A Survey on Inductive Semi-supervised Learning

Gholamreza Haffari

March 6, 2006

Contents

1 Introduction

The problem of learning in the presence of labeled and unlabeled data, also known as Semi-Supervised Learning, has recently attracted a great deal of attention. In many real life machine learning tasks, providing a labeled training set is costly while there is a lot of unlabeled data which can be obtained easily. For example in part-of-speech tagging it takes a lot of time to annotate sentences with their part of speech tags but a huge amount of unannotated sentences can easily be provided by crawling the web.

The natural question comes into mind is the possibility of taking the advantage of unlabeled data in order to construct more accurate classifiers. The focus of research on semi-supervised learning in machine learning community has been twofold: on one hand analyzing the situations in which unlabeled data can be useful, and on the other hand, giving learning algorithms which can actually do this. These proposed learning algorithms are often very different with respect to the assumptions they make about the problem.

The purpose of this report is to introduce the semi-supervised learning problem. However, in this sub-field of machine learning there are several approaches and we limit our scope of attention to the cases where the result of the method is a *classifier* and not just the labeling of the current unlabeled data. Hence, the output classifier can be used to label the current unlabeled data as well as new unseen data. This point of view is in contrast to the methods which try to label only the current unlabeled data, in other words, they do not produce a classifier. When new data comes in, these methods have to be run again from scratch to be able to label new points[1].

Another distinction we made is *classifier based* methods and *data based* methods. Roughly speaking, classifier based approaches start from an initial classifier (or an ensemble of classifiers), and then iteratively try to produce new training data by injecting (noisy) labels to unlabeled data, and iteratively learn new classifier(s). On the other hand, data based methods try to reveal the *geometry* (if there is any) of

[1] In [CWS03] a solution is proposed for this problem. When a new unseen point is given, approximate it by writing it as a combination of old labeled and unlabeled points, and then use this approximation to label the point (based on the labels of old points).

the distribution of data[2] with the help of unlabeled data, make some assumptions about a good function class w.r.t the geometry, and then find the best function in this function class.

The theme of this report is as following. First we discuss situations where additional unlabeled data is expected to help the classification task, and also where it is harmful. The next section contains a survey on classifier based methods followed by a section with survey on data based methods. Semi-supervised learning in structured domains is discussed in the next section. We finish the report by making some concluding remarks.

1.1 Problem Definition and Notation

For each labeled instance (x, y), we call $x \in \mathcal{X}$ the input instance and $y \in \mathcal{Y}$ its class label. Input instances belong to the instance space $\mathcal{X}$, and class labels belong to $\mathcal{Y} = \{c_1, ..., c_k\}$ where $|\mathcal{Y}| = k$ and each c_i is a class label. We denote the set of labeled instances $\{(x_1, y_1), ..., (x_l, y_l)\}$ as D_l with l as its cardinality, and the set of unlabeled instances $\{x_{l+1}, ..., x_{l+u}\}$ as D_u where u is its cardinality. We may also refer to $\{x_1, .., x_l\}$ as X_l, and $\{y_1, ..., y_l\}$ as Y_l. The labels of unlabeled instances $y_{l+1}, ..., y_{l+u}$ are hidden (latent).

Often it is assumed that there is a fixed but unknown probability distribution $P_{x,y} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ where each labeled instance has been drawn independently and identically distributed (i.i.d.) based on it. Moreover, each unlabeled instance x' has been generated i.i.d. based on the marginal distribution $P_x(x') = \sum_{y' \in \mathcal{Y}} P_{x,y}(x', y')$. For simplicity, we will write $P(x', y')$ instead of $P_{x,y}(x', y')$ and $P(x')$ instead of $P_x(x')$. Having a sample containing labeled and unlabeled data $D = D_l \cup D_u$, the aim of semi supervised learning is to find a *good* classifier $h : \mathcal{X} \to \mathcal{Y}$ from a set of possible classifiers $\mathcal{H}$, which maps input instances to class labels correctly.

More formally, assume there is a loss function $loss(y, y^t, x)$ which gives the cost of assigning a (predicted) label y instead of the true label y^t to an input instance x. The

[2]Like a *manifold* on which input data might be sitting.

goal of semi-supervised learning (as well as fully supervised learning) is to produce the best classifier which has the minimum possible expected loss or *risk*:

$$\arg\min_{h\in\mathcal{H}} E_{(x',y')\sim P(x,y)}[loss(h(x'), y', x')]$$

It is worth mentioning that the only difference between fully supervised and semi supervised learning is that, in the training sample for the semi-supervised learning we have access to unlabeled data as well as labeled data but for the supervised learning we only have access to labeled data.

2 Theoretical insights

Traditionally, there are two different learning scenarios in machine learning problems. First, supervised learning (or learning with a teacher) where we are given a sample containing input points and their corresponding outputs, and asked to find the (probabilistic) predictive relationship between input and output. Second, unsupervised learning (or learning in the absence of a teacher) where we are only given some input points, and asked to find the *patterns* in them. Assume that data is sent from a source to a destination, and the aim is to use the communication channel as less as possible. Good patterns in the data (which are generated in the source) lead to efficient compression and transmission through the channel.

The semi-supervised learning problem is somewhere between the two extreme cases of supervised and unsupervised learning ([See00]). Compared to the unsupervised learning problem, there is more information because some labels are given. Compared to supervised learning problem we can have more or less information depending of what we consider as the potential input for the supervised learning problem.

There are two main strategies to attack supervised learning problems: generative and discriminative model based approaches. In generative model based approaches, the aim is to build a model for the joint probability distribution of input and output $P(x, y|\theta)$ which often belongs to a family of models parameterized

by θ, and then build the classifier on top of it, e.g. using Bayes optimal decision rule $y^* = \arg\max_y P(y|x,\theta) = \arg\max_y P(x,y|\theta)$. In contrast, discriminative model based approaches only focus on the conditional probability distribution $P(y|x,\theta)$, and try to directly learn the relationship of input and output. Generative models often have lots of parameters because they have to approximate the input-output distribution, but discriminative models need less parameters because they do not need to model the density over input space.

There is an argument in [See00] which essentially says discriminative based approaches cannot be used to solve semi-supervised learning unless, in some way, they use the information of input density in the process of constructing the classifier. Intuitively it can be explained as follows: in discriminative model based approaches what we care about is the conditional probability distribution $P(y|x)$ but unlabeled data gives only information about the input density $P(x)$, so somehow we have to use this information in constructing the conditional probability distribution. However, generative models have this potential to be used for solving semi-supervised learning problem because they use the information of input density.

In this section, we first describe the theoretical framework proposed in [BB05, CZS06] as a possible formal framework for thinking about semi-supervised learning problem. Then for the particular case of generative models and *Maximum Likelihood* (ML) estimator, some statistical analysis is given which investigates when unlabeled data can be useful and when it can be harmful in the process of building classifiers.

2.1 Augmented PAC model

Supervised learning problems and algorithms can be expressed and analyzed in the Probably Approximately Correct (PAC) model. The standard PAC framework is distribution free, i.e. the results do not depend on the input distribution. Loosely speaking, this framework is only suitable for analyzing *discriminative* learning algorithms. It is a good idea to change the standard PAC framework in such a way that the new framework enables us to think about semi-supervised learning problems

and algorithms. This attempt has been done in [BB05, CZS06] by incorporating the distribution over the input space $\mathcal{X}$ into the PAC framework via the notion of *compatibility* function χ.

In the standard PAC framework the only a priori assumption is that the output hypothesis must be chosen in some hypothesis space $\mathcal{H}$. The new framework goes further and assumes that the final hypothesis must be compatible with the input distribution based on the compatibility function $\chi : \mathcal{H} \times \mathcal{X} \to [0, 1]$. The function χ gets an input instance $x \in \mathcal{X}$ and a hypothesis $h \in \mathcal{H}$, and produces a number which shows how reasonable the hypothesis is w.r.t the input point x, furthermore the quantity $\chi(h, P) = E_{x \sim p(x)}[\chi(h, x)]$ represents the plausibility of the hypothesis h w.r.t the distribution over the input space. The amount of incompatibility $1-\chi(h, P)$ can be considered as the unlabeled error rate of the hypothesis h, i.e. how likely a priori we think that h is *not* the target hypothesis.

As an example suppose training data live in some Euclidean space $\mathbb{R}^d$ and consider the class of linear separators as the hypothesis space. The prior belief might be that the target function should separate the two classes of data points with the margin at least γ. So the compatibility $\chi(h, x)$ is one if the distance of x to the hyperplane $h^T \cdot z = 0$ is greater than γ and is zero otherwise (here the hypothesis $h \in \mathbb{R}^d$ is just a vector of coefficients). Consequently, the incompatibility $1-\chi(h, P)$ shows how much mass of the input distribution is placed on the space within the distance γ from the hyperplane.

Intuitively unlabeled data helps to reduce the size of hypothesis space by just keeping the plausible ones, and labeled data is used to choose a good hypothesis from these plausible hypotheses. Denote by $\mathcal{H}_{P,\chi}(\tau)$ the subset of the concept class $\mathcal{H}$ which have the incompatibility less than τ. Often the input distribution is unknown so the *empirical* measure of compatibility (or incompatibility) $\hat{\chi}(h, S) = \frac{1}{l+u}\sum_1^{l+u} \chi(h, x_i)$ is calculated based on the available sample S; therefore $\mathcal{H}_{S,\chi}(\tau)$ denotes the subset of the hypothesis class $\mathcal{H}$ which have the empirical incompatibility less than τ. The sample complexity results (i.e. how much labeled and unlabeled data we need to build a good

classifier) are expressed based on some measure of complexity of $\mathcal{H}_{S,\chi}(\tau)$ or $\mathcal{H}_{P,\chi}(\tau)$ as well as other quantities such as the allowed approximation error ϵ and confidence parameter δ of the output hypothesis. Several theorems are stated in [BB05], and the conclusion is that unlabeled data is helpful when (i) the target hypothesis is highly compatible with the notion of compatibility χ, (ii) not many hypotheses in $\mathcal{H}$ have a low unlabeled error rate (which depends on the input distribution P as well as χ), and (iii) there is enough unlabeled data to approximate well the empirical unlabeled error rate.

The generative model approach to semi-supervised learning can be fit into this framework ([CZS06]) as an extreme case. Usually generative model based approaches assume that the marginal distribution over the input space is an identifiable mixture:

$$P(x|\theta) = \sum_j \alpha_j P(x|\theta_j)$$

where $\sum_j \alpha_j = 1$ and all mixture components belong to a known parametric family of models. Moreover each mixture component is labeled with only one class. Therefore the strategy is to first discover these mixture components based on unlabeled data, and then assign labels to components based on labeled data. The scoring function which is used in the first phase (to find the mixture components) can be considered as the degree of compatibility of hypotheses. As an example, the likelihood score can be used as the compatibility function $\chi(h_\theta, D) = E_{x\sim D}[\log P(x|\theta)]$ where h_θ is the resulting classifier based on $P(x|\theta)$, and D refers to the distribution over the input space (it is used instead of P which is over (x, y) to prevent confusion).

2.2 When is Semi-supervised learning expected to help?

The basic question comes into mind is the value of unlabeled data for the classification task [CCS+04]. Intuitively, it seems that having more (unlabeled) data should be useful; however as we will see, it is not true for all situations. In this section for the particular case of generative models approach and Maximum Likelihood (ML) estimator, we will discuss situations in which unlabeled data improves the accuracy

of the classifier, and also situations in which unlabeled data degrades the performance of the learned classifier.

Consider the scenario where we estimate the joint probability distribution of input and output $P(x, y)$ within a parametric model $P(x, y|\theta)$, and construct the classifier using the Bayes decision rule:

$$y^* = h(x) = \arg\max_{y \in \mathcal{Y}} P(y|x) = \arg\max_{y \in \mathcal{Y}} P(x, y) \tag{1}$$

Assume 0-1 loss, i.e. in the case of correct prediction the loss is zero and otherwise it is one, consequently the risk of the classifier $\int_{\mathcal{X}\times\mathcal{Y}} loss(h(x), y, x)\, dP(x, y)$ is actually the probability of making an error. It has been shown that the Bayes decision rule (1) has the minimum possible probability of error (see [CCS+04] for references), denote this minimum error by R^*.

Suppose training data is generated i.i.d. based on $P(x, y)$, and for each instance the label is kept by probability λ and removed by probability[3] $1 - \lambda$. The probability of generating the sample is:

$$P(D_l, D_u|\theta) = \prod_{i=1}^{l} \lambda P(x_i, y_i|\theta) \cdot \prod_{i=l+1}^{l+u} (1 - \lambda) P(x_i|\theta)$$

$$L(\theta) \triangleq \log P(D_l, D_u|\theta)$$

To select the best probability distribution within the assumed parametric family, Maximum Likelihood estimation rule is used $\hat{\theta} = \arg\max_\theta L(\theta)$. Denote by $\hat{\theta_N}$ the estimated value for the parameters when the sample size is $N = l + u$. We investigate the behavior of ML estimation for partially labeled sample in two cases: when the size of sample is infinite (asymptotic behavior) and when it is finite. But let us first consider a special case where we assume that the *true* parametric family of distributions is already known, and is a family of identifiable mixtures of two components (these assumptions are very strong).

[3] λ could be assumed to be different for each instance which is the case considered, for example, in [RJZH04].

2.2.1 Special case: Mixture of two Components

Consider the special case where there are two classes, and each class has been generated based on a probability density function in a (possibly parametric) family of models $\mathcal{M}$. The generation process first tosses a coin with $\alpha \in [0, 1]$ as the probability of coming head, then it generates an instance of the first class w.r.t $p_1(x)$ if head comes up, otherwise it generates an instance of the second class w.r.t $p_2(x)$. Assume the following family of mixtures is identifiable:

$$\mathcal{N} = \{\beta q_1 + (1 - \beta)q_2 | q_1, q_2 \in \mathcal{F}, \beta \in [0, 1]\}$$

Identifiability means that given a $g \in \mathcal{N}$, we can uniquely determine its two constituent components in $\mathcal{M}$ and mixing parameter β.

Suppose we already know that mixture components are from the family $\mathcal{M}$, what is the value of labeled and unlabeled data based on the best classifier which can be build (i.e. Bayes classifier)? In [CC95, Cas94] this question is answered step by step.

Denote the risk of the best classifier one can build from l labeled data and u unlabeled data by $R(l, u)$. Without seeing any labeled instance the risk is $R(0, u) = R(0, \infty) = \frac{1}{2}$ which means that in the absence of labeled data, the best thing we can do is a random guess. After receiving the first labeled instance, the risk goes down to $R(1, \infty) = 2R^*(1 - R^*)$. Upon receiving more labeled instances, the risk of the best classifier converges exponentially fast to the Bayes optimal risk:

$$R(l, \infty) - R^* = \exp\Big(- lB + o(l)\Big)$$

where B is the following constant:

$$B = -\log\Big(2\sqrt{\alpha(1 - \alpha)} \int_{\mathcal{X}} \sqrt{p_1(x)p_2(x)}\, dx\Big)$$

The idea of the classification in the presence of infinite unlabeled instances is to first use unlabeled data to (approximately) find the underlying marginal distribution over the input space $\mathcal{X}$. Then based on the identifiability assumption, the two mixture components and mixing parameter can be uniquely determined. Now, the only problem left is to decide which component is responsible for the first class and which one

is responsible for the second class. Labeled data is used in this phase to assign labels to mixture components.

In the general case of finite labeled and unlabeled data, if some technicalities are hold (for the number of labeled l and unlabeled u instances, and for the probability density functions q_1 and q_2) the risk of the best classifier converges to the risk of the Bayes classifier as follows:

$$R(l,u) - R^* = O\Big(\frac{1}{u}\Big) + \exp\Big(- lB + o(l)\Big) \tag{2}$$

The first term in the above expression can be traced to the "uncertainty in recovering the decision regions from data", and the second term to the "uncertainty in labeling the recovered regions" ([Cas94]).

If the number of unlabeled data grows faster than $\exp(lB)$, the right hand side of the expression (2) is dominated by the term which is related to the labeled data. In other words, the difference between the risk of the optimal classifiers converges exponentially to zero in the number of labeled data l. On the other hand, if $u = o(\exp lB)$, the difference of the risk of the best classifiers depends on the number of unlabeled data and converges to zero with the rate $O(u^{-1})$.

2.2.2 Asymptotic behavior

It is shown in [CCS+04] that as the size of the sample tends to infinity, the *limiting* value of the maximum likelihood estimator $\theta^* = \lim_{N\to\infty} \hat{\theta}_N$ will be:

$$\theta^* = \arg\max_{\theta} \lambda \underbrace{E[\log P(x,y|\theta)]}_{L_l(\theta)} + (1-\lambda) \underbrace{E[\log P(x|\theta)]}_{L_u(\theta)} \tag{3}$$

where $L_l(\theta)$ is the log-likelihood of the labeled data and $L_u(\theta)$ is the log-likelihood of unlabeled data. The first term in expression (3) represents the expected log-likelihood of a labeled instance and the second term shows the expected log-likelihood of an unlabeled instance (expectations are taken based on $P(x,y)$ and $P(x)$ respectively). Additionally, the random variable $\sqrt{N}(\hat{\theta_N} - \theta^*)$ is normally distributed with mean

zero and a fixed covariance matrix $C_\lambda(\theta^*)$. The above expression shows that, in the limit of large sample size, semi-supervised learning can be viewed as a convex combination of supervised and unsupervised learning, because its objective function is a convex combination of the objective functions of these two learning paradigms. In fact λ can be considered as the mixing parameter which combines information contained in labeled and unlabeled data.

We distinguish two cases: when the model is correct and when it is incorrect. The correct model means that the chosen parametric model $P(x, y|\theta)$ contains the true probabilistic model $P(x, y)$ which generated the data, namely there is a value for the parameters θ^{opt} where $P(x, y|\theta^{opt}) = P(x, y)$. Denote the maximum point of the likelihood functions of labeled and unlabeled data, namely $L_l(\theta)$ and $L_u(\theta)$, by $\hat{\theta}_l$ and $\hat{\theta}_u$ respectively. When the model is correct and identifiable, in the limit of large sample $N \to \infty$, we have $\hat{\theta}_l = \hat{\theta}_u = \theta^{opt}$. Since (in the limit) the maximum of the likelihood functions of labeled data $L_l(\theta)$ and unlabeled data $L_u(\theta)$ happens at θ^{opt}, the maximum of any convex combination of these two functions, in particular expression (3), also occurs at a set of points where θ^{opt} is in this set. At the beginning of this section we mentioned that (in the limit) semi-supervised learning tries to maximize expression (3) to find θ^*, so it must be the case that $\theta^* = \theta^{opt}$. In other words, semi-supervised learning finds the true model in this case.

It also shows that in this case the maximum likelihood estimator is consistent, i.e. it finds the parameters for producing the true model. Moreover, it is shown in [SL94] that reduction in the variance of θ^* would decrease the classification error. Increasing the sample size (by adding labeled or unlabeled data) would cause a reduction in the variance of the estimator. Consequently, adding unlabeled data would reduce the classification error.

When the model is not correct, in general we have $\hat{\theta_u} \neq \hat{\theta}_l$, and hence $R_{\hat{\theta_u}} \neq R_{\hat{\theta}_l}$, where $R_{\hat{\theta}_l}$ and $R_{\hat{\theta_u}}$ are classifier errors associated to Bayes decision rule applied to $P(x, y|\hat{\theta}_l)$ and $P(x, y|\hat{\theta}_u)$ respectively. As often is the case, suppose $R_{\hat{\theta_u}} \geq R_{\hat{\theta}_l}$. When we have a large number of labeled data or equivalently λ is close to one, the estimated

parameter $\hat{\theta}$ will be close to $\hat{\theta}_l$ because the objective function $L(\theta)$ is mostly affected by $L_l(\theta)$. As we increase the number of unlabeled data or equivalently decrease λ toward zero, the estimated parameter $\hat{\theta}$ moves toward $\hat{\theta_u}$. Intuitively, we start from classification error $R_{\hat{\theta}_l}$ and move toward $R_{\hat{\theta_u}}$. Therefore, it explains why unlabeled data can be harmful.

2.2.3 Finite data set behavior

For the finite data set, findings in [CCS+04] are based on empirical analysis. Experiments in [CCS+04] show that when the model is correct, adding unlabeled data reduces the error of the classifier. Moreover, as we increase the number of labeled data, adding unlabeled data cause a little reduction in the classifier error, because the classifier is already near the Bayes optimal error.

When the model is incorrect, adding unlabeled data can increase the estimation bias, in the same time it reduces the estimation variance. Classification error is a function of both estimation error and variance. Suppose the number of labeled data is small. The increase in the bias introduced by unlabeled data can be dominated by reduction in the variance, therefore, classification error can be reduced by increasing unlabeled data. However, if unlabeled data keep added, this large number of unlabeled data can increase the bias in such a way that it deteriorates the accuracy. This kind of instability in accuracy is further discussed in section 3.4.

In summary, statistical inference cannot be made without some assumptions. For the semi-supervised learning problem, if our assumption about the model is correct, then unlabeled data improves the performance. However if a wrong model is assumed, unlabeled data may degrade the performance.

3 Classifier based methods

This general approach starts by building an ensemble of classifiers from labeled data, and then tries to iteratively enhance them with the help of labeled and unlabeled

data. In each iteration, the classifiers collectively make some suggestions about the labels of unlabeled data, and then this information is used to update the classifier ensemble. This process is continued until a convergence condition is met.

3.1 The Yarowsky Algorithm and its variants

This algorithm starts by building a classifier from the labeled data, and then tries to iteratively enhance it. The performance measure iteratively being enhanced is the conditional log-likelihood. It uses unlabeled data as incomplete points, in the sense that their labels are missing. It can be shown that by maximizing the conditional likelihood the tendencies of the algorithm are twofold: first, labeling the unlabeled data, and second, choosing the true label for the labeled data. Sometimes conditional log-likelihood performance measure is replaced by a lower bound, and then that lower bound is maximized ([Yar95],[Abn04]). The generic Yarowsky algorithm is illustrated in Algorithm 1.

Algorithm 1 Yarowsky

1: Initially set pool of training data Λ to D_L
2: **repeat**
3: Train classifier h based on the current pool of training data Λ
4: reset pool of training data Λ to D_L
5: **for** each instance x in the unlabeled data **do**
6: Construct prediction distribution π_x for instance x where $\pi_x(c)$ shows the probability that classifier h predicts label c for x.
7: Set $\hat{c} = \arg\max_{c \in \mathcal{Y}} \pi_x(c)$
8: **if** $\pi_x(\hat{c}) > \zeta$ **then**
9: Add $(x, \hat{c})$ to Λ (do not remove x from unlabeled data)
10: **end if**
11: **end for**
12: **until** some condition is met

Let's look at the underlying probabilistic model. Suppose each instance x_i is represented by a feature vector $f_1^i, .., f_W^i$. We assume that each instance has W features, however, these W features can be different for different instances, denote the

set of (indices of) features for instance x as F_x. We assume each feature f_j individually can predict labels based on a probability distribution over the labels[4]: θ_{jc} shows the probability that feature j produces label c. It follows that $\forall j, \sum_{c \in \mathcal{Y}} \theta_{jc} = 1$. A collection of features can interact in two different ways (corresponding to two different probabilistic models) to generate a label for a point x:

- Choose the label randomly based on the prediction probability distribution π^x over the labels where $\pi^x_c = \frac{1}{W} \sum_{i \in F_x} \theta_{ic}$
- Choose the label randomly based on the prediction probability distribution π^x where $\pi^x_c \propto \max_{i \in F_x}(\theta_{ic})$

Note that the model parameters are θ_{ij}. They are learned from labeled and unlabeled data as the result of maximizing the following objective function:

$$L(\Phi, \theta) = \sum_{i=1}^{l+u} \sum_{c \in \mathcal{Y}} \Phi^{x_i}_c \log \pi^{x_i}_c \tag{4}$$

where Φ^x represents the (true) probability distribution over labels for a labeled or an unlabeled instance x. For labeled data $x \in \Lambda$, Φ^x has all of its mass on the corresponding label. For unlabeled instances $x \in D_u - \Lambda$, Φ^x is the uniform distribution over the labels (each label has the probability $\frac{1}{l}$).

Note that if there is no unlabeled data, then expression (4) would be the conditional log-likelihood of the labeled data, i.e. $L(\Phi, \theta) = \log P(Y_l | X_l, \theta)$. Another way to look at the objective function $L(\Phi, \theta)$ is to view it as the summation of negative cross-entropy $-L(\Phi, \theta) = \sum_{i=1}^{l+u} H(\Phi^{x_i} || \pi^{x_i})$ where negative cross-entropy[5] $H(p||q) = -\sum_i p_i \log q_i$. It is easily shown that $H(p||q) = H(p) + D(p||q)$ where $H(p)$ is the entropy of distribution p and $D(p||q)$ is the KL-divergence between the two probability distributions. For labeled instances, $H(\Phi^x)$ is zero since the probability distribution Φ^x is deterministic and does not have any randomness, so the

[4]It equivalently means that each feature corresponds to a different view to the instance. We will see again later learning methods in the framework of "multiple view" look to the instances.

[5]It shows the expected length of a symbol code sent when the coding is based on distribution q and selecting symbols is based on distribution p.

algorithm tries to minimize $D(\Phi^x||\pi^x)$, which equivalently means to make the prediction distribution π^x similar to the true distribution Φ^x. For unlabeled instances, $H(\Phi^x)$ is dominating since the uniform distribution has the highest entropy, so the algorithm tries to minimize it (to zero) by labeling the unlabeled data.

Sometimes the objective function $L(\Phi, \theta)$ is replaced by a lower bound, and then this lower bound is being maximized. Maximizing the lower bound may be easier and can be done analytically. For example, suppose we use the probabilistic model where the final predictive distribution is the average of the individual predictive distributions of each feature, then we have:

$$
\begin{aligned}
L(\theta, \Phi) &= -\sum_{x \in X_l \cup D_u} H(\Phi^x||\pi^x) \\
&= \sum_{x \in X_l \cup D_u} \sum_{c \in \mathcal{Y}} \Phi_{xc} \log \sum_{i \in F_x} \frac{1}{W} \theta_{ic} \\
&\geq \sum_{x \in X_l \cup D_u} \sum_{c \in \mathcal{Y}} \Phi_{xc} \sum_{i \in F_x} \frac{1}{W} \log \theta_{ic} \\
&= -\frac{1}{W} \sum_{x \in X_l \cup D_u, i \in F_x} H(\Phi^x||\theta_i) = K(\Phi, \theta)
\end{aligned}
$$

$K(\Phi, \theta)$ is the lower bound function which is maximized.

3.1.1 Discussion

It seems that there is a similarity in principle between transductive support vector machine (TSVM) [Joa99] and Yarowsky algorithm. Both methods try to maximize a performance measure under different labeling of the unlabeled data. In TSVM, the idea is to consider all possible labeling of the unlabeled data, and compute the maximal margin hyperplane for each case. Among all of the possible labeling, the one which has the highest maximum margin is chosen[6]. In a similar manner for the Yarowsky algorithm, all possible labelings of the unlabeled data is considered (in the Φ vector), and for each case the maximum likelihood estimation of the parameters

[6] Actually this problem is NP-Hard and often approximations of it are used.

is calculated. Then the model parameters corresponding to the highest maximum likelihood labeling is selected. This is a kind of transductive learning which is used to build an inductive learner: In the sense that not only the algorithm produces labels for unlabeled data but also it produces labels for new (unseen) instances by use of the constructed classifier.

3.2 EM, Co-training, Co-EM

3.2.1 EM

Perhaps Expectation Maximization (EM) is the oldest method to deal with incomplete data. In the setting of semi-supervised learning problem, missing data is the latent labels of unlabeled data. We assume a joint probability distribution over the space of input instances and their labels $P(x, y)$. Then we choose a parametric model $P(x, y|\theta)$ for modeling the input-output joint probability distribution and try to find the best $\hat{\theta}$ by maximizing the incomplete log likelihood of the sample:

$$L(\theta) = \sum_{i=1}^{l} \log P(x_i, y_i|\theta) + \sum_{j=l+1}^{l+u} \log P(x_j|\theta)$$

$$\hat{\theta} = \arg\max_{\theta} L(\theta)$$

where marginal distribution $P(x|\theta) = \sum_{c \in \mathcal{Y}} P(x, y = c_i|\theta)$.

The idea in EM is to optimize the incomplete log likelihood of the observed data $\mathcal{L}(\theta)$ via the iteration of the two phases in Algorithm 2.

Any optimization algorithm can be used to optimize the incomplete log likelihood but EM is widely used because it has an intuitive interpretation. In the first step it makes *inference* about the (distribution over) latent labels, and in the second step, it maximizes the parameters of the model based on the log likelihood of the current *complete* data.

As noted in [See00] doing EM on $P(x, y)$ might be dangerous. EM tries to find the model which best fits $P(x, y)$. Since the fitness measure is not based on the

Algorithm 2 EM

1. **Expectation** step: Given the current model parameters and observed data, compute the probability distribution over the unobserved data $P(y_{l+1}, ..., y_{l+u}|D_l, D_u, \hat{\theta})$.
2. **Maximization** step: Search for the optimum model parameters which maximizes the log likelihood of the data $\hat{\theta} = \arg\max_\theta \sum_{i=1}^{l} \log P(x_i, y_i|\theta) + \sum_{l+1}^{l+u} E(\log P(x_j, y_j|\theta))$.

discriminative ability (or power) of x in predicting y, i.e. $P(y|x)$, the selected model might not be a good predictor of class labels. Furthermore, the log-likelihood function has many local optima and it is highly probable that EM gets stuck in a local optimum.

3.2.2 Co-training

Co-training [BM98] is one of the first algorithms proposed for semi-supervised learning, and analyzed thereafter [BBY04, DML01]. Co-training assumes there are two sets of features which are conditionally independent given the labels, and each of which is enough to build a good classifier[7]. Each feature set is called a *view*, and Co-training learns two classifiers where each of them corresponds to a single view.

Co-training is illustrated in Algorithm 3. In each iteration, algorithm selects a subset of unlabeled data (N_p instances in D_u which are labeled positive, and N_n instances in D_u which are labeled negative) whose labels are assigned with high confidence by the current classifiers, and add them to the pool of labeled training data. The number of selected positive and negative instances is proportional to their ratio in the labeled sample. Then the classifiers are retrained based on this expanded training data. This process continues till it converges.

The conditional independence assumption means that if we label an unlabeled instance in one view, the resulting labeled instance is a random instance in the other

[7]This method is naturally suited in domains where the information about each instance comes from two separate sources or sensors.

Algorithm 3 Co-training

1: Initially, train classifiers h^1 and h^2 on labeled data D_l.
2: **repeat**
3: **for** each view w=1..2 **do**
4: Remove N_p elements with greatest $h^w(x_u)$ from D_u and add $(x_u, +1)$ to D_l.
5: **end for**
6: **for** each view w=1..2 **do**
7: Remove N_n elements with smallest $h^w(x_u)$ from D_u and add $(x_u, -1)$ to D_l.
8: **end for**
9: Retrain h^1 and h^2 using the updated D_l
10: **until** D_u becomes empty

view. As a result each view provides random labeled instances for the opposite view with some slight *noise*. As the result, We can use any machinery for building classifiers which works based on the (weak) i.i.d. assumption in collecting training instances. Furthermore since the views are independent, it reduces the chance that both classifiers (with high confidence) label an unlabeled instance *wrongly* at the same time.

3.2.3 Co-EM

Co-EM is a variant of Co-training and EM which is introduced in [NG00]. It can be seen as a probabilistic version of Co-training (Algorithm 4).

Algorithm 4 Co-EM

1: Initially, train f^2 on labeled data.
2: **repeat**
3: **for** each view w=1..2 **do**
4: Estimate the class probabilities of unlabeled data D_u based on the *other* classifier $f^{\bar{w}}$.
5: Retrain f^w based on the labeled data D_l and probabilistically labeled data D_u.
6: **end for**
7: **until** some condition is met

As we can see, this algorithm is very similar to the EM. In the main loop we consider each classifier f^w (corresponding to each view), and retrain it based on the

labeled data and probabilistically labeled data D_u, labeled by the classifier corresponding to the other view. To this end, the first step can be considered as **E** step which is performed by the classifier of the other view, and the second step as the **M** step on the classifier in the current view.

3.3 Agreement Maximization among Classifiers

In co-training, each learner (corresponding to each view) gradually labels *some* unlabeled data which is mostly confident about their labels, add them to the pool of training data, and then use them in the next iteration to *inform* the other learner about its *opinion*. In other words, unlabeled data is a platform for the two learners to communicate their opinions. After a while, as a side effect of the algorithm, two learners are motivated to *agree* on unlabeled data. What if we make the agreement of the learners as the explicit goal of the algorithm? Does this help to learn from unlabeled data? In [Les05, CS99], these questions are answered.

It is proven [Les05] that reducing the disagreement, or equivalently maximizing the agreement, helps the process of learning from labeled and unlabeled data[8]. Unlike Co-training which makes commitment about the labels of unlabeled data, here we do not make decision about the label of unlabeled data. To make the idea clear, suppose we have different views of the data, and each view is enough to learn the target concept. For each view we have a concept space, denote the concept corresponding to the target concept in each view as c^w_{opt}. It is clear that all c^w_{opt} must agree on the label of unlabeled data because essentially they are different representations of the (same) target concept. It causes the size reduction of the hypothesis space in each view. Hence, learning the target concept in this reduced hypothesis space becomes easier.

[8]In the framework of Probably Approximately Correct (PAC) learning.

3.3.1 The Agreement Boost Algorithm

This approach to the semi-supervised learning is brought to an algorithm by using the Boosting framework (Algorithm 5). At first classifiers are initialized in each view to a simple one. In each iteration of the main loop of the algorithm, each unlabeled instance is re-weighted based on the disagreement of the current classifiers (each one corresponds to a different view); the bigger the disagreement, the bigger the assigned weight[9]. Then, each learner is retrained based on the new weights of unlabeled instances and *pseudo-labels* assigned to them by weighted voting among classifiers. Then for each view, the new classifier would be a combination of the old classifiers and the new learned one, and the main loop iterates till it converges. Finally, the best classifier among the views is selected as the output of the algorithm.

Algorithm 5 Agreement Boost

1. initialize $h^w \equiv 0$ for each view
2. Iterate until done
   - For each view w do the followings:
     - **a.** Set the weights for labeled data based on the classification error and for unlabeled data based on variance of the disagreement.
     - **b.** Set the pseudo-label for each unlabeled instance based on the weighted votes of classifiers.
     - **c.** Learn classifier f^w based on the weights
     - **d.** Find α^w which minimizes $F(h^1, .., h^w + \alpha^w f^w, .., h^T)$
     - **e.** Set $h^w = h^w + \alpha^w f^w$
3. Output classifier $sign(h^w)$ which has the minimum error on the sample

In the Agreement Boost algorithm, f^w and h^w are binary classifiers for each of the W views. Let us look at $F(h^1, ..., h^W)$ which represents the cost function[10] of the

[9] In this way, *controversial* instances are located.

[10] Note that we can consider the negative of cost function $-F(h^1, ..., h^W)$ as the performance measure.

main algorithm:

$$F(h^1, ..., h^W) \triangleq \sum_{w=1}^{W} \frac{1}{W} \sum_{i=1}^{l} er(-y_i h^w(x_i)) + \eta \sum_{j=l+1}^{l+u} er(V(x_j)) \tag{5}$$

where $er : R \rightarrow R$ is some convex and strictly increasing function, and $V(x_j)$ is a measure of disagreement of the classifiers on an unlabeled point x_j. For each labeled instance (x_i, y_i), the quantity $y_i h^w(x_i)$ can be considered as the margin of this labeled instance w.r.t the classifier h^w. For an unlabeled point x_j, the intuition of $V(x_j) = Var(r_1, ..., r_W)$ is the variance in the prediction vector, where the prediction vector is $(r_1, ..., r_W) = (h^1(x_j), ..., h^W(x_j))$ and:

$$Var(r_1, ..., r_W) = \frac{1}{W} \sum_{w=1}^{W} r_w^2 - \Big(\frac{1}{W} \sum_{w=1}^{W} r_w \Big)^2$$

What happens when the cost function (5) is minimized? The cost function consists two parts: the first part encourages large margin for each classifier while the second part encourages agreement of the classifiers on unlabeled data. The effect of labeled and unlabeled data in the learning process is controlled by parameter $\eta \in \mathbb{R}^+$. Bigger values for η causes the agreement constraint to be imposed more strongly.

3.4 Stable Mixing of Complete and Incomplete Data

As we saw before, EM tries to maximize the incomplete log-likelihood of data. Equivalently, it can be shown that EM tries to find, in a given family of models $\mathcal{M}$, a probability distribution which is closest (in the sense of KL-Divergence) to the empirical probability distribution:

$$P^*(x, y) = \arg \min_{Q \in \mathcal{M}} (1 - \lambda) D(P^l(x, y) || Q(x, y)) + \lambda D(P^u(x) || Q(x))$$

where $\lambda = \frac{u}{l+u}$ is the fraction of unlabeled data, $P^l(x, y)$ is the empirical distribution of labeled data, and $P^u(x)$ is the empirical marginal distribution of unlabeled data. As we increase the number of unlabeled data (λ grows to one), the role of labeled data

is decreased in the above objective function. As a result we may loose information contained in labeled data because the solution is largely affected by unlabeled data.

For a given value of λ, we can solve the optimization problem by expectation maximization. If we consider the E and M steps together, each iteration of EM tries to update the parameters of the proposed model to eventually reach a fixed point. Hence it can be viewed as an operator on probability distributions [CJ01]; for a fixed λ, we call this operator EM_λ. The fixed point of this operator satisfies $Q_\lambda = EM_\lambda(Q_\lambda)$ which is a function of λ.

In stable mixing of complete and incomplete data, we vary λ from zero to one continuously and trace the path of solutions, i.e. fix points of the EM_λ operator. When λ is zero, the solution corresponds to the fully labeled data. As we increase the λ, we inject the information from unlabeled data to our estimation. However, we may reach a value for λ where the solution does not exist (discontinuity in the path) or it is not unique (bifurcation). We call such a point a *critical point*. Increasing λ beyond a critical point and doing EM may change the solution drastically and put it in a completely different part of the solution space where the estimation can no longer be tied to the (few) labeled data. Hence setting λ to the first critical point, causes maximal advantage of unlabeled data while it maintains information from labeled data.

Instead of finding the fixed point of EM_λ for each $\lambda \in [0, 1]$, the differential equation which characterizes the continuous path of solutions (fixed points) is solved:

$$\frac{dQ}{d\lambda} = \frac{\partial EM_\lambda}{\partial \lambda}(Q)\frac{d\lambda}{d\lambda} + \frac{\partial EM_\lambda(Q)}{\partial Q}\frac{dQ}{d\lambda}$$

which yields:

$$\frac{dQ}{d\lambda} = (I - \nabla_Q EM_\lambda(Q))^{-1}\frac{\partial EM_\lambda}{\partial \lambda}(Q).$$

Any initial fixed point can be traced in a unique continuous path until the (transferred) Jacobian $I - \nabla_Q EM_\lambda(Q)$ becomes singular [CJ01], which is the location of the first critical point. Hence, the solution breaks down near the first critical point and we are unable to trace the path beyond that point. However, sometimes we want

to trace the path until $\lambda = 1$. To eliminate this difficulty, the path can be traced in the joint space of (Q, λ) [CJ02]. By doing this, discontinuities are resolved but bifurcations still exist; fortunately, the number of bifurcations in critical points is of measure zero.

For example if the probability distribution Q belongs to a parametric class with the parameter vector η, the path is sought in the joint space of (η, λ). Let t be the parameter of the path, then based on the identity $\eta = EM_\lambda(\eta)$ the path is characterized by the following differentiation equation:

$$\frac{d\eta}{dt} = \frac{\partial EM_\lambda(\eta)}{\partial \lambda} \cdot \frac{d\lambda}{dt} + \frac{\partial EM_\lambda(\eta)}{\partial \eta} \cdot \frac{d\eta}{dt}$$

which yields:

$$\left[\begin{array}{cc} \nabla_\eta EM_\lambda(\eta) - I & \frac{\partial EM_\lambda(\eta)}{\partial \lambda} \end{array} \right] \cdot \left[\begin{array}{c} \frac{d\eta}{dt} \\ \frac{d\lambda}{dt} \end{array} \right] = 0$$

Ordinary numerical methods, such as Runge-Kutta, can be used to solve the above differential equation with a given initial condition, i.e. the point corresponding to the solution of complete data.

3.4.1 Cross Validation

Related to this idea, is the method used in [NMTM00] where cross validation is used to determine the value of the mixing parameter. They trained a naive Bayes classifier for the document classification task: some documents are given, and we are asked what is the topic (class label) of each of them. In the context of generative model based approaches, the joint distribution of the input (documents) and class labels is modeled by a mixture model (each mixture component is responsible for producing the documents of a particular class). As we have seen before, when the prior assumptions about the model are not correct, unlabeled data may hurt classification accuracy. To prevent the degradation of accuracy, authors in [NMTM00] suggest to control the effect of unlabeled data during the estimation of model parameters. More precisely, the parameters are found based on the following optimization problem using EM

algorithm:

$$\theta^{opt} = \arg\max_{\theta \in \Theta} \sum_{i=1}^{l} \log P(x_i, y_i|\theta) + \lambda \sum_{j=l+1}^{l+u} \log \sum_{y \in \mathcal{Y}} P(x_j, y|\theta)$$

where Θ is the search space of θ, and $\lambda \in [0, 1]$ controls the influence of unlabeled data in estimating the model parameters. If $\lambda = 1$ then the effect of an unlabeled data point is the same as the effect of a labeled data point, and if $\lambda = 0$ unlabeled data does not have any effect in estimating the model parameters. The best value for λ is found by one-leave-out cross validation in contrast to the previous section where the best value for λ is specified by finding the location of the first critical point.

4 Data based methods

In the general setting of the learning problem, we are given some data and we are asked to produce a *good* classifier, the one which not only behaves well on the given data but also has a low classification error on the unseen data. What is a good behavior for a function w.r.t the given data? In the supervised learning problem, good behavior means having low classification error on the given training sample. However in the semi-supervised learning problem we are given unlabeled data as well as labeled data: not only the classification error on the training sample has to be low but also the function must be compatible with the input distribution by inspecting its values on unlabeled points.

Basically, unlabeled data can be used to find how data is distributed in the feature space, i.e. it gives us information about the (marginal) probability distribution $P(x)$. So if we use the knowledge of the marginal $P(x)$ in assessing the prediction power of the classifiers, we have (indirectly) used information contained in unlabeled data to find a good classifier.

4.1 Background

We use regularization framework to express the ideas in this section. In this framework, there exists a function class $\mathcal{H}$ which includes all possible functions (each function corresponds to a classifier). The following optimization problem is typically posed to find the best classifier:

$$\arg\min_{f\in\mathcal{H}} \frac{1}{l}\sum_{i=1}^{l} loss(f(x_i), y_i, x_i) + \lambda\Omega(f)$$

where l is the number of labeled instances, λ is a tradeoff (regularization) parameter, and Ω is called the regularization term which shows the complexity of the function. The reason for posing the above objective function is that we are looking for a classifier which has the lowest (true) expected risk, but it is not possible to find the true risk of a function f because the input distribution $P(x)$ is unknown. Instead, the true risk is upperbounded by an expression composed of empirical risk and complexity terms (which is exactly the above objective function), and then this expression is minimized. Based on our belief and prior knowledge about the problem, we can hopefully design good upperbound expressions (or objective functions). For semi-supervised learning, the natural idea is to use unlabeled data for a better estimation of the complexity term $\Omega(f)$, so the complexity term depends on input sample among other factors.

The space of possible functions is usually the reproducing kernel Hilbert space $\mathcal{H}_k$ corresponding to a Mercer kernel[11] $k(.,.)$, although there are cases where it is the space of (conditional) probability distributions. Simply speaking, the reproducing kernel Hilbert space $\mathcal{H}_k$ is constructed by considering all linear combinations of the functions $\{k(.,x)\}_{x\in\mathcal{X}}$, i.e. $\mathcal{H}_k = \{f(.)|f(y) = \sum_{i=1}^{m}\alpha_i k(y,x_i)\}$ for arbitrary $\{x_i \in \mathcal{X}\}_{i=1}^{m}$ ([Her01]). The norm of a function f in this space is defined to be $\|f\|_k = [\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j k(x_i,x_j)]^{\frac{1}{2}}$. The inner product of two functions $k(x,.)$ and $k(x',.)$ in this space is $k(x,x')$. The reproducing property comes from this fact that for every function f in this space, the inner product of f with the function $k(a,.)$ in this space equals $f(a)$, in other words $\langle f(.), k(a,.)\rangle_{\mathcal{H}_k} = f(a)$.

[11] It means that the kernel $k(.,.)$ is positive semidefinite, i.e. $\forall g, \int_{\mathcal{X}}\int_{\mathcal{X}} k(x,y)g(x)g(y)\,dx\,dy \geq 0$.

The complexity term Ω depends often on the norm of a function f: the bigger norm corresponds to the more complexity, so the optimization problem becomes as follows:

$$\arg\min_{f\in\mathcal{H}_k} \frac{1}{l}\sum_{i=1}^{l} loss(f(x_i), y_i, x_i) + \lambda\Omega(\|f\|_k) \tag{6}$$

where Ω is an increasing function of the norm. The Representer theorem ([Her01]) guarantees that the best function in the optimization problem (6) admits this form: $f(y) = \sum_{i=1}^{l} \beta_i k(y, x_i)$, in other words it can be written as a linear combination of the kernel function at the (limited number of) training points. As a result, all we have to do to find the best function f, is to plug in its parametric form in the objective function (6) and search for the weight vector $(\beta_1, ..., \beta_l)$. Standard optimization techniques can be used to solve the transformed optimization problem. The power of Representer theorem is that it reduces the search for the best function from an infinite dimensional hypothesis space $\mathcal{H}_k$ to the search for the weight vector in a finite dimensional space.

4.2 Measure based Regularization

Many learning algorithms require the classifier to be an smooth function on the instance space $\mathcal{X}$ by means of controlling its complexity, i.e. more smooth functions have less complexity. Smoothness requirement comes from a fundamental assumption that *two close points in the instance space should have the same label*. However this assumption is a weak one, in other words, for some real world learning problems we can expect a stronger requirement on the classification function. For example, often data is not scattered uniformly throughout the whole space, it is distributed in the form of dense regions (or clusters). So, it is natural to make this stronger assumption that *two points which are connected by a path going trough dense regions should have the same label* (cluster assumption). Unlabeled data can be used to find the clusters in the input space, and then these clusters can be used to characterize good (or smooth) functions (see figure 1). This idea can be implemented using regularization framework in different ways. In [BCH04] three ways are mentioned to implement

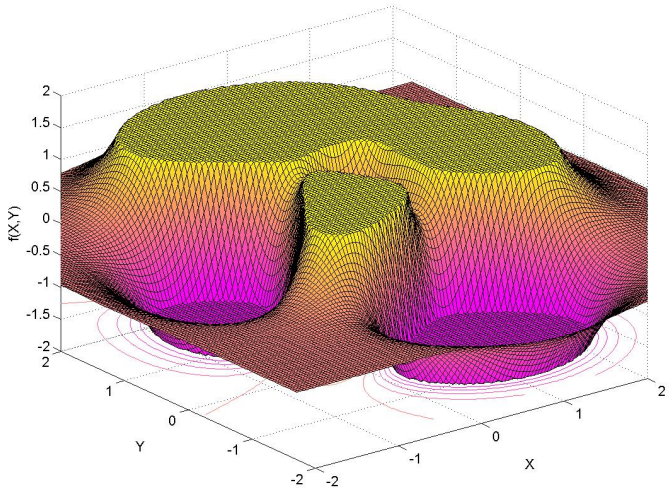

Figure 1: Example of a smooth function on the 2D input domain. There are four clusters: two of them belong to one class and the other two belong to the other class. The value of function within a cluster must be roughly the same, and in this case it is constant. The function is allowed to fluctuate more where the density of data is low, i.e. outside the clusters. Note that the decision *boundary* will be put outside the clusters.

cluster assumption.

The first method is to incorporate marginal distribution $p(x)$ into the smoothness term. Most learning algorithms impose smoothness constraint on the learned function by requiring it to have a bounded gradient, i.e. the regularization term is $\Omega(f) = \sup_x \|\bigtriangledown f(x)\|$. A natural way to implement the cluster assumption is to penalize variation of the function more in the dense regions, in other words changing the regularization term to $\Omega(f) = \sup_x \|p(x) \bigtriangledown f(x)\|$. We can generalize this idea by considering other smoothness functionals $\mathcal{L}(.)$ rather than gradient, and other increasing functions $\Psi(.)$ of $p(x)$ rather than identity function. Hence, the general form of the regularization term becomes the following:

$$\Omega(f) = \|\mathcal{L}(f)\Psi(p)\| \tag{7}$$

The second approach is to look at the problem from the geometric viewpoint. The

idea is to change the metric of the instance space $\mathcal{X}$ based on the marginal distribution $p(x)$ ([VB03], [CZ05]). When the density $p(x)$ is high, it means that we can locally *stretch* the space. Conversely, when the density is low, it means that the space can be *blown up* locally. In this changed space with its metric, we can use the original smoothness assumption to infer the best classifier.

The third approach is to find a good representation of points that captures well the clusters. The general idea is to construct an adjacency graph for labeled and unlabeled points whose weights are given by adjacency matrix W. Then input points can be well represented by first eigen vectors of a modified version of matrix W, and the resulting weighted graph[12] is an approximation to the (possible) low dimensional manifold on which data is sitting, and captures its clusters. Now any function has to be smooth in terms of the neighborhood structure of this weighted graph.

4.2.1 Solving the Regularized Optimization Problem

Generally, solving the optimization problem having a regularization in the form of (7) is not an easy task. The reason is the following. Suppose we could find a reproducing kernel Hilbert space $\mathcal{H}_{k'}$ and its associated kernel $k'(.,.)$ which satisfy the following:

$$\Omega(f) = \|\mathcal{L}(f)\Psi(p)\| = \Big(\langle f, f\rangle_{\mathcal{H}_{k'}}\Big)^{\frac{1}{2}} = \|f\|_{k'}$$

$$f(x) = \langle f(.), k'(x,.)\rangle_{\mathcal{H}_{k'}}$$

We can solve the optimization problem by using the Representer theorem and writing f as a weighted combination of kernel $k'(.,.)$ at labeled points. However, except for some special cases of smoothness functional $\mathcal{L}(.)$ and density $p(.)$, finding the Hilbert space $\mathcal{H}_{k'}$ (with kernel $k'(.,.)$ satisfying the above conditions) is not an easy task since it amounts to solve difficult differential equations for finding the kernel. Hence, the idea in [BCH04] is to write f in terms of a linear combination of some basis functions

[12] Actually it is the adjacency matrix of the *image* of original points in the space where its axes are the eigen vectors of the original adjacency matrix.

$\phi_i(.)$:

$$f(x) = \sum_{i=1}^{m} \alpha_i \phi_i(x) + b$$

If we consider a regularization term like (7), after substitution we will have:

$$\Omega(f) = \int \bigtriangledown f(x) \cdot \bigtriangledown f(x) p(x)\, dx = \sum_{i,j=1}^{m} \alpha_i \alpha_j \int \bigtriangledown \phi_i(x) \cdot \bigtriangledown \phi_j(x) p(x)\, dx$$

Finding the integrals $\int \bigtriangledown \phi_i(x) \cdot \bigtriangledown \phi_j(x) p(x)\, dx$ in the above expression could be difficult. So even for this special case, solving the problem analytically could be intractable because the integrals depend on $p(x)$ and how we approximate it. As stated in [BCH04], unlabeled data can be used to estimate the density function $p(x)$ and to select good basis functions $\phi(x)$; for example we may put a gaussian RBF $\exp(-\frac{\|x-x_j\|^2}{2\sigma})$ on each unlabeled point x_j to approximate $p(x)$.

4.3 Manifold Regularization

At the conceptual level, this approach fits in the third method of the section 4.2. Recall from section 4.1 that a good classification function in a reproducing kernel Hilbert space $\mathcal{H}_k$ must have a small norm. Moreover consider the case where input data is not scattered in the whole input space but in a compact submanifold $\mathcal{M}$ of it: the best classification function must be smooth w.r.t. this submanifold. For example, input space can be the three dimensional Euclidean space $\mathbb{R}^3$ but data may be distributed on a sphere which is a two dimensional submanifold of the whole space. In this case the best function has to be smooth on this sphere in addition to the requirement that it has to have a small norm in the $\mathcal{H}_k$. To achieve these constraints, the optimization problem is as follows:

$$\arg\min_{f \in \mathcal{H}_k} \sum_{i=1}^{l} loss(f(x_i), y_i, x_i) + \underbrace{\lambda_k \|f\|_k + \lambda_I \|f\|_I}_{\Omega(f)}$$

where λ_k and λ_I are tradeoff parameters. Let submanifold $\mathcal{M}$ to be the support of the input density function $p(x)$. A natural choice for $\|f\|_I$ is $\int_{\mathcal{M}} \langle \bigtriangledown_{\mathcal{M}} f, \bigtriangledown_{\mathcal{M}} f \rangle$ which essentially accounts for the gradient of the function on the data manifold ([BNS05]).

The problem with the loss expression is that we do not have explicitly $p(x)$ to find $\mathcal{M}$ and then calculate $\int_{\mathcal{M}} \langle \bigtriangledown_{\mathcal{M}} f, \bigtriangledown_{\mathcal{M}} f \rangle$, so it has to be approximated from labeled and unlabeled data. Construct an adjacency graph for labeled and unlabeled points whose weights are given by adjacency matrix W. The loss term $\|f\|_I$ can be approximated by the following expression:

$$\frac{1}{2} \sum_{i=1}^{l+u} \sum_{j=1}^{l+u} (f(x_i) - f(x_j))^2 W_{ij} = [f(x_1)...f(x_{l+u})] \mathcal{L} [f(x_1)...f(x_{l+u})]^T$$

where the combinatorial graph laplacian $\mathcal{L} = D - W$, and D is a diagonal matrix which has the sum of each row of W as its main diagonal elements. Based on a modified version of the Representer theorem ([BNS04]), the form of the solution will be the following:

$$f(y) = \sum_{i=1}^{l+u} \alpha_i k(x_i, y).$$

Notice that in this case the best function is a linear combination of the functions $k(x, .)$ at labeled and unlabeled points $\{x_i\}_{i=1}^{l+u}$.

In [SNB05] this work is extended to multiple view setting where we have two views to the instance space $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$. Each view has its own kernel, and its own geometry on the instance space. Data might be sitting on different manifolds, each of which corresponding to a different view. The true manifold structure (which can be modeled as a weighted graph) may be obtained by merging these different manifold representations appropriately. Therefore, we may combine the regularization operators $\mathcal{L}_1$ and $\mathcal{L}_2$ of these different views to achieve a regularizer in the new geometrical space: $\mathcal{L} = \alpha \mathcal{L}_1 + (1 - \alpha) \mathcal{L}_2$. The regulizer $\mathcal{L}$ is used in the penalty term $\|f^w\|_I$ of the optimization problem of each view.

4.4 Information Regularization

The derivation of this method is first presented in [SJ02], and then extended in [CJ03] where learning theoretical bounds are also given. This approach is another attempt

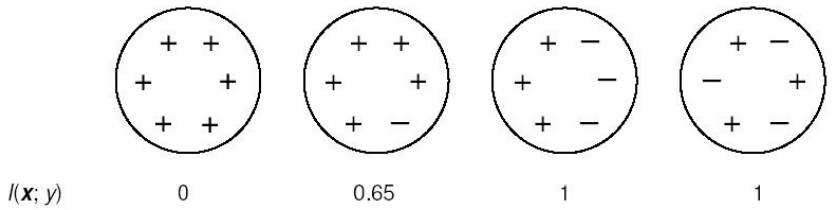

Figure 2: Examples of small regions and the mutual information $I_Q(x, y)$ associated to them [SJ02]. Data is uniformly distributed within each region.

to put the decision boundary in regions where data is not distributed densely, i.e. connected dense regions (or clusters) must have the same label. In other words, in a local area of these dense regions label does not vary so much or is *independent* of the instance. Mutual information is used to quantify the level of independency between x and y in the local regions. So, we are looking for a conditional probability distribution $p(y|x)$ which induces the least possible mutual information in dense regions, and at the same time, is constrained to produce correct label for labeled data[13].

Consider a small region Q. The expected mutual information between x and y in this region I_Q is:

$$I_Q(x, y) = \sum_y \int_{x \in Q} p_Q(x, y) \log \frac{p_Q(x, y)}{p_Q(x) p_Q(y)} \, dx \tag{8}$$

where p_Q is the probability distribution restricted to the local region Q. $I_Q(x, y)$ is zero if x and y are independent (see figure 2), and its value is not sensitive to the perturbation of the labels within the local region.

Mutual information shows the average information in a region, and we are interested in this quantity for each *point* in the region, so it is weighted by the density $p(x)$ which we have assumed can be approximated in the presence of having large unlabeled data. After considering some technicalities and doing some work, the objective

[13]Note that the resulting discriminative classifier is $\hat{y} = \arg\max_y p(y|\hat{x})$

function for maximization is:

$$p_{opt}(y|x) = \max_{p(y|x)} \sum_{i=1}^{L} \log p(y_i|x_i) - \lambda \int_{\mathcal{X}} p(x) Tr[F(x)]\, dx \tag{9}$$

where $Tr(F)$ is the trace of the matrix F (the sum of elements in the main diagonal), $F(x)$ is the fisher information $F(x) = E_{p(y|x)}[\bigtriangledown_x \log p(y|x) \cdot \bigtriangledown_x \log p(y|x)^T]$, and λ is a tradeoff parameter. The first term in (9) is the log-likelihood which causes fitness to the labeled data, and the second term is a penalty representing the complexity.

If we do not assume any parametric form for $p(y|x)$, then calculus of variation can be used to derive a differential equation which characterizes the solution of the optimization problem (9). However, if we assume a parametric model $p(y|x;\theta)$, then the optimization becomes easier and the solution can be obtained by maximizing over the parameters:

$$p(y|x;\theta^{opt}) = \max_{\theta} \sum_{i=1}^{L} \log p(y_i|x_i;\theta) - \lambda \int_{\mathcal{X}} p(x) Tr[F(x;\theta)]\, dx$$

Gradient ascent or Newton's method can be used to solve the above optimization problem.

4.4.1 Distributed Information Regularization

In [CJ04] information regularization principle is casted as a communication problem. Consider a set of regions[14] $\mathcal{Q} = \{Q_1, .., Q_m\}$ where each region $Q \subseteq X_{L \cup U}$ has a prior probability $p(Q)$. For any fixed $p(y|x)$ the communication problem is defined as follows. Sender randomly selects a point x by choosing a region Q based on $p(Q)$ and then a point x based on $p(x|Q)$. To send the label of the point x, sender uses the coding scheme specific to the region Q, where optimal coding scheme for this region has been designed based on $p(y|Q)$.

Receiver has access to the selected region Q (and its specific coding scheme) and point x; its task is to decode the received information to find the label. It is clear that

[14]Each region captures the similarity among some points, i.e. a region is a set of similar points based on some *notion of similarity*.

to minimize the communicated bits (or information), $p(y|Q)$ has to put its total mass into one label (in this case the number of bits needed to send is zero), or has to be close to such probability distribution. In other words, minimizing the communicated bits encourages to assign one label to (similar) points which are included in one region. We also have to take into account the number of bits to encode the label of labeled data, so the regularization problem is as follows:

$$p_{opt}(y|x) = \arg\min_{P(y|x)} -\sum_{i=1}^{L} \hat{p}(x_i, y_i) \log p(y_i|x_i) + \lambda \sum_{Q \in \mathcal{Q}} p(Q) I_Q(x, y) \qquad (10)$$

where $\hat{p}(x, y)$ is the estimated empirical distribution, $I_Q(x, y)$ is defined in (8), and λ is a tradeoff parameter. The first term in (10) motivates $p(y|x)$ to fit properly on the labeled data, and the second term forces it to assign similar labels to points in each region. In [CJ04], a distributed propagation algorithm is given to solve the optimization problem (10). This work extends the method to situations where we have multiple view to the objects, and where $p(y|x)$ can be represented by a tree structured graphical model.

4.4.2 Entropy Regularization

Suppose the decision boundary for classification does not cut dense regions. So given a dense region, the distribution over the labels is very close to the delta function on one of the labels (which is the label of that region). In other words, the entropy of the conditional $p(y|x)$ is very low (it is zero when $p(y|x)$ is a delta function). Therefore, we are looking for a conditional probability distribution $p(y|x; \theta^{opt})$ which correctly predicts the observed labels for labeled data and in the same time has a low conditional entropy:

$$\theta^{opt} = \arg\max_{\theta} \sum_{i=1}^{l} \log p(y_i|x_i; \theta) - \lambda H_{emp}(p(y|x; \theta))$$

where λ is a trade of parameter to control the effect of labeled and unlabeled data in the objective function, the first term is the (conditional) likelihood of labeled data,

and $H_{emp}(p(y|x;\theta))$ is the empirical conditional entropy based on unlabeled data [GB04]:

$$H_{emp}(p(y|x;\theta)) = \frac{1}{u}\sum_{i=l+1}^{l+u}\sum_{y\in\mathcal{Y}} p(y|x_i;\theta)\log p(y|x_i;\theta)$$

There is also a Bayesian interpretation of this method based on the above objective function. In a parametric model of probability distributions with the parameter vector θ, we encode our prior belief about assignments to θ based on the (empirical) conditional entropy of the resulting distribution $p(\theta) \propto \exp[-\lambda H(p(y|x;\theta))]$. This prior is the most unbiased distribution given that $E_\theta[H(p(y|x;\theta))] = Const(\lambda)$ where $Const(\lambda)$ is a constant related to λ and shows the (allowed) expected conditional entropy.

4.5 Harmonic Mixtures

This method is an extension to Harmonic functions for transductive learning [ZGL03] for enabling it to deal with unseen data points. The idea of Harmonic mixtures [ZL05] is to extract clusters from (labeled and unlabeled) data, and consider them as *supernodes*. Having a weighted graph on these supernodes representing their neighborhood structure (geometry), we assign label to them by doing random walk and propagating labels among (super)nodes.

More specifically, assume that we have only two classes: **0** and **1**. Suppose we have a weighted graph where vertices are supernodes v_i and edge weights w_{ij} show the similarity of the connected pair of vertices v_i and v_j. As we saw in manifold regularization, we can assign soft label f_i (namely a number between zero and one) to each vertex v_i based on this intuition that labels must vary smoothly on this graph. If we use the combinatorial laplacian operator $\mathcal{L}$ to impose the smoothness, we are interested to minimize the following energy function subject to this constraint that for the labeled points (x_i, y_i) we must have $f_i = y_i$:

$$\varepsilon(f) = f^T\mathcal{L}f = \sum_{ij=1}^{l+u} w_{ij}(f_i - f_j)^2$$

where $f \in [0,1]^{l+u}$ is the probability vector for all of the vertices. Note that this objective function is very similar to that of manifold regularization, but here we are interested in the value of the function only on the available vertices. We can interpret this method in the statistical framework as follows: each assignment of the (soft) labels to the vertices corresponds to an energy for that particular configuration of the graph. A probability value is assigned to each configuration of the graph based on its energy $p(f) \propto e^{-\varepsilon(f)}$, so the space of possible configurations defines a gaussian field. By looking for the minimum energy configuration, we are after the peak (or mean) of this gaussian field. We can also interpret this method in the random walk framework: from each vertex we are doing a random walk based on the weights of the graph, and assign (soft) label f_i to vertex v_i which is the probability of reaching a vertex labeled **1** (labeled points are absorbing boundary). In either case of random walk, gaussian field, or smoothness interpretations the solution must satisfy this equation $\mathcal{L}f = 0$, which characterizes it as a *harmonic* function.

In Harmonic mixtures, a mixture of gaussians (in general, a generative model) is trained to model the joint distribution $p(x, y)$ for the purpose of finding clusters. At the same time labels are propagated from labeled data to unlabeled data so that to exploit the (possible) manifold structure of the mixture components. The objective function to be minimized is:

$$-\lambda L(\theta) + (1-\lambda)\varepsilon(\theta)$$

where θ is the parameter vector of our generative model, $L(\theta)$ is the incomplete log-likelihood, and $\varepsilon(\theta)$ is the minimum graph energy corresponds to current model parameters θ. Minimizing the above objective function is equivalent to maximizing the log-likelihood and minimizing the graph energy. As a tradeoff parameter, λ controls the effect of the generative based (first) term and discriminative based (second) term in the objective function. Note that the first and second terms are not independent. Maximizing the data likelihood specifies the graph structure, and the graph structure determines the minimum possible energy for this particular structure.

4.6 Learning Predictive Structures from Multiple Tasks

As we have seen, the main idea of the data based method is to restrict the class of functions considered by requiring them to be smooth w.r.t a similarity measure in the input space $\mathcal{X}$. One way to achieve this is to first define a distance measure in the input space, and then consider a smoothness functional for functions defined on this space. The distance measure encodes our belief for similar points, and smoothness functional shows our desire for similar points to get similar labels. However, choosing the *right* distance measure for a domain is not a trivial task. Moreover choosing the right smoothness functional is not obvious. Instead of looking into the domain of (classification) functions and imposing smoothness indirectly on the *range* of functions, the other alternative is to *learn directly* the class of smooth functions without using any smoothness functional and distance measure in the input domain ([AZ04])).

Suppose we have several related learning problems, each of which has its own sample S_i and hypothesis space $\mathcal{H}_i$. We may have a different learning algorithm $\mathcal{A}_i$ for each problem. Suppose we run our learning algorithms, and get a classifier $f_i(.)$ for each problem. The point is that good classifiers automatically encode smoothness requirement w.r.t the *intrinsic* similarity structure of the input space $\mathcal{X}$. We may not know the intrinsic similarity structure (or geometry) of $\mathcal{X}$, but by looking into the range of the resulted functions we can extract these smoothness constraints. More abstractly, by investigating the output classifiers, the common structure among them can be exploited to design a better classifier for a new learning task (figure 3).

To do the algorithmic implementation of this idea, assume that the hypothesis spaces of all learning tasks is parameterized by a common parameter θ, so for each problem we have $\mathcal{H}_{i\theta}$ as its hypothesis space. Furthermore, assume that for each problem, we have a procedure $\mathcal{O}_i$ that given θ, sample S_i and additional information T_i (such as a validation set), can evaluate the performance of the learned classifier $f_{i\theta}$. In structural learning, we find the best $\hat{\theta}$ by solving the following optimization problem:

$$\hat{\theta} = \arg\min_{\theta} r(\theta) + \sum_{i} \mathcal{O}_i(S_i, T_i, \theta)$$

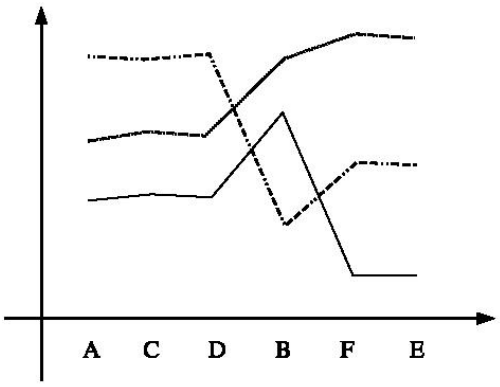

Figure 3: These three plots show the learned classifiers for 3 learning tasks in a discrete domain $\mathcal{X} = \{A, B, C, D, E, F\}$ ([AZ04]). All classifiers assign same values to $\{A, C, D\}$ and same values to $\{E, F\}$. So we can conclude that a good *smooth* classifier assigns similar values to $\{A, C, D\}$ and similar values to $\{E, F\}$.

where $r(\theta)$ is a regularization term showing our prior belief about the preferred values of θ. As an example if T_i be a validation set, then the validation procedure might be:

$$\mathcal{O}_i(S_i, T_i, \theta) = \frac{1}{|T_i|} \sum_{x_j \in T_i} loss(f_{i\theta}(x_j^i), y_j^i, x_j^i)$$

where $f_{i\theta}$ is the learned classifier. This method could be computationally expensive and infeasible if the possible values for θ is large, e.g. it is a continuous variable, because for each value of θ the classification function $f_{i\theta}$ must be learned and then its performance be estimated. A more natural formulation of the problem is to pose an optimization problem in the joint space of classifiers $f_{i\theta}$ and θ on the training set:

$$[\theta_{opt}, f_{i\theta}] = \arg\min_{\theta, f_i} \sum_{i=1}^{m} \left(\sum_{j \in S_i} \frac{loss(f_i(\theta, x_j^i), y_j^i, x_j^i)}{|S_i|} + r_i(f_i) \right) + r_0(\theta) \qquad (11)$$

where $r_i(.)$ is the regularization term to control the model complexity for the ith learning problem, and $r_0(.)$ encodes our prior belief about the value of θ.

This approach can be applied to the semi-supervised learning problem as follows. First it generates a lot of *auxiliary problems* from unlabeled data which are related to the main learning problem, and then tries to learn a classifier for each set of these

training data. After all the method looks for common structure among these classifiers (or predictors) so that it can use this information in building a new classifier for the main task (which consists of labeled data only).

4.6.1 Linear Model for Structural Learning

The derivation of the linear case is given in [AZ04] and [AZ05]. Suppose the main classifier is linear: $f(x) = sign(u \cdot x)$. At first a lot of, let say m, auxiliary problems are generated from the unlabeled data, and then a linear classifier f_i is learned for each of them with the weight vector u_i. Consider these m weight vectors as points in a high dimensional space, so a low dimensional structure can be sought among these *points*. In fact we want to do dimensionality reduction in the space of weight vectors. We can assume square regularization of weight vectors, and assume $u_i = w_i + \theta^T \cdot v_i$ where θ specifies the common structure, and (w_i, v_i) have to be optimized for the following objective function:

$$[\hat{\theta}, \{\hat{w}_i, \hat{v}_i\}_{i=1}^m] = \arg \min_{\theta, \{w_i, v_i\}} \sum_{i=1}^{m} \left(\sum_{j \in S_i} \frac{loss((w_i + \theta^T \cdot v_i)^T \cdot x_j^i, y_j^i, x_j^i)}{|S_i|} + \lambda_i \|w_i\| \right)$$

subject to the constraint $\theta \cdot \theta^T = I$. The regularization term $r_0(.)$ in (11) is not needed since it is absorbed to the constraint.

4.6.2 Discussion

The idea in this section is related to the agreement boosting approach that we saw before. In agreement boosting, we have *one* problem, and the classifiers (corresponding to multiple views) must agree on the label assigned to unlabeled data. Here we have several problems, and the classifiers (corresponding to different learning tasks) must agree on their *behavior* on the input space.

5 Semi-Supervised learning for Structured Domains

Inductive semi-supervised learning aims to learn, from labeled and unlabeled data, a classifier $f(.)$ that assigns an output y to an input x. However, for many interesting cases the input is a complex object and the output is a structured label. For example the input might be a sentence (a sequence of words) and the output might be its parts-of-speech tags (a sequence of labels).

In structured (inductive) semi-supervised learning, a sample including labeled and unlabeled complex objects is given, and the goal is to find a classifier which can be used to assign structured labels to unlabeled data as well as future unseen objects. One way to solve this problem is to forget about the structure among output labels and predict them independently. However, the hope is that more accurate classifiers can be constructed by taking into account the inter-dependencies among output labels.

This problem can be approached based on either generative or discriminative methods. In the generative methods such as hidden markov models (HMM), a joint probability distribution over input and output is considered and its parameters are estimated based on the available data. Then an structured label for a new object can be predicted based on the joint probability distribution (estimated by the model) for each structured label.

In the discriminative setting, we are looking for a scoring function $\mathcal{S}(x, y)$ which assigns a compatibility score to an object x and its proposed label y. In classifying a new object, all of the feasible labelings $\mathcal{Y}_x$ of the object x are examined and the best one is selected[15]:

$$y = \arg\max_{y' \in \mathcal{Y}_x} \mathcal{S}(x, y') \tag{12}$$

As an example, for a sentence x, $\mathcal{Y}_x$ might be the collection of its parse trees based on a probabilistic context free grammar. Usually other assumptions, such as smoothness of $\mathcal{S}(.,.)$ over (an induced) geometry of x, are also made so that unlabeled data can be used effectively in this setting.

[15]Often, $\mathcal{Y}_x$ is exponentially large.

5.1 Background

Consider a sample $S = \{(x_i, y_i)\}_1^l \cup \{x_j\}_{l+1}^{l+u}$ consisting of labeled and unlabeled instances. Let $\mathcal{X}$ be the set of all complex objects and $\mathcal{Y}$ be the set of all structured labels. Often it is assumed that each input-output pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ can be decomposed into a set $R(x, y) \subseteq \mathcal{R}$ of simpler constituent parts[16] where $\mathcal{R}$ is a countable set of parts. As an example, for a sentence and the chain of its parts-of-speech tags, the simple parts are edges connecting adjacent nodes in the chain of tags. In general if the inter-dependencies of the output form a graph, the simple parts can be the cliques of this graph. For an unlabeled object x, the set of its simple parts is defined as $R(x) = \cup_{y \in \mathcal{Y}_x} R(x, y)$, i.e. the parts of all of its feasible labelings. Let $R(S)$ to be the set of all simple parts of labeled and unlabeled objects in the sample S.

Each part $r \in \mathcal{R}$ can be represented as a vector $\phi(r)$ in a feature space $\mathbb{R}^d$. As a simple example, form a big vector such that each position in the vector corresponds to a part, in other words the dimensionality of this vector is $|\mathcal{R}|$. Now for representing r, put a single one in the position corresponding to this part and zero else where. Usually an input-output pair (x, y) is embedded into the feature space $\mathbb{R}^d$ using the following mapping:

$$\Phi(x, y) = \sum_{r \in R(x,y)} \phi(r) \tag{13}$$

5.2 Discriminative Approach

Based on the discriminative formulation (12), the problem will be solved by finding any *good* scoring function $\mathcal{S}$. A good scoring function should have a good generalization capability. Furthermore, we would like to use the information contained in unlabeled data in designing a good scoring function. The assumption is that the scoring function can be written as the sum over the scores of the simple parts:

$$\mathcal{S}_f(x, y) = \sum_{r \in R(x,y)} f(r)$$

[16] The terminology is taken from [BCTM04]

where f is a scoring function applied, in a lower level, to the simple parts. Hence the problem is reduced to finding a good f.

Suppose f is searched in some functional space $\mathcal{H}$. It should have a low training error on labeled data $\sum_i loss(f(x_i), y_i, x_i)$ where $loss$ is the loss function. Moreover, if there are several functions achieving the same minimum error value, the simpler one is preferred (Occam Razor). Briefly, the best function is the solution of the following optimization problem:

$$\arg\min_{f\in\mathcal{H}} \sum_i loss(f(x_i), y_i, x_i) + \|f\|^2_{\mathcal{H}}$$

where $\|f\|_{\mathcal{H}}$ refers to the norm of the function in the functional space and is a measure of its complexity. Indeed, $\mathcal{H}$ is the reproducing kernel Hilbert space $\mathcal{H}_k$ corresponding to a Mercer kernel $k : \mathcal{R} \times \mathcal{R} \to \mathbb{R}$ which assigns high values to similar parts.

Up to now, all things depend on labeled data; what about unlabeled data? We add another term which shows the dependency of functions f on unlabeled data. We notice that on *similar* simple parts, the value of f must be similar, i.e. it changes slowly or smoothly over the space of all simple parts. So we consider the following optimization problem:

$$\arg\min_{f\in H_k} \sum_{i=1}^{l} loss(x_i, y_i, f) + \lambda_1 \|f\|^2_{\mathcal{H}_k} + \lambda_2 \|f\|^2_I \tag{14}$$

where the last term in the above optimization problem connects f to the density of parts which, in turn, is estimated by parts in the input sample $R(S)$. As an example for $\|f\|_I$, we can construct d-nearest neighbor graph on $R(S)$ and force f to be smooth on this neighborhood structure:

$$\|f\|^2_I = \sum_{r,r'\in R(S)} W_{r,r'}(f(r) - f(r'))^2 \tag{15}$$

where $W_{r,r'} = 1$ for connected parts r and r'. Note that $K(r, r')$ can be used to infer distances in the designing of nearest neighbor graph. Any *reasonable* kernel I can be used in (14) as far as it is not the same as K. If the two kernels be the same, the optimum function will only depend on labeled points and unlabeled data is useless.

5.2.1 Semi-supervised Structured SVM

In this section we show how to extend structured support vector machines to handle semi-supervised learning via the proposed framework. Suppose regularization term (15) is used in the optimization problem (14), then based on the Representer theorem the best function f^{opt} can be written as follows ([AMB05]):

$$f^{opt}(r') = \sum_{r \in R(S)} w_r k(r, r')$$

The norm of f in the functional space $\mathcal{H}_k$ is $\|f\|^2_{\mathcal{H}_k} = \mathbf{w}^T G \mathbf{w}$ where $\mathbf{w}$ is the vector of coefficients (its dimensionality is $|R(S)|$) and G is the gram matrix $G_{ij} = k(r_i, r_j)$. Furthermore, the input dependent regularization term for f is written as $\|f\|^2_I = \mathbf{w}^T G(D - W)G\mathbf{w}$ where D is the diagonal matrix each of its elements is the sum of the corresponding row in W. Therefore the optimization problem (14) becomes the following:

$$\arg\min_{\mathbf{w}} \sum_{i=1}^{l} loss(f_{\mathbf{w}}, y_i, x_i) + \mathbf{w}^T Q \mathbf{w}$$

where $Q = \lambda_1 G + \lambda_2 G(D - W)G$, and $f_{\mathbf{w}}$ is to emphasize that the weight vector $\mathbf{w}$ is used in the construction of the function f.

Now we are left with the specification of the loss function. Like maximum margin Markov models ([TCKG05]), for the labeled objects (x, y) the requirement is that the difference between the score of the true label $\mathcal{S}_f(x, y)$ and any other label $\mathcal{S}_f(x, y'), y' \in \mathcal{Y}_x$ be greater than the distance of the label to the true label for this object $\Delta(x, y, y')$ (up to the addition of some slack variable). $\Delta(x, y, y')$ could be the hamming distance between the two labels y and y'. Hence, the best scoring function is the solution of the following quadratic problem:

$$\arg\min_{\mathbf{w}, \varepsilon} \sum_{i=1}^{l} \varepsilon_i + \mathbf{w}^T Q \mathbf{w}$$

$$\underbrace{\mathbf{w}^T \cdot G \cdot \Phi(x_i, y_i)}_{\mathcal{S}_f(x_i, y_i)} - \underbrace{\mathbf{w}^T \cdot G \cdot \Phi(x_i, y')}_{\mathcal{S}_f(x_i, y')} \geq \Delta(x, y_i, y') - \varepsilon_i, \forall y' \in \mathcal{Y}_{x_i}, \forall i$$

Based on the structured SVM formulation ([THJA04]), the following optimization problem can be posed:

$$\arg\min_{\mathbf{w},\varepsilon} \sum_{i=1}^{l} \varepsilon_i + \mathbf{w}^T Q \mathbf{w}$$

$$\mathbf{w}^T \cdot G \cdot \Phi(x_i, y_i) - \mathbf{w}^T \cdot G \cdot \Phi(x_i, y') \geq 1 - \frac{\varepsilon_i}{\Delta(x, y_i, y')}, \forall y' \in \mathcal{Y}_{x_i}, \forall i$$

Maximum margin formulation and SVM formulation only differ in the right hand side of the constraints.

5.2.2 Semi-supervised KCRF

Kernel conditional random field (KCRF) ([LZL04]) puts a probability distribution over all possible structured labelings of a particular input x. In KCRF the negative log-likelihood is considered as the loss function for labeled data, i.e. $loss(f_{\mathbf{w}}, y, x) = -\log p(y|x, f_{\mathbf{w}})$. The conditional probability distribution of $p(y|x, \mathbf{w})$ is a gibbs distribution parameterized by the weight vector $\mathbf{w}$:

$$p(y|x, \mathbf{w}) = \frac{\exp[\mathbf{w}^T \cdot G \cdot \Phi(x, y)]}{\sum_{y' \in Y_x} \exp[\mathbf{w}^T \cdot G \cdot \Phi(x, y')]}$$

The weight vector is found by maximizing the posterior distribution $p(\mathbf{w}|S) \propto p(\mathbf{w})p(S|\mathbf{w})$ where as before S is the training sample including labeled and unlabeled data and the prior distribution over the weight vector is assumed to be $p(\mathbf{w}) = \exp[-\frac{\lambda}{2}\mathbf{w}^t \cdot Q \cdot \mathbf{w}]$. The prior distribution encodes the information contained in the unlabeled data and our belief that the norm of the weight vector should be small so that it has good generalization capability:

$$\log p(\mathbf{w}|S) = -\frac{\lambda}{2}\mathbf{w}^T \cdot Q \cdot \mathbf{w} + \sum_{i=1}^{L} \left(\mathbf{w}^T \cdot G \cdot \Phi(x_i, y_i) - \left(\log \sum_{y' \in \mathcal{Y}_{x_i}} \exp[\mathbf{w}^T \cdot G \cdot \Phi(x_i, y')] \right) \right)$$

The gradient of the above expression can be used to find the optimal weight vector via gradient descent or any other optimization algorithm.

6 Conclusion

We reviewed different theoretical and practical issues related to semi-supervised learning problem. Based on [BB05, CZS06, CC95], a theoretical framework for thinking about semi-supervised learning was presented. This framework generalizes the standard PAC model for fully supervised learning to the semi-supervised learning scenario. As its authors mentioned in [CZS06], several proposed algorithms for semi-supervised learning can be fit into this framework. Then a statistical analysis for usability of ML estimator was mentioned based on [Cas94, CCS$^+$04]. Several learning algorithms were mentioned in the middle sections afterwards. As we saw, there are two main strategies for attacking semi-supervised learning problem. The first strategy starts from a classifier (or an ensemble of classifiers) and uses unlabeled data to (iteratively) enhance the classifier(s) [Yar95, Abn04, BM98, NG00, Les05, CS99, NMTM00]. The focus of the second strategy is on the data to discover the (possible) structure inherent in the data and exploit this information to characterize good classifiers [BCH04, BNS05, CJ03, CJ04, ZL05, ZGL03, AZ05]. We agree with [See00] that prior knowledge is very important to successfully take advantage of unlabeled data in constructing a good classifier. As the number of labeled data grows, the risk of wrong assumptions decreases, and as unlabeled data increases, wrong assumptions might be very dangerous ([See00]). It seems that semi-supervised learning in the structured domains ([AMB05]) needs more attention, and more work can be done in this exciting direction. For future research, we mention two directions: extending Abney's analysis ([Abn04]) of the Yarowski algorithm, and investigating more practical semi-supervised algorithms for the structured prediction problem.

Acknowledgments

We are very thankful to Anoop Sarkar for many insightful discussions. We also thank V. Castelli, Y. Altun, and H. Daume for sending their research manuscripts to us.

References

[Abn04] S. Abney. Understanding the yarowsky algorithm. *Computational Linguistics*, 30(3), 2004.

[AMB05] Y. Altun, D. Mcallester, and M. Belkin. Maximum margin semi-supervised learning for structured variables. In *Proceedings of Neural Information Proccessing Systems*, 2005.

[AZ04] R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. Technical report, IBM Watson Research, 2004.

[AZ05] R. Ando and T. Zhang. A high-performance semi-supervised learning method for text chunking. In *Proceedings of Association for Computational Linguistics*, 2005.

[BB05] M. Balcan and A. Blum. A pac-style model for learning from labeled and unlabeled data. In *Proceedings of Computational Learning Theory*, 2005.

[BBY04] N. Balcan, A. Blum, and K. Yang. Co-training and expansion: Towards bridging theory and practice. In *Proceedings of Neural Information Proccessing Systems*, 2004.

[BCH04] O. Bousquet, O. Chapelle, and Matthias Hein. Measure based regularization. In *Proceedings of Neural Information Proccessing Systems*, 2004.

[BCTM04] P. Bartlett, M. Collins, B. Taskar, and D. McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *Proceedings of Neural Information Proccessing Systems*, 2004.

[BM98] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of Computational Learning Theory*, 1998.

[BNS04] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: a geometric framework for learning from examples. Technical report, University of Michigan, 2004.

[BNS05] M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In *AISTAT*, 2005.

[Cas94] V. Castelli. *The Relative Value of Labelled and unlabelled Samples in Pattern Recognition*. PhD thesis, Stanford University, 1994.

[CC95] V. Castelli and T. M. Cover. On the exponential value of labeled samples. *Pattern Recogn. Lett.*, 16(1):105–111, 1995.

[CCS+04] Ira Cohen, Fabio G. Cozman, Nicu Sebe, Marcelo C. Cirelo, and Thomas S. Huang. Semi-supervised learning of classifiers: Theory, algorithms for bayesian network classifiers and application to human-computer interaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2004.

[CJ01] A. Corduneanu and T. Jaakkola. Stable mixing of complete and incomplete informaation. Technical report, CSAIL, MIT, 2001.

[CJ02] A. Corduneanu and T. Jaakkola. Continuation methods for mixing hetrogenuous sources. In *Proceedings of Uncertainty in Artificial Intelligence*, 2002.

[CJ03] A. Corduneanu and T. Jaakkola. On information regulrization. In *Proceedings of Uncertainty in Artificial Intelligence*, 2003.

[CJ04] A. Corduneanu and T. Jaakkola. Distributed information regularization on graphs. In *Proceedings of Neural Information Proccessing Systems*, 2004.

[CS99] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of EMNLP*, 1999.

[CWS03] O. Chapelle, J. Weston, and B. Schoelkopf. Cluster kernels for semi-supervised learning. In *Proceedings of Neural Information Proccessing Systems*, 2003.

[CZ05] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *AISTAT*, 2005.

[CZS06] O. Chapelle, A. Zien, and B. Scholkopf, editors. *Semi-Supervised Learning*. MIT Press, 2006.

[DML01] S. Dasgupta, D. McAllester, and M. Littman. Pac generalization bounds for co-training. In *Proceedings of Neural Information Proccessing Systems*, 2001.

[GB04] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Proceedings of Neural Information Proccessing Systems*, 2004.

[Her01] Ralf Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, Cambridge, MA, USA, 2001.

[Joa99] T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, 1999.

[Les05] B. Leskes. The value of agreement: A new boosting algorithm. In *Proceedings of Computational Learning Theory*, 2005.

[LZL04] J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: Representation and clique selection. In *Proceedings of International Conference on Machine Learning*, 2004.

[NG00] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, 2000.

[NMTM00] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 30(3), 2000.

[RJZH04] S. Rosset, H. Zou J. Zhu, and T. Hastie. A method for inferring label sampling mechanisms in semi-supervised learning. In *Proceedings of Neural Information Proccessing Systems*, 2004.

[See00] M. Seeger. Learning with labeled and unlabeled data. Technical report, Edinburge, 2000.

[SJ02] M. Szummer and T. Jaakkola. Information regularization with partilly labelled data. In *Proceedings of Neural Information Proccessing Systems*, 2002.

[SL94] B. Shahshahani and D. Landgrebe. The effect of unlabeled samples in reducing the small sample size problem and mitigating the hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, 1994.

[SNB05] V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *Workshop on Learning with Multiple Views, Proceedings of International Conference on Machine Learning*, 2005.

[TCKG05] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of International Conference on Machine Learning*, 2005.

[THJA04] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of International Conference on Machine Learning*, 2004.

[VB03] P. Vincent and Y. Bengio. Density-sensitive metrics and kernels. In *Snowbird Learning Workshop*, 2003.

[Yar95] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of Association for Computational Linguistics*, 1995.

[ZGL03] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of International Conference on Machine Learning*, 2003.

[ZL05] X. Zhu and J. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proceedings of International Conference on Machine Learning*, 2005.