and virtual avatars can incorporate the proposed methods and algorithms to enhance their perceptual capabilities. Note, that once trained the kernel classifiers for the affect recognition tasks takes around a few milliseconds to classifiy a new point. One of the advantage of the discriminative models is that the classification can be performed quickly; thus, they can be easily used with the real-time characters.

- **Computer Vision:** There are many applications in computer vision besides object detection and segmentation. There are imminent application of semi-supervised learning in tracking objects in videos. The consecutive frames in the

videos often lie in a low-dimensional manifold; hence, they can exploit the information from other unannotated frames to recover variables of interest. Further, often it is tedious to obtain very good annotations or labels in the videos due to insufficient resolution. The coarse and label noise models can be effectively used in those scenarios. Finally, Gaussian process regression, classification and their extensions can be applied to learn from examples effectively.

6.3 Future Work

There are many opportunities for future work relating to Gaussian process classification, their extensions and applications to real-world scenarios.

Gaussian process classification in general is more flexible than some of the other discriminative counterparts. And as shown in this thesis it can be extended to handle problem specific scenarios. However, to achieve a popularity equal to other methods such as SVMs, the computational time needs to be reduced. There have been sparse GP based classification methods [17, 48] and an interesting direction is to combine these approaches with the extensions proposed in this thesis.

Further despite anecdotal evidence, the convergence of Expectation Propagation for Gaussian process classification is not well understood theoretically. It can be shown that EP is an energy minimization procedure. However, the energy it minimizes is not necessarily convex. One interesting direction would be to convexify the EP objective using techniques such as power EP. Besides guaranteed convergence convex EP might also be able to provide insights to the issue of convergence in regular EP. Further, we should also investigate semidefinite relaxations of the EP objective as there are efficient methods to solve these problems and these can alleviate some of the computational challenges in Gaussian process classification.

Recently, there has been much interest in semi-supervised learning of the structured data. As a lot of models that handle structured data are probabilistic, an interesting question is how to extend the discriminative framework to handle these cases. Can we combine the generative models such as Hidden Markov Models (HMMs) and the discriminative framework, such that they "co-train" each other?

Evidence maximization has been a method of choice for model selection tasks. However, evidence is not well-defined for the class of models with improper priors. An interesting question is how can we perform model selection in these cases? Are there other metrics besides evidence maximization that we can use? Further, we need to compare different model selection criteria such as evidence maximization and leave-one-out cross-validation. Are there cases in which one measure should be preferred over others?

There are many extensions to the proposed LHRF model for the purpose of object detection. Currently, LHRF can handle rigid objects well, but might not perform well when the objects are deformable. One possible extension of this model is to introduce edges between the location labels. Such a model would be robust to significant partial occlusion of the object, to object deformation and would also be able to detect multiple object instances in one pass. Further, the LHRF can be extended

to model multiple object classes and learn parts that can be shared between these classes. Finally, we should also apply HRF and LHRF to activity recognition and other applications in human-computer interaction.

An exciting possibility is incorporation of active learning and other decision theoretic formulations that consider cost-benefit trade offs in obtaining different kinds of labels. Further, there are interesting possibilities where the mixture of GPs model is combined with different noise models, semi-supervised classification and other extensions proposed in these thesis.

Finally, this thesis shows the discriminative models can be expressive enough to handle different kinds of incompleteness. There are many applications in different domains such as sensor networks, social networks and other multimodal scenarios where the proposed method can be used effectively. Further depending upon the applications, the kind of incompleteness encountered can be different then the ones handled here, requiring us to develop other extensions which are theoretically and practically interesting.

Appendix A

Feature Extraction for Affect Recognition

There are many different features we use for affect recognition in learning environments. The non-verbal behaviors can be sensed through a camera, a pressure sensing chair, a pressure mouse and a device that measure the skin conductance. The camera is equipped with Infrared (IR) LEDs for structured lighting that help in real-time tracking of pupils and extracting other features from the face. Similarly the data sensed through the chair is used to extract information about the postures. Features are extracted from the activity that the subject is doing on the computer as well, which are then sent to a multimodal pattern analyzer that combines all the information to predict the current affective state. Below we describe the algorithms to extract features from these different sensors.

A.1 Facial Features & Head Gestures

The feature extraction module for face and head gestures is shown in figure A(a). We use an in-house built version of the IBM Blue Eyes Camera that tracks pupils unobtrusively using two sets of IR LEDs. One set of LEDs is on the optical axis and produces the red eye effect. The two sets of LEDs are switched on and off to generate two interlaced images for a single frame.The image where the on-axis LEDs are on has white pupils whereas the image where the off-axis LEDs are on has black pupils. These two images are subtracted to get a difference image, which is used to track the pupils. The pupils are detected and tracked using the difference image, which is noisy due to the motion artifacts and other specularities. We have elsewhere described [34] an algorithm to track pupils reliably using the noisy difference image. Once tracked, the pupil positions are passed to an HMM based head-nod and head-shake detection system,which provides the likelihoods of head-nods and head-shakes. Similarly, we have also trained an HMM that uses the radii of the visible pupil as inputs to produce the likelihoods of blinks. Further, we use the system described in [34] to recover shape information of eyes and the eyebrows.

Given pupil positions we can also localize the image around the mouth. Rather

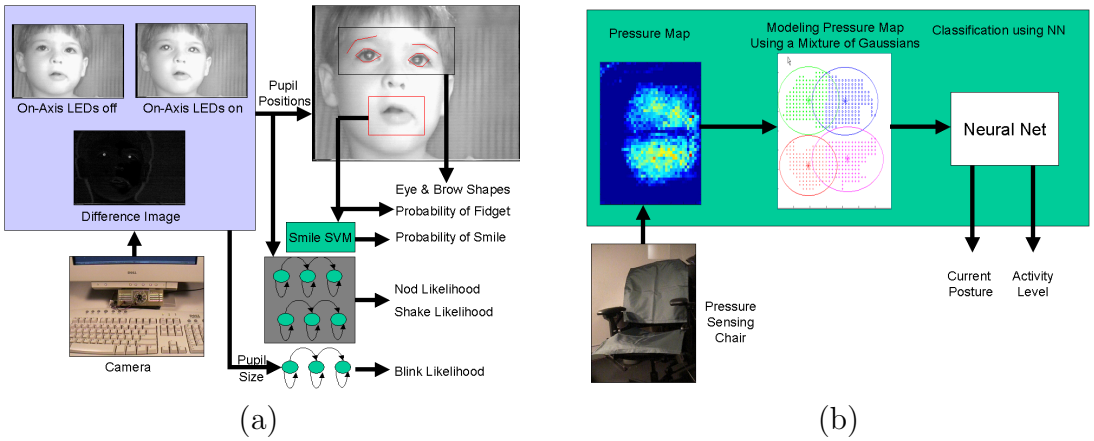


Figure A-1: Module to extract (a) facial features and (b) posture features.

than extracting the shape of the mouth we extract two real numbers which correspond to two kinds of mouth activities: smiles and fidgets. We look at the sum of the absolute difference of pixels of the extracted mouth image in the current frame with the mouth images in the last 10 frames. A large difference in images should correspond to mouth movements, namely mouth fidgets. Besides a numerical score that corresponds to fidgets, the system also uses a support vector machine (SVM) to compute the probability of smiles. Specifically, an SVM was trained using natural examples of mouth images, to classify mouth images as smiling or not smiling. The localized mouth image in the current frame is used as an input to this SVM classifier and the resulting output is passed through a sigmoid to compute the probability of smile in the current frame. The system can extract features in real time at 27-29 frames per second on a 1.8 GhZ Pentium 4 machine. The system tracks well as long as the subject is in the reasonable range of the camera. The system can detect whenever it is unable to find eyes in its field of view, which might occasionally happen due to large head and body movements. Also, sometimes the camera can only see the upper part of the face and cannot extract lower facial features, which happens if the subject leans forward. Due to these problems we often have missing information from the face; thus, we need an affect recognition system that is robust to such tracking failures.

A.2 The Posture Sensing Chair

Postures are recognized using two matrices of pressure sensors made by Tekscan. One matrix is positioned on the seat-pan of a chair; the other is placed on the backrest. Each matrix is 0.10 millimeters thick and consists of a 42-by-48 array of sensing pressure units distributed over an area of 41 x 47 centimeters. A pressure unit is a variable resistor, and the normal force applied to its superficial area determines its resistance. This resistance is transformed to an 8-bit pressure reading, which can be interpreted as an 8-bit grayscale value and visualized as a grayscale image.

Figure A(b) shows the feature extraction strategy used for postures in [59]. First, the pressure maps sensed by the chair are pre-processed to remove noise and the structure of the map is modeled with a mixture of Gaussians. The parameters of the Gaussian mixture (means and variances) are used to feed a 3-layer feed-forward neural network that classifies the static set of postures (for example, sitting upright, leaning back, etc.) and activity level (low, medium and high) in real time at 8 frames per second, which are then used as posture features by the multimodal affect classification module.

A.3 Detecting Interest: Final Set of Features

This is the new section I added to describe the features used in affect recognition. Table A.1 shows the extracted features from the face, the posture and the game state that were used for the task of recognizing the affective state of interest in learning environments (Chapter 3).

There are two features extracted from the game. They correspond to the state of the game and the level of difficulty. The state of the game indicates if a new game has started or ended, or whether the child asked for a hint, or whether the solution provided was correct etc. In this work, all the game features were manually extracted.

The posture recognition system can extract features at 8 frames per second (fps). The facial feature tracker performs at 27-29 frames per second, but we subsample the observations to 8 fps to extract the final features. Each sample in the database summarizes 8 seconds of activity. Specifically, we average the values of the features (see table A.1) observed continuously for 8 secs (64 frames). Note, that the samples do not explicitly encode any temporal dynamics of the feature over the 8 seconds of activity and only represent the average values.

The same database is used for experiments in Chapter 4, except that rather than using the automatically extracted features, we use manually coded action units based on the Facial Action Coding System (FACS)[20]. Specifically, from the face tracker we extract a 6 dimensional feature that encodes the average presence of an action unit (AU 1,2, 4, 5, 6, 7).

Table A.1: Extracted features for the task of detecting interest.

Face Tracker	**Posture Sensor**	**Game Information**
2D vector encoding standard deviation in (x, y) positions of pupils in the frame	1D vector encoding average activity level	1D vector encoding average difficulty level
Average likelihood of shakes Average likelihood of nods Average likelihood of blink	5D vector encoding average number of frames in postures:	6D vector encoding average number of frames in the game state:
Average probability of fidget Average probability of smile	{*leaning forward, sitting upright, leaning backward, slumped back, sitting on edge*}	{*new game, game running success, failure hint error, hint help*}
4D vector encoding the shape of the eyes and the brows		

Bibliography

[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *European Conference on Computer Vision*, 2002.

[2] Y. Altun, T. Hofmann, and A. Smola. Gaussian process classification for segmenting and annotating sequences. In *International Conference on Machine Learning*, 2004.

[3] M. J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, University College London, 2003.

[4] M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning, Special Issue on Clustering*, 56, 2004.

[5] M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In *Workshop on AI and Statistics*, 2005.

[6] A. Blum and S. Chawla. Learning with labeled and unlabeled data using graph mincuts. In *ICML*, 2001.

[7] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Conference on Computational Learning Theory*, 1998.

[8] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *Proceedings IEEE workshop on Perceptual Organization in Computer Vision, CVPR*, 2004.

[9] L. Breiman. Bagging predictors. *Machine Learning*, 26(2), 1996.

[10] T. W. Chan and A.B. Baskin. *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education*, chapter 1: Learning companion systems. 1990.

[11] O. Chapelle, J. Weston, and B. Scholkopf. Cluster kernels for semi-supervised learning. *Advances in Neural Information Processing Systems*, 15, 2003.

[12] I. Cohen, F. G. Cozman, N. Sebe, M. C. Cirelo, and T. S. Huang. Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction. *PAMI*, 26(12):1553–1567, 2004.

[13] C. Conati. Probabilistic assessment of user's emotions in educational games. *Applied Artificial Intelligence, special issue on Merging Cognition and Affect in HCI*, 16, 2002.

[14] A. Corduneanu and T. Jaakkola. On information regularization. In *Uncertainity in Artificial Intelligence*, 2003.

[15] F. Cozman, I. Cohen, and M. Cirelo. Semi-supervised learning of mixture models. In *Proceedings of International Conference on Machine Learning*, 2003.

[16] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *Computer Vision and Pattern Recognition*, 2005.

[17] L. Csató. *Gaussian Processes - Iterative Sparse Approximation*. PhD thesis, Aston University, 2002.

[18] L. Csató, E. Fokoué, M. Opper, B. Schottky, and O. Winther. Efficient approaches to Gaussian process classification. In *Neural Information Processing Systems*, 1999.

[19] G Dietterich, T., Lathrop R. H., and T. Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[20] P. Ekman and W. V. Friesen. *The Facial Action Coding System: A Technique for Measurement of Facial Movement*. Consulting Psychologists Press, San Francisco, CA, 1978.

[21] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition*, 2003.

[22] A. Garg, S. Agarwal, and T. S. Huang. Fusion of global and local information for object detection. In *International Conference on Pattern Recognition*, 2002.

[23] M. N. Gibbs and D. J. C. MacKay. Variational Gaussian process classifiers. *IEEE-NN*, 11(6):1458, November 2000.

[24] T. Graepel. Kernel matrix completion by semidefinite programming. In *ICANN*, 2002.

[25] J. Han and B. Bhanu. Statistical feature fusion for gait-based human recognition. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2004.

[26] R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001.

[27] L. Hong and A. K. Jain. Integrating faces and fingerprints for personal identification. *Pattern Analysis and Machine Intelligence*, 20(12), 1998.

[28] T. S. Huang, L. S. Chen, and H. Tao. Bimodal emotion recognition by man and machine. In *ATR Workshop on Virtual Communication Environments*, 1998.

[29] Y. Ivanov, T. Serre, and J. Bouvrie. Confidence weighted classifier combination for multi-modal human identification. Technical Report AI Memo 2005-035, MIT Computer Science and Artificial Intelligence Laboratory, 2005.

[30] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[31] T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, 2003.

[32] A. Kapoor, H. Ahn, and R. W. Picard. Mixture of Gaussian processes to combine multiple modalities. In *Workshop on MCS*, 2005.

[33] A. Kapoor, S. Mota, and R. W. Picard. Towards a learning companion that recognizes affect. In *AAAI Fall Symposium*, Nov 2001.

[34] A. Kapoor and R. W. Picard. Real-time, fully automatic upper facial feature tracking. In *Automatic Face and Gesture Recognition*, May 2002.

[35] A. Kapoor and R. W. Picard. Multimodal affect recognition in learning environments. In *ACM Conference on Multimedia*, Nov 2005.

[36] A. Kapoor, R. W. Picard, and Y. Ivanov. Probabilistic combination of multiple modalities to detect interest. In *Proceedings of International Conference on Pattern Recognition*, August 2004.

[37] A. Kapoor, A. Qi, H. Ahn, and R. W. Picard. Hyperparameter and kernel learning for graph based semi-supervised classification. In *Advances in Neural Information Processing Systems*, 2005.

[38] A. Kapoor and J. Winn. Located hidden random fields: Learning discriminative parts for object detection. In *European Conference on Computer Vision*, 2006.

[39] C. Kemp, T. Griffiths, S. Stromsten, and J. Tenenbaum. Semi-supervised learning with trees. In *Advances in Neural Information Processing Systems*, 2003.

[40] H. Kim and Z. Ghahramani. The EM-EP algorithm for Gaussian process classification. In *ECML*, 2004.

[41] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[42] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. In *Workshop on Artificial Intelligence and Statistics*, 2005.

[43] R. Kondor and Lafferty J. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, 2002.

[44] M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, 2005.

[45] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *International Conference on Computer Vision*, 2003.

[46] M. Kuss and C. E. Rasmussen. Assesing approximations for Gaussian process classification. In *Advances in Neural Information Processing Systems*, 2005.

[47] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

[48] N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process method: Informative vector machines. In *Advances in Neural Information Processing Systems*, volume 15, 2003.

[49] N. D. Lawrence and M. I. Jordan. Semi-supervised learning via Gaussian processes. In *Advances in Neural Information Processing Systems*, 2004.

[50] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision*, 2004.

[51] D. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, 1999.

[52] Opper M. A Bayesian approach to online learning. In *Online Learning in Neural Networks*. Cambridge University Press, 1998.

[53] Szummer. M. and T. Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[54] D. J. Miller and L. Yan. Critic-driven ensemble classification. *Signal Processing*, 47(10), 1999.

[55] T. P. Minka. Divergence measures and message passing. Technical Report MSR-TR-2005-173, Microsoft Research.

[56] T. P. Minka. EP: A quick reference. http://research.microsoft.com/~minka/papers/ep/minka-ep-quickref.pdf.

[57] T. P. Minka. Expectation Propagation for approximate Bayesian inference. In *Uncertainity in Artifical Intelligence*, 2001.

[58] T. P. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachussetts Institute of Technology, 2001.

[59] S. Mota and R. W. Picard. Automated posture analysis for detecting learner's interest level. In *Workshop on Computer Vision and Pattern Recognition for Human-Computer Interaction*, June 2003.

[60] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998.

[61] K. Nigam, A. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[62] N. Oliver, A. Garg, and E. Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. In *Proceedings of the International Conference on Multimodal Interfaces*, 2002.

[63] M. Opper and O. Winther. Mean field methods for classification with Gaussian processes. *Advances in Neural Information Processing Systems*, 11, 1999.

[64] M. Pantic and L. J. M. Rothkrantz. Towards an affect-sensitive multimodal human-computer interaction. *Proceedings of IEEE*, 91(9), 2003.

[65] R. W. Picard, E. Vyzas, and J. Healey. Toward machine emotional intelligence: Analysis of affective physiological state. *Pattern Analysis and Machine Intelligence*, 2001.

[66] Y. Qi, Minka T. P., R. W. Picard, and Z. Ghahramani. Predictive automatic relevance determination by expectation propagation. In *Proceedings from the International Conference on Machine Learning*, 2004.

[67] Y. Qi, M. Szummer, and T. P. Minka. Bayesian conditional random fields. In *Workshop on AI and Statistics*, 2005.

[68] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems*, 2004.

[69] C. Reynolds. The sensing and measurement of frustration with computers. Master's thesis, MIT, 2001.

[70] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3), 2004.

[71] D. Rubin. *Multiple Imputation for Nonresponse in Surveys*. Wiley, NY, 1987.

[72] P. Rujan. Playing billiards in version space. *Neural Computation*, 9:99–122, 1997.

[73] R. Schapire. A brief introduction to boosting. In *Proceedings of International Conference on Algorithmic Learning Theory*, 1999.

[74] M. Seeger. Input-dependent regularization of conditional density models. Technical report, University of Edinburgh, 2001.

[75] M. Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(2), 2004.

[76] M. Seeger and M. I. Jordan. Sparse gaussian process classification with multiple classes. Technical Report TR-661, Department of Statistics, University of California at Berkeley, 2004.

[77] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *International Conference on Computer Vision*, 2005.

[78] V. Sindhwani, W. Chu, and S. S. Keerthi. Semi-supervised Gaussian processes. Technical Report YRL-2005-60, Yahoo! Research Labs, 2005.

[79] A. Smola and R. Kondor. Kernels and regularization on graphs. In *COLT*, 2003.

[80] M. Strauss, C. Reynolds, S. Hughes, K. Park, G. McDarby, and R. W. Picard. The handwave bluetooth skin conductance sensor. In *ACII*, pages 699–706, 2005.

[81] M. Szummer. Learning diagram parts with hidden random fields. In *International Conference on Document Analysis and Recognition*, 2005.

[82] M. Szummer and T. Jaakkola. Information regularization with partially labeled data. *Advances in Neural Information Processing Systems*, 15, 2003.

[83] K. Toyama and E. Horvitz. Bayesian modality fusion: Probabilistic integration of multiple vision algorithms for head tracking. In *Proceedings of the Asian Conference on Computer Vision*, 2000.

[84] V. Tresp. Mixture of Gaussian processes. *Advances in Neural Information Processing Systems*, 13, 2001.

[85] K. Tsuda, S. Akaho, and K. Asai. The EM algorithm for kernel matrix completion with auxillary data. *Journal of Machine Learning Research*, 4, 2003.

[86] V. N. Vapnik. *Statistical Learning Theory.* Wiley, NY, 1998.

[87] C. K. I Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

[88] C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems*, 1996.

[89] D. Williams, X. Liao, Y. Xue, and Carin L. Incomplete-data classification using logistic regression. In *ICML*, 2005.

[90] J. Winn and N. Jojic. Locus: Learning object classes with unsupervised segmentation. In *International Conference on Computer Vision*, 2005.

[91] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. *Advances in Neural Information Processing*, 16, 2004.

[92] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, 2003.

[93] X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *NIPS*, 2004.

[94] X. Zhu, J. Lafferty, and Z. Ghahramani. Semi-supervised learning: From Gaussian fields to Gaussian processes. Technical Report CMU-CS-03-175, CMU, 2003.