Inductive Semi-supervised Learning with Applicability to NLP

Anoop Sarkar and Gholamreza Haffari

anoop,ghaffar1@cs.sfu.ca

School of Computing Science

Simon Fraser University

Vancouver, BC, Canada

http://natlang.cs.sfu.ca/

Outline

- Introduction to Semi-Supervised Learning (SSL)
- Classifier based methods: Part 1
  - EM, Stable mixing of Complete and Incomplete Information
- SSL using Generative Models for Structured Labels
- Classifier based methods: Part 2
  - Self-training, the Yarowsky Algorithm, Co-training
- Data based methods
  - Manifold Regularization, Harmonic Mixtures, Information Regularization
  - Learning Predictive Structure from Multiple Tasks
- SSL using Discriminative Models for Structured Labels

Learning Problems

- Supervised learning:
  - Given a sample consisting of object-label pairs (x_i,y_i), find the predictive relationship between objects and labels.

- Un-supervised learning:
  - Given a sample consisting of only objects, look for interesting structures in the data, and group similar objects.

- What is Semi-supervised learning?
  - Supervised learning + Additional unlabeled data
  - Unsupervised learning + Additional labeled data

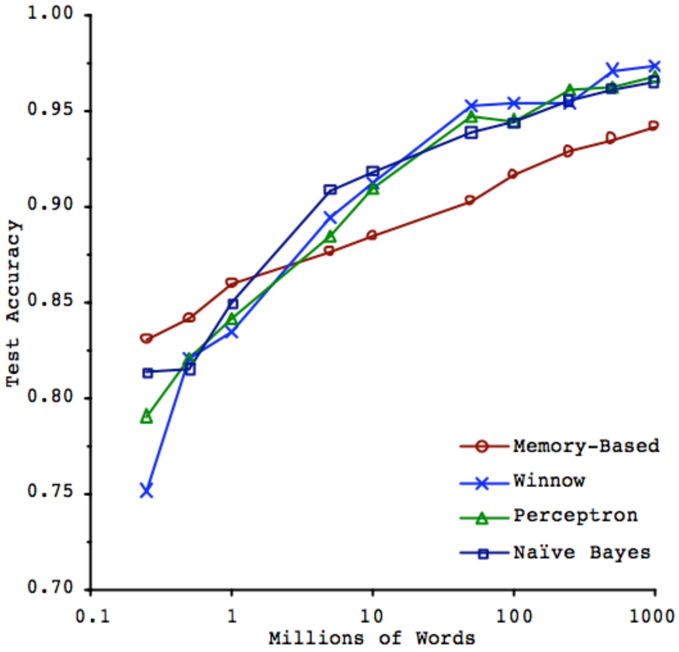
Motivation for SSL

- Pragmatic:
  - Unlabeled data is cheap to collect (compared to labeled data).
  - Example: Classifying web pages,
    - There are some annotated web pages.
    - A huge amount of un-annotated web pages are easily available by crawling the web.

- Philosophical:
  - The brain can exploit unlabeled data.
  - Learning in a setting where data is randomly labeled or labeled by a lazy teacher.
  - Reduces to unsupervised learning in the worst case.

Why should more data help?

(Banko & Brill, 2001)



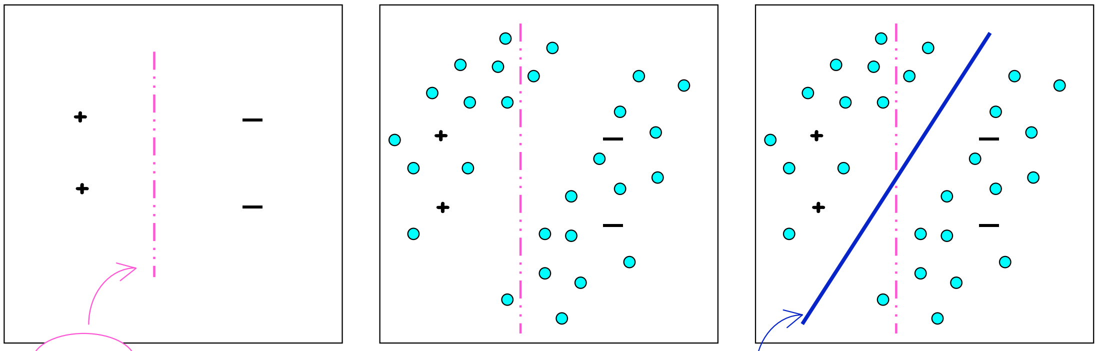
Why should unlabeled data help?

- If you have labeled data, why bother with unlabeled data?
  - Don't! If you have sufficient labeled data or very few parameters = no sparse data problem: rarely occurs in NLP!
  - Injecting unlabeled data can be a way to address the sparse data problem
  - Too many languages: cannot afford to annotate a million words in each one
  - For task of predicting *y* given *x*, you (should?) have a good idea of how to predict p(*x*)
  - *Redundantly sufficient* learners can be built (Mitchell, 1999)
  - We provide more intuition in different learning settings in the next few slides

Preliminaries ...

Intuition in Large-Margin Setting

Training a Support Vector Machine
(SVM), input *x*, label = {+,-}



SVM

Labeled data only

Transductive SVM

Key idea: avoid regions with high values for p(*x*)

(Vapnik, 2000; Joachims, 1999)

(Zhang and Oles, 2000)

(Dempster, Laird & Rubin, 1977)

Intuition when using the EM algorithm

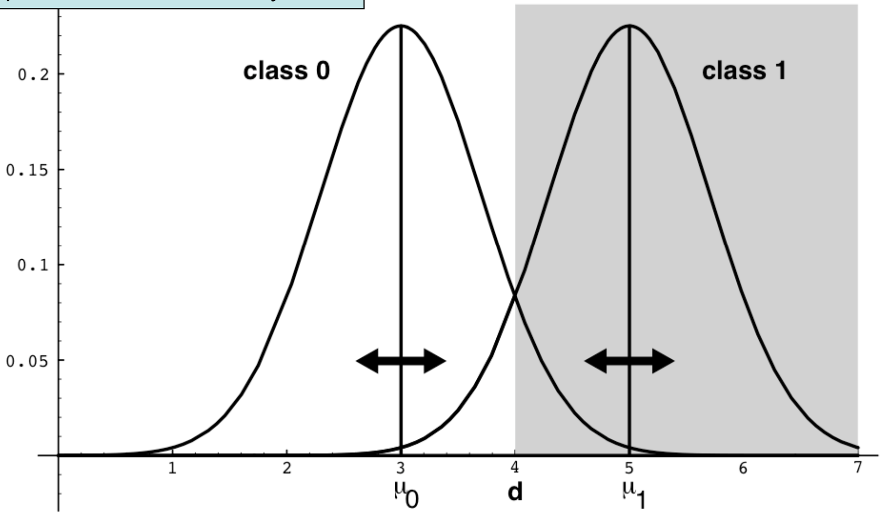
(Castelli, 1994; Castelli and Cover, 1995; Castelli and Cover, 1996; Cohen et.al., 2004)

- Assume we have "good" generative model for the data
- That is, there is a parameter setting such that probability p(x,y) from the generative model captures the labeled data, and
- There is a parameter setting such that probability p(x) from the generative model captures the probability of the unlabeled data
- Then EM can be used to identify which unlabeled examples belong to the same class without knowing the true class label
- Labeled data can then be used to identify the actual class label for each group
- For example, let us consider that a mixture of two Gaussians is a good model for our binary classification task

Intuition when using the EM algorithm

Figure from (Nigam et.al., 2000)

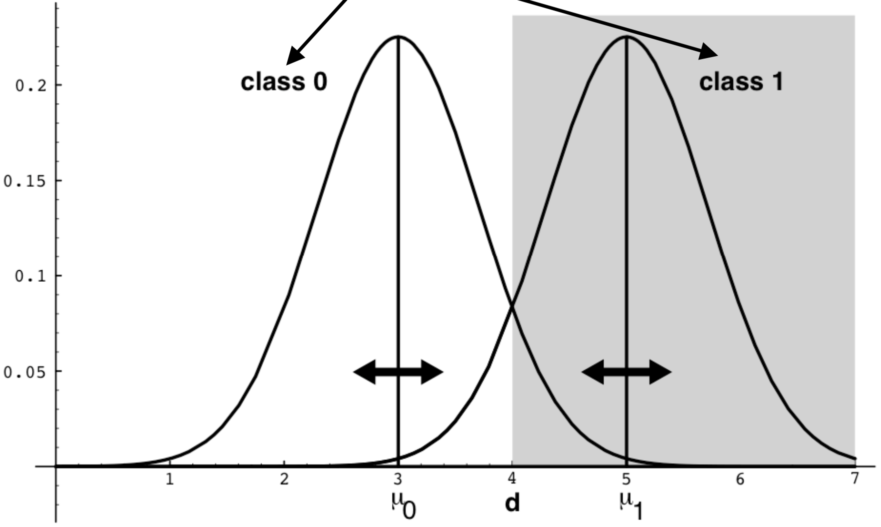
The parameters μ_0, μ_1, the variance and mixing parameters can be learnt using EM which gives the Bayes optimal decision boundary **d**



Intuition when using the EM algorithm

But the clusters are not labeled yet! **Assume** that labeled data can be used to identify the class labels for each cluster.

(Castelli and Cover, 1995) show that this process converges exponentially wrt number of labeled examples



Intuition when using the EM algorithm

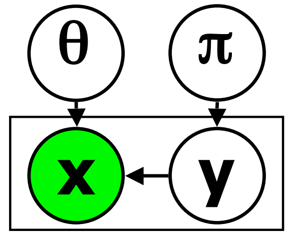
- Note that we had to assume that the labeled data can be used to identify the two classes
- Mixtures of Gaussians are known to be identifiable; the details are in (Castelli & Cover, 1995)
- However, other kinds of models may not be identifiable;
- For more on this see the (Zhu, 2005) survey which covers `Identifiability’

Intuition for Generative models

(Zhang & Oles, 2000; Seeger, 2000)

- Class distributions P(x|y,θ) and class prior P(y|π) are parameterized by θ and π, and used to derive:

$$P(y|x,\theta,\pi) \propto P(x|y,\theta) \cdot P(y|\pi)$$



- Unlabeled data gives information about the marginal P(x|θ,π) which is:

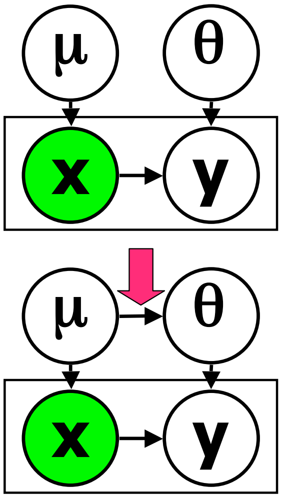
$$P(x|\theta,\pi) = \sum_y P(x|y,\theta) \cdot P(y|\pi)$$

- Unlabeled data can be incorporated naturally

Intuition for Discriminative models

(Zhang & Oles, 2000; Seeger, 2000)

- In Discriminative approach $P(y|\mathbf{x},\theta)$ and $P(x|\mu)$ are directly modeled.
- Unlabeled data gives information about μ, and P(y|x) is parameterized by θ**.**
- If μ affects θ then we are done!
  - Impossible: θ and μ are independent given unlabeled data.
- What is the cure?
  - Make μ and θ a priori dependent.
  - **Input Dependent Regularization**



Semi-Supervised Learning Methods

- A wide variety of methods exist:
  - EM with generative mixture models (mix L + U)
  - Self-training
  - Co-training
  - Data based methods
  - Transductive Support Vector Machines (TSVMs)
  - Graph-based methods
- In this tutorial we will make a distinction between:
  - Inductive SSL methods
  - Transductive SSL methods

Inductive vs.Transductive

- Transductive: Produce label only for the available unlabeled data.
  - The output of the method is not a classifier.
- Inductive: Not only produce label for unlabeled data, but also produce a classifier.
- Analogy from (Zhu, 2005):
  - Transductive learning: take-home exam
  - Inductive learning: in-class exam

Inductive vs.Transductive

- Based on our definition: a Transductive SVM (TSVM) is an inductive learner!
- This is because TSVM can be naturally used on unseen data
- However, the name TSVM originates from the following argument given in (Vapnik, 1998)
  - Learning on the entire data space is solving a more difficult problem
  - If task is to annotate the test data, then only work on the observed data (L+T): solve a simpler problem first!

Inductive vs.Transductive

- TSVM can be seen as a different way to do supervised learning:
  - we can get around i.i.d. assumption by learning a classifier geared towards each test case (or all test cases together)
  - e.g. when learning to recognize handwriting, transduction should help if all test cases were handwritten digits by the same person; compare with (Hinton and Nair, 2005)
- Training a TSVM is NP-hard
- But approximations exist: e.g. (Joachims, 1999) and many others
- (Zhang and Oles, 2000) argue against TSVMs, but empirically TSVMs seem to be beneficial

Inductive vs.Transductive

- (Goutte et. al., 2002) use transductive SVMs for finding named entities in Medline abstracts, i.e. learns a binary classifier
- (Niu, Ji & Tan, 2005) provide a semi-supervised feature clustering algorithm for word-sense disambiguation; it is transductive because it clusters features from test data with those from the training data
- There are many IR related works in this area, see e.g.
  - (Joachims, 1999) for text classification and
  - (Okabe, Umemura and Yamada, 2005) for query expansion
- If TSVM is not an example of a transductive SSL method, then what is?

Inductive vs.Transductive

Example

- Graph mincuts (Blum and Chawla, 2001)
  - Pose SSL as a graph mincut (also called *st*-cut) problem
  - Two class classification setting: Positive labels (1) act as sources and Negative labels (0) as sinks
  - Unlabeled nodes are connected to other nodes with weights based on similarity between examples (L or U)
  - Objective is to find a minimum set of edges to remove so that all flow from sources to sinks is blocked
  - In other words, given the constraint that each label y_i is either 0 or 1 the task is to minimize the function:

Do not change labels on labeled data: weight is infinity

$$\infty \sum_{i\in L}(y_i - y_{i|L})^2 + \frac{1}{2}\sum_{i,j} w_{ij}(y_i - y_j)^2$$

Provides the "flow" across the edges in the graph

Inductive vs.Transductive

- Graph mincuts have been used in NLP: (Pang and Lee, 2004)
- To train a binary sentence classifier: subjective vs. objective
  - Can be used to create a subjective extract/summary from a movie review
- Then the extract is categorized as a positive/negative review
- Labeled data external to dataset was used to train sentence level subjective vs. objective classifiers
  - Sentences were labeled using this classifier trained on labeled data
- Unlabeled data: sentences were given weights based on simple proximity to other sentences
- Graph mincut method was used to extract the sentences "attracted" to the subjective class

Focus on Inductive SSL

- For this tutorial we will focus on Inductive SSL methods: Why?
- Most graph-based transductive methods scale badly with respect to the time complexity, which is typically O(n^3)
- It is possible to improve the complexity to O(n) but these ideas are based on assumptions or methods that may or may not apply to common NLP tasks
- Most interest in NLP is for the use of very large datasets (like those used to train language models) and inductive methods are more suitable for such a setting

Focus on Inductive SSL

- Other surveys do a good job of covering transductive SSL methods, see
  - Semi-Supervised Learning (Chappelle et.al., to appear)
  - Semi-Supervised Learning Literature Survey (Zhu, 2005)
  - Learning with Labeled and Unlabeled Data (Seeger, 2000)
  - Learning from L and U data: An Empirical Study Across Techniques and Domains (Chawla & Karakoulas, 2005)
- In particular Chapter 25 of (Chappelle et. al., to appear) which is available online has a lengthy discussion comparing semi-supervised learning and transductive learning.

Note: these surveys also cover many Inductive SSL methods!

Two Algorithmic Approaches

- Classifier based methods:
  - Start from initial classifier(s), and iteratively enhance it (them)
- Data based methods:
  - Discover an inherent geometry in the data, and exploit it in finding a good classifier.

Classifier based Methods: Part 1

- The first classifier based method we will study uses the EM algorithm which provides the Maximum Likelihood (ML) estimates for unlabeled data treating the labels as hidden data
- For labeled data: ML estimates usually reduce to simply the relative frequency counts
- For unlabeled data: ML estimates are computed using EM (an iterative re-estimation algorithm to find parameter values)
- If we have a prior over models, MAP estimation can be used to find a good model from the space of all models and likelihood of the data given this model.
- We will assume some familiarity with EM in this tutorial (but typically we will use it as a black box inside other algorithms)

EM: Combining Labeled and Unlabeled Data

(Dempster, Laird & Rubin, 1977)

- Basic EM:
  1. Initialize model using parameter estimation from labeled data
  2. Re-estimate model on unlabeled data using the EM algorithm
  3. Return model after EM converges; This model is used to measure performance on test data

Forward link ...

EM: Mixtures of Labeled and Unlabeled Data

(Dempster, Laird & Rubin, 1977)

- Use EM to maximize the joint log-likelihood of labeled and unlabeled data:

$$\sum_i \log \Big(P(y_i|\pi) P(x_i|y_i,\theta) \Big) +$$

L_l : Log-likelihood of labeled data

$$\sum_j \log \Big(\sum_y P(y|\pi) P(x_j|y,\theta) \Big)$$

L_u : Log-likelihood of unlabeled data

EM: Mixtures of Experts

(Miller and Uyar, 1997; Shahshahani and Landgrebe, 1994)

- Labeled examples from X_L: $(\mathbf{x}_i, y_i)$, where y_i is the label and unlabeled examples $\mathbf{x}_i$ from X_U
- Each input $\mathbf{x}_i$ is generated from a mixture m_i with prob $f(\mathbf{x}_i \mid \theta_i)$
- If there are L mixture components then the density has mixing parameters α_k which sum to 1

$$f(x \mid \theta) = \sum_{k=1}^{L} \alpha_k f(x \mid \theta_k)$$

$$P(m_i = j \mid x_i) = \frac{\alpha_j f(x_i \mid \theta_j)}{\sum_{k=1}^{L} \alpha_k f(x_i \mid \theta_k)}$$

EM: Mixtures of Experts

(Miller and Uyar, 1997; Shahshahani and Landgrebe, 1994)

- Labeled examples from X_L and unlabeled examples from X_U
- The likelihood of the data X_L and X_U is given by:

$$logL = \sum_{x_i \in X_U} \log \sum_{k=1}^{L} \alpha_k f(x_i \mid \theta_k)$$

$$+ \sum_{x_i \in X_L} \log \sum_{k=1}^{L} \alpha_k P(c_i \mid x_i, m_i = k) f(x_i \mid \theta_k)$$

EM: Mixtures of Experts

(Miller and Uyar, 1997; Shahshahani and Landgrebe, 1994)

- (Miller and Uyar, 1997) show that EM can be used to learn the parameters $f(\mathbf{x}_i \mid \theta_i)$ as well as the mixture parameters α_k
- They provide two variants:
  - (1) the mixture components and the class labels are conflated to be the same, and
  - (2) the mixture components are predicted first given the feature value and the class label is predicted given the mixture component

EM: Mixtures of Labeled and Unlabeled Data

- So far we have equal contributions from labeled and unlabeled data
- In practice, it is better to discount the unlabeled data
- We can do this in a mixture model

$$\lambda L_U + (1 - \lambda) L_L$$

- Standard ML estimation means that the value of λ is set proportional to the size of each set, L_L and L_U, but this is not what we want
- We generally discount the estimates from L_U since they are less reliable

EM: Mixtures of Labeled and Unlabeled Data

(Nigam et.al, 2000)

- (Nigam et.al., 2000) combine labeled and unlabeled data for document classification using EM
- Classification model is a Naive Bayes model
- Setting is similar to (Miller and Uyar, 1997) except for model parameters
- $f(\mathbf{x}_\mathrm{i} \mid \theta_\mathrm{i})$ is now a Naive Bayes classifier defined as the product of all the words x_{ij} in document $\mathbf{x}_\mathrm{i}$ given the doc. class/feature value
- The mixture parameters α_k indicate the likelihood of each word in a document belonging to a topic or a class

EM: Mixtures of Labeled and Unlabeled Data

(Nigam et.al, 2000)

- They provide two variants:
  - (1) the mixture components and the class labels are conflated to be the same **and** EM counts are discounted using an extra parameter λ, and
  - (2) each class has several sub-topics each with a word distribution; each word is conditioned on a mixture component and the class label is conditioned on the component (a many to one mapping)
- Both λ and the number of mixture components for unlabeled data are tuned on a held-out set.
- In several expts, these variants of EM are shown to exploit unlabeled data effectively in the document classification task.

EM: Mixtures of Labeled and Unlabeled Data

- (Callison-Burch et. al., 2004) proposes a mixture model for statistical MT
  - The model combines human annotated word-aligned data with EM learnt word-alignments (using IBM models)
  - It uses the discounting idea; Setting $\lambda = 0.9$ (almost all weight on labelled data) seemed to perform the best in the experiments

- (McClosky et. al, 2006) use discounting to combine counts from parsed output with labeled data for statistical parsing: improves parse f-score from 91.3 to 92.1 for parsing WSJ

- (Corduneanu & Jaakkola, 2002) provide a general framework for optimizing the log-likelihood $\lambda L_U + (1 - \lambda) L_L$ and the optimal value of λ

skip ahead ...

Stable Mixing of Information

(Corduneanu & Jaakkola 2002)

- Use λ to combine the log-likelihood of labeled and unlabeled data in an optimal way:

$$\lambda L_U + (1 - \lambda) L_L$$

- EM can be adapted to optimize it.

- Additional task is determining the best value for $\boldsymbol{\lambda}$.

Stable Mixing: EM_λ Operator

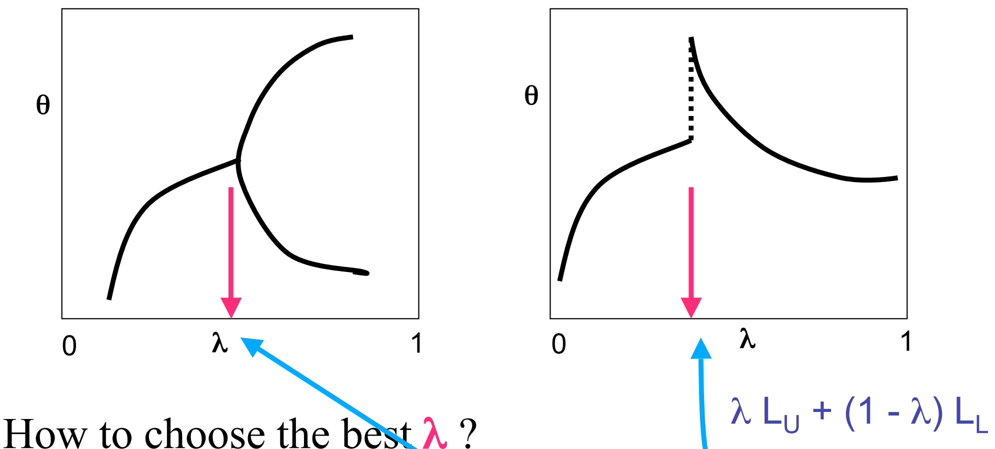
- E and M steps update the value of the parameters for an objective function with particular value of **λ**.
- Name these two steps together as EM_λ operator:

$$\theta^{new} = EM_\lambda(\theta)$$

- The optimal value of the parameters is a fixed point of the EM_λ operator:

$$\theta = EM_\lambda(\theta)$$

Stable Mixing: Path of solutions



$\lambda L_U + (1 - \lambda) L_L$

- How to choose the best λ ?
  - By finding the path of optimal solutions as a function of **λ**
  - Choosing the first λ where a bifurcation or discontinuity occurs; after such points labeled data may not have an influence on the solution.

SSL for Structured Labels

- Generative model based:
  - Hidden Markov Model (HMM)
  - Stochastic Context Free Grammar (SCFG)

- Discriminative model based (to be covered later):
  - Co-Hidden Markov Perceptron
  - Semi-Structured Support Vector Machines (SSVM)
  - Co-Structured SVM (Co-SSVM)
  - Semi-Kernel Conditional Random Fields (KCRF)

Hidden Markov Models

- Hidden Markov Model (HMM) is the standard generative model for sequence learning: input sequence labeled with output sequence of same length.

- EM can be used to train HMM when unlabeled data exists: Forward-Backward algorithm

- Decoding (finding the best label sequence for given input sequence) can be done in linear time by the Viterbi algorithm

Probabilistic Context Free Grammars

- Probabilistic Context Free Grammar (PCFG) is the standard generative model for tree structure: input sequence is labeled with a tree (input = leaves)
- EM can be used to train PCFG when unlabeled data exists: Inside-Outside algorithm
- Decoding (finding the best parse tree for a given input) can be done in polynomial time by the Viterbi algorithm

Basic EM for HMMs

- (Cutting et. al., 1992) used Basic EM with HMMs for a part-of-speech tagging task and produced great results by boosting performance using unlabeled data and EM
- (Brill, 1997) did a similar likelihood estimation in the Transformation-based learning framework
- Both rely on implicitly or explicitly knowing the tag dictionary for words in the unlabeled data.

each word in the *unlabeled data* is associated with a list of possible tags,using clustering or morphological classes

Basic EM for HMMs

- (Merialdo, 1994) and (Elworthy, 1994) used varying amounts of labeled and unlabeled data to test effectiveness of basic EM using HMMs for the part-of-speech tagging task
- Different settings corresponded to varying amounts of supervision (or quality of labeled data)
- (Merialdo, 1994) also tried various constraints to keep p(*t*|*w*) fixed or to keep the marginal probability p(*t*) fixed at each EM iteration -- although these were not very effective
- These expts showed that EM for HMMs seems to work only in cases of very little labeled data and hurts accuracy in all other cases with large or medium amount of labeled data

Basic EM for HMMs

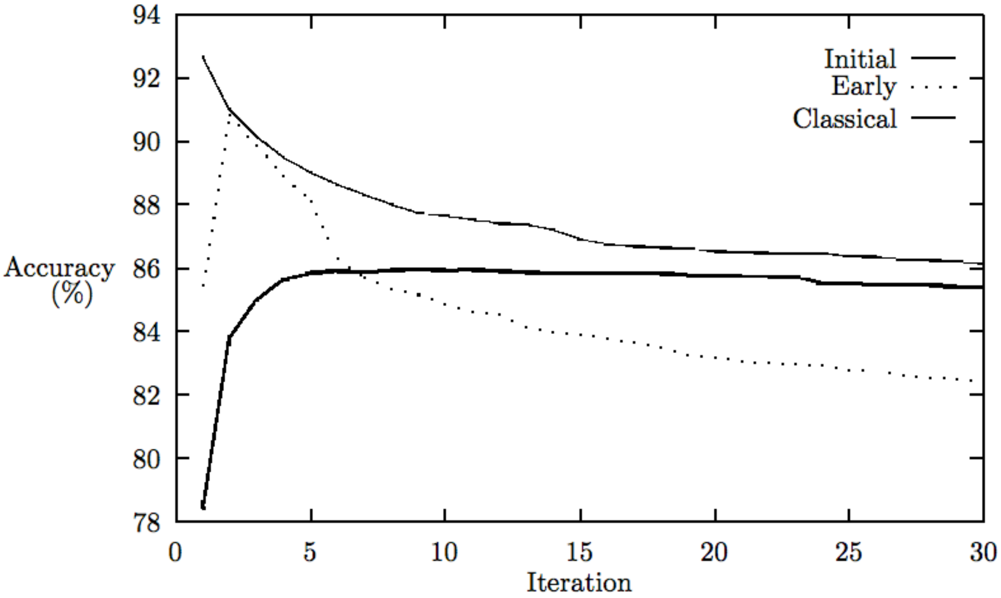


Figure from (Elworthy, 1994)

Classifier based Methods: Part 2

- The second class of classifier based methods we will study are bootstrapping methods
- These include methods like: self-training, the Yarowsky algorithm, co-training, etc.
- In these methods, we start by training model(s) on some labeled data
- Then unlabeled data is labeled using model(s) and some examples are selected to be added as newly labeled examples
- This procedure is iterated and the labeled data set is grown

Self-Training

- Self-training procedure:
  - A classifier is trained with a small amount of labeled data
  - The classifier is then used to classify the unlabeled data
  - Typically the most confident unlabeled points, along with the predicted labels are incorporated into the training set
  - The classifier is re-trained and the procedure is repeated
- This is the simplest form of a bootstrapping method
- Learning using EM is related to self-training;
  - self-training only uses the mode of the prediction distribution

Self-Training

- (Charniak, 1997) reported a single round of self-training on 30M words for statistical parsing: resulted in a 0.2~0.4 point f-score improvement on WSJ parsing

- (McClosky et. al., 2006) improve on vanilla self-training by discounting the events learnt from unlabeled data: resulted in a f-score improvement from 91.3 to 92.1

- (Riloff et. al., 2003; Phillips & Riloff, 2002) use self-training to extract patterns that identify subjective nouns

Self-Training

- (Maeireizo et. al., 2004) use self-training between two classifiers to classify dialogues as *emotional* or *non-emotional*. Each classifier was trained on a single class

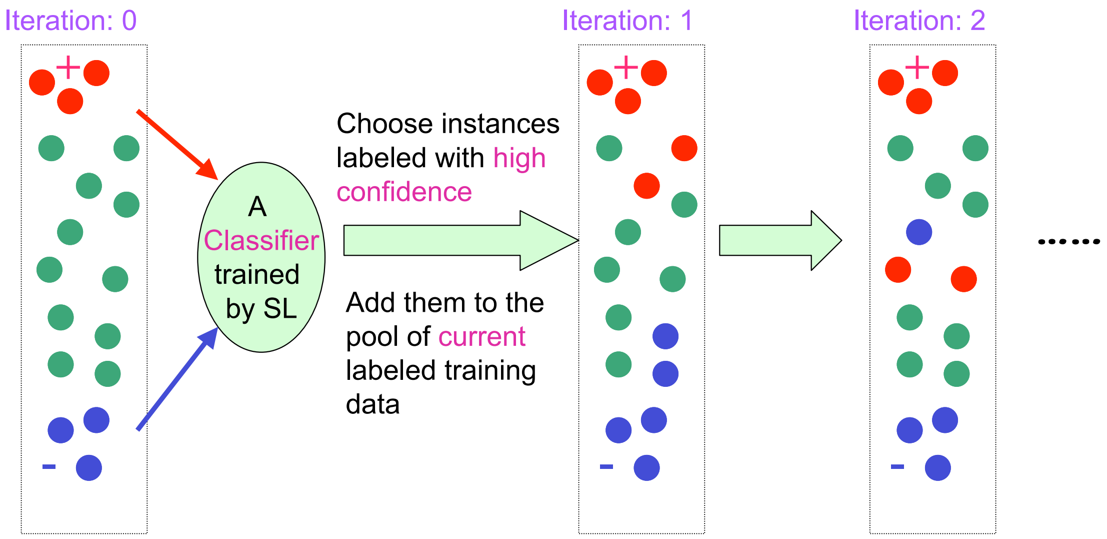
- (Hindle & Rooth, 1993) proposed an idea for prepositional phrase (PP) attachment disambiguation which is now commonly used in self-training for NLP:
- Extract unambiguous cases from a large unlabeled corpus and then use those cases as training data in a disambiguation classifer
- (Ratnaparkhi, 1998) has further experiments along these lines, also for PP attachment

Self-Training

- (Yarowsky, 1995) created a new form of self-training for word-sense disambiguation which incorporated high confidence examples as labeled data
- The algorithm is similar to self-training but also used a second constraint: “one sense per discourse/document” to bootstrap new features.
- We refer to this and other variants of self-training that depend on high precision models as the Yarowsky algorithm

The Yarowsky Algorithm

(Yarowsky, 1995)



Iteration: 0

Iteration: 1

Iteration: 2

A Classifier trained by SL

Choose instances labeled with high confidence

Add them to the pool of current labeled training data

Figure from (Yarowsky, 1995)

The Yarowsky Algorithm

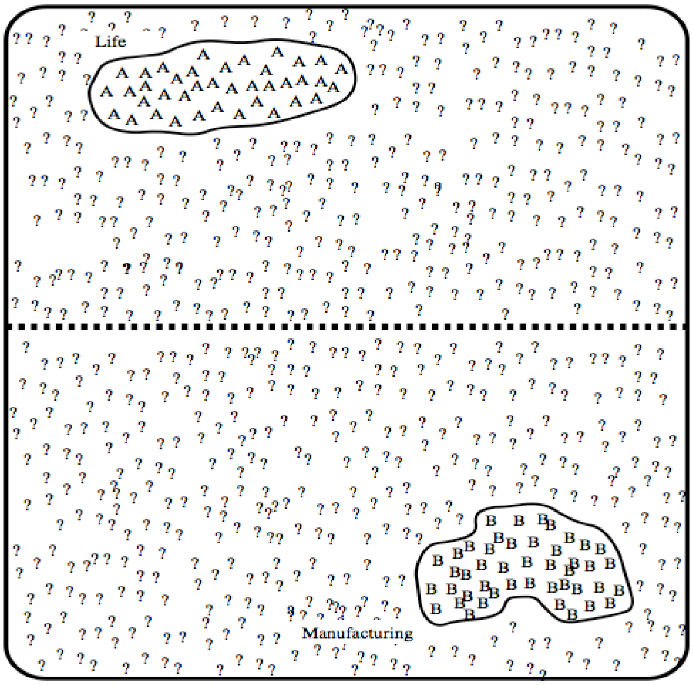
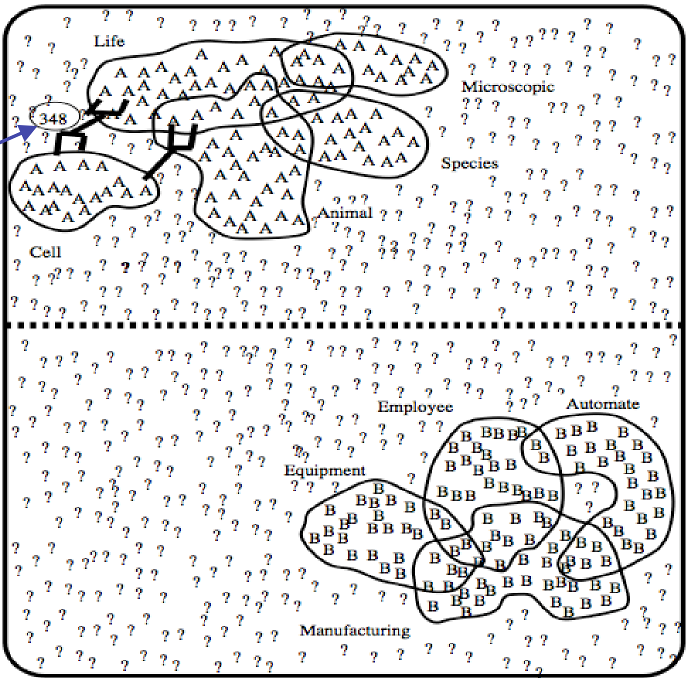


Figure from (Yarowsky, 1995)

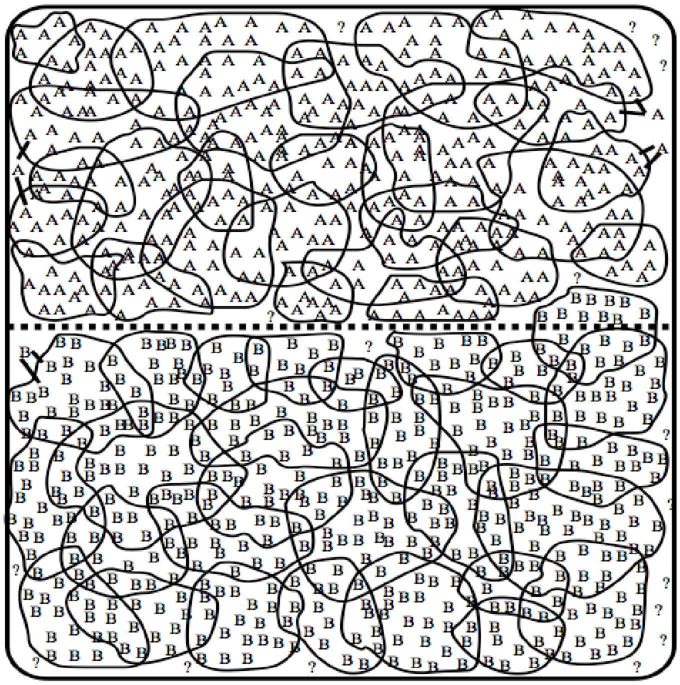
The Yarowsky Algorithm



One sense per discourse constraint can help bootstrapping by injecting new patterns

Figure from (Yarowsky, 1995)

The Yarowsky Algorithm



The Yarowsky Algorithm

- Input: each example x is either labeled L(x) or unlabeled $U^0(x)$
- For $U^0(x)$ a special class label is used for unknown: $\perp$
- Classifier prediction:

  $y = \arg\max_j \Pr(j \mid x, \theta)$ if $\Pr(j \mid x, \theta) >$ threshold ζ

  $y = \perp$ otherwise

```
t = 0
Loop:
for each example x:
    estimate Pr(j | x, θ) using L and U^t(x)
    U^{t+1}(x) = y, where
    y = arg max_j Pr(j | x, θ) if Pr(j | x, θ) > threshold ζ
    y = ⊥ otherwise
If U^{t+1}(x) = U^t(x) then Stop
Else t = t+1 and restart Loop
```

Analysis of the Yarowsky Algorithm

(Abney, 2004)

- It can be shown that some variants of the Yarowsky algorithm optimize either negative log likelihood H or an upper bound on it, called K

- Definition:
  - Empirical labeling distribution $\phi_x(j)$
    - For a labeled data point *x*, it is 1 if *j* is the label of *x* and 0 otherwise
    - For an unlabeled data point *x* it is the uniform distribution over the possible labels.
  - Model's prediction distribution $\pi_x(j) = P(j \mid x , \theta)$
    - θ is the parameter vector of the model

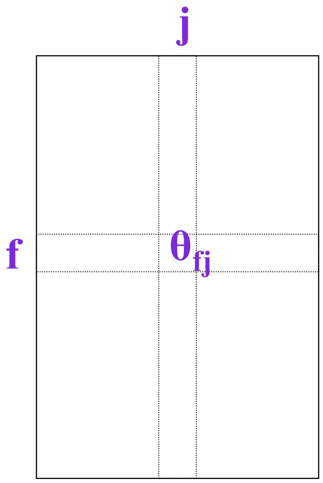
A Modified Algorithm: Y1

- Once an unlabeled example *x* is labeled, it remains labeled (its label is recomputed and may change)
- The threshold ς is eliminated
- The resulting algorithm Y1 optimizes the following objective function:

$$H \equiv \underbrace{\sum_{x \in V} H(\pi_x)}_{\text{Entropy}} + \underbrace{\sum_{x \in X} D(\phi_x \,||\, \pi_x)}_{\text{KL-Divergence}}$$

A Decision List based Model

- Each rule: f → j
  - θ_{fj} : the score of feature f in predicting the label j
  - θ : the parameter vector of the model



- Let F_x to be the set of features of the example x where $|F_x| = m$

- Define the prediction distribution for the example x :

$$\pi_x(j) = \frac{1}{m} \sum_{f \in F_x} \theta_{fj}$$

A Modified Algorithm: DL-1-R

The empirical labeling distribution $\phi_x(j)$:

1. For a labeled data point x, it is 1 if j is the label of x and 0 otherwise
2. For an unlabeled data point x it is the uniform distribution over the possible labels.

```
Initialize N[f, j] = 0, Z[f] = 0
    for all f, j

For each example-label pair
    (x,j)
    For each feature f
        increment N[f, j] and
            Z[f]

For each feature f and label j
    Set $\theta_{fj} = \frac{N[f,j]}{Z[f]}$
```

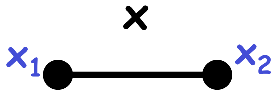
The Objective Function K

$$
\begin{aligned}
H &= -\sum_{x\in X}\sum_{j}\phi_{xj}\log\sum_{g\in F_x}\frac{1}{m}\theta_{gj}\\
&\leq -\sum_{x\in X}\sum_{j}\phi_{xj}\sum_{g\in F_x}\frac{1}{m}\log\theta_{gj}\\
&= \frac{1}{m}\sum_{x\in X}\sum_{g\in F_x}H(\phi_x||\theta_g)\\
&= \frac{1}{m}K
\end{aligned}
$$

Co-Training

(Blum & Mitchell, 1998)

- Instances contain two sufficient sets of features
  - i.e. an instance is x=(x_1,x_2)
  - Each set of features is called a View



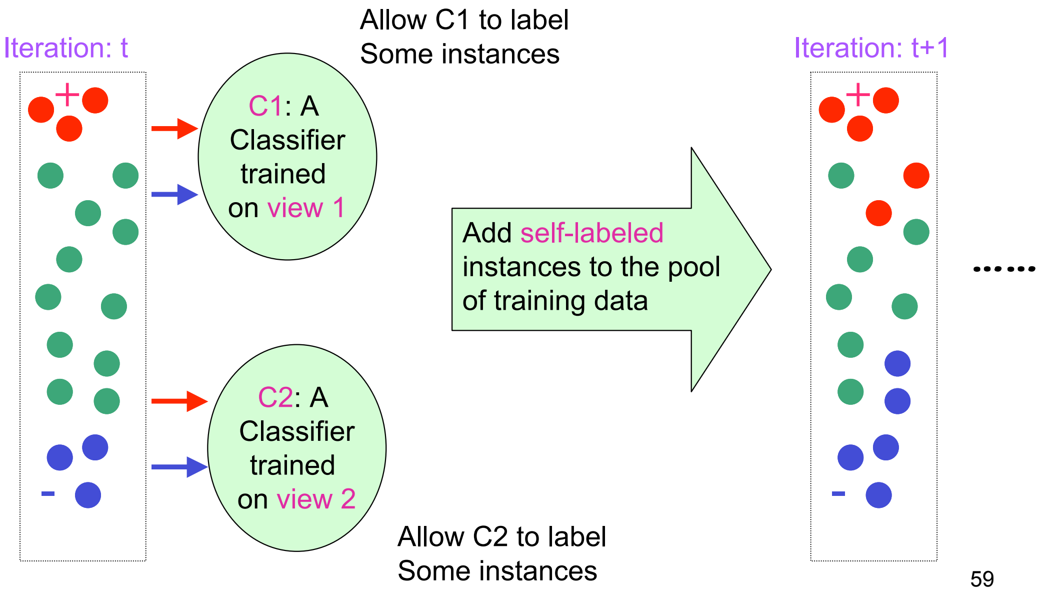
- Two views are independent given the label:

$$
P(x_1|x_2,y) = P(x_1|y)\\
P(x_2|x_1,y) = P(x_2|y)
$$

- Two views are consistent:

$$
\exists c_1, c_2 : c^{opt}(x) = c_1(x_1) = c_2(x_2)
$$

Co-Training



Iteration: t

+

-

C1: A Classifier trained on view 1

Allow C1 to label Some instances

C2: A Classifier trained on view 2

Allow C2 to label Some instances

Add self-labeled instances to the pool of training data

Iteration: t+1

+

-

……

Co-Training

(Blum & Mitchell, 1998)

- An example: build a classifier that categorizes web pages into two classes:
  - + is a course web page and
  - – is not a course web page
- Usual model, build a Naive Bayes classifier

$$P(C = c_k \mid X = \mathbf{x}) = \frac{P(c_k)P(\mathbf{x} \mid c_k)}{P(\mathbf{x})}$$

$$P(\mathbf{x} \mid c_k) = \prod_{x_j \in \mathbf{x}} P(x_j \mid c_k)$$

Co-Training

(Blum & Mitchell, 1998)

- Notice that we can choose to classify each labeled example in two natural ways

- x_1 = text in the hyperlink to the page

<a href="...">CSE 120, Fall semester</a>

- x_2 = text in the web page

<html> ... Assignment #1 </html>

Co-Training

(Blum & Mitchell, 1998)

- Train one NB classifier for x_1 and another NB classifer on x_2
- Baseline model trained on L: Px_1(.) * Px_2(.)
- Co-training model is a modified version of self-training: with two views
- The x_1 classifier produces high confidence output from U and provides it to x_2 (and vice versa)
- On the WebKB dataset, co-training outperforms the baseline;
- One trick used was to ensure that the label distribution did not change when new labeled data was added using co-training
- Instead of labeling entire unlabeled set, a cache was used for computational efficiency

Co-Training

- Assume we are learning binary classifiers F and G for class labels 0 and 1. F and G use two views.

  $a = P(F = 1, G = 1)$ $c = P(F = 1, G = 0)$

  $b = P(F = 0, G = 0)$ $d = P(F = 0, G = 1)$

- Assume that $a * b > c * d + \delta$
- But when G agrees with F on the unlabeled data, F still does not know which label to predict
- We assume we have a weak predictor h for F which uses the labeled data to return the label

$$\alpha = P(h = 0 \mid F = 1)$$
$$\beta = P(h = 1 \mid F = 0)$$

- Theorem (Blum & Mitchell, 1998):

$$P(h = 0 \mid G = 1) + P(h = 1 \mid G = 0) = \frac{a\alpha + d(1 - \beta)}{a + d} + \frac{c(1 - \alpha) + b\beta}{b + c}$$

Intuition behind the Theorem

- Highly confident labeled data points in the view 1 provide *randomly scattered* data points in the view 2 (conditional independence).
- These highly confident labeled data points are noisy, however learning in view 2 is immune to noise.

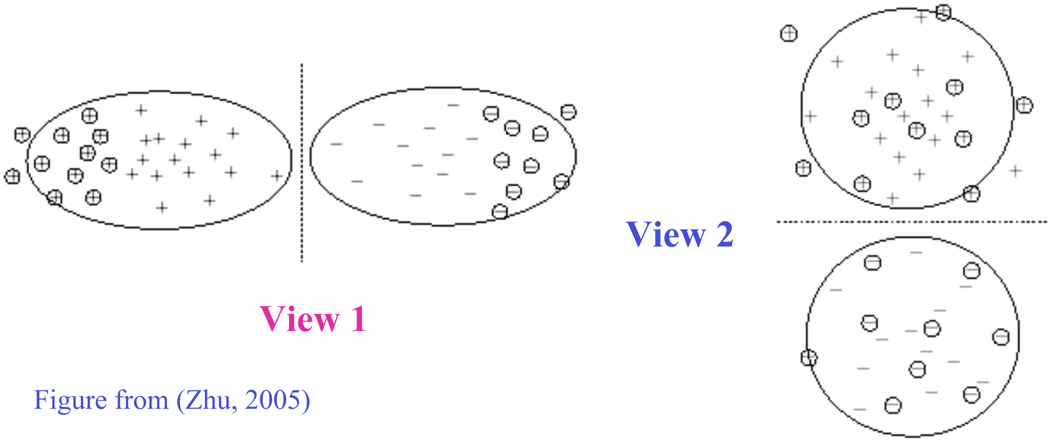


Figure from (Zhu, 2005)

Incremental vs. Iterative and Feature splits

- (Nigam & Ghani, 2000) consider the following table of variants of EM and Bootstrapping

Method	Uses Feature split	Does not use Feature split
Incremental	co-training	self-training
Iterative	co-EM	EM

- Using WebKB they use Naive Bayes in the four settings shown in the table

Incremental vs. Iterative and Feature splits

- (Nigam & Ghani, 2000) experiments on document classification show that using a feature split helped on this task and iterating over the unlabeled set also helped
- The lowest error rate was for co-EM (3.3) vs. co-training (3.7); self-training (5.8) and EM (8.9) did far worse on this task

- They also reported results on choosing conditionally independent feature splits vs. a random feature split
- The theory behind co-training was borne out as the random split did worse than the carefully chosen split

Agreement Maximization

- A side effect of the Co-Training: Agreement between two views.
- What if the agreement becomes the explicit goal?
- Intuitively, it helps by reducing the concept space in views to the consistent concepts.
  - Unlabeled data is used to check consistency
  - Labeled data is used to locate the target concept in the reduced spaces

History of Agreement Maximization

- (Collins & Singer, 1999) suggest a variant of co-training where the agreement of the two classifiers is explicitly optimized. The algorithm is a variant of the AdaBoost algorithm that considers agreement on unlabeled data as part of the objective function.
- (Dasgupta et al, 2001) Provide bound on generalization error of the learned classifier in co-training based on the *empirically measurable quantities*. More analysis is done in (Abney, 2002).
- It formalizes the agreement maximization suggested by (Collins & Singer, 1999)

History of Agreement Maximization

- (Leskes 2005) provides theoretical justification for agreement maximization among multiple views and suggests the Agreement Boost algorithm (belongs to the boosting family of algorithms)

- (Banko & Brill, 2001) also provide a method to maximize agreement among a large number of *bagged* classifiers and show good results on spelling correction when using upto 10 classifiers

- Early versions of the idea of using multiple views in learning from unlabeled data occur in:
  - (Becker and Hinton, 1992; Becker 1995)
  - (de Sa, 1994)
- For a discussion on these and other related cases, see the survey article: (Seeger, 2000)

Analysis of Co-training

(Dasgupta et al, 2001, Abney 2002)

- EM: Maximize likelihood of unlabeled data
- Co-training: Maximize agreement between two views on unlabeled data

- *What can this second objective buy us?*
- **Theorem** (Dasgupta et. al., 2001): The *agreement rate* between two classifiers (under certain strict conditions) is an upper bound on the *error* of either classifier
- Note: the classifiers are partial, i.e. they will say “I do not know” or $\perp$ if they cannot predict the class

Analysis of Co-training

- Assume two views X_1 and X_2 used to build two classifiers F and G, respectively
- Standard assumption of co-training is *view independence*

$P(X_1 = x_1 \mid X_2 = x_2, Y = y) = P(X_1 = x_1 \mid Y = y)$

- But this does not directly tell us about the agreement rate between F and G
- A new assumption: *classifier independence*

$P(F = u \mid G = v, Y = y) = P(F = u \mid Y = y)$

- If view independence holds then classifier independence must hold as well

Analysis of Co-training

- An additional assumption for *non-trivial* cases:

$\min_u P(F = u) > P(F \neq G)$

- To keep things simple, let us consider only classifiers over 2 classes and there's no "I don't know" class $\perp$
- If Y is the true labeling and F and G are classifiers which satisfy classifier independence and are non-trivial then either:

$P(F \neq Y) \leq P(F \neq G)$ or $P(F' \neq Y) \leq P(F \neq G)$

- Since each example is one of two classes: we define F' to be the complement of F

Analysis of Co-training

$P(F \neq Y) \leq P(F \neq G)$ or $P(F' \neq Y) \leq P(F \neq G)$

- This is an upper bound on error: we assume we can pick between F and F' using the labeled data
- Co-training learns upto a *permutation* of predicted labels: the correct labeling from this permutation requires labeled data
- The precision of a classifier F is $P(Y = u \mid F = u)$
- (Abney, 2002) shows that classifier independence implies that if we know the precision of F then we know precision of G
- This can result in trivial agreement between F and G

Greedy Agreement Algorithm

(Abney, 2002)

- Input: Seed rules h_1 and h_2
- **Loop:**
  - ***for*** each atomic rule g
    - $H_2 = h_2 + g$
    - evaluate **cost** of (h_1, H_2)
    - keep lowest cost H_2
  - ***if*** H_2 is worse than h_2 ***then*** quit
  - swap h_1, H_2

Attention shifts to the other classifier

Cost of a pair of Classifiers

$\mathrm{Cost}(h_1, h_2) = ½\,[$ upper bound of $\mathrm{err}_{h2}(h_1)$ +
upper bound of $\mathrm{err}_{h1}(h_2)$]

disagreement $\delta = P(h_1 \neq h_2 \mid h_1, h_2 \neq \perp)$

minor probability $\mu = \min_u P(h_1 = u \mid h_1 \neq \perp)$

$$\mathrm{err}_{h_2}(h_1) \leq \frac{\delta}{(\mu - \delta)}$$

Estimation of the error of h_1 by the help of the classifier in the other view h_2

More PAC Analysis

- (Balcan, Blum & Yang, 2004) try to relax the strong assumptions needed to theoretically analyze the performance of co-training; they also heuristically analyze the error propagation during co-training

- (Balcan & Blum, 2005) provide a PAC-style model for the analysis of learning from labeled and unlabeled data; and discuss the special case of co-training with linear separators

Co-training Experiments

- (Collins & Singer, 1999) proposed the co-boosting algorithm and performed expts on the named entity classification task
- natural feature split: spelling vs. contextual classifier
- using only 7 simple seed rules, unlabeled data was used to achieve 83.1% accuracy;
- although self-training seemed to perform just as well

- (Barzilay & McKeown, 2001) uses co-training to find lexical and syntactic paraphrases
- natural feature split: contextual vs. paraphrase classifier

Co-training Experiments

- (Pierce & Cardie, 2001) proposed a feature split for noun-phrase chunking: one view was a left-context chunker while the other was a right-context chunker; used a NB learner
- the learning curves for co-training was disappointing
- there was no analysis of the feature split so it is hard to tell if the conditions for co-training were satisfied in this case
- problems with noise entering into the labeled data

Co-training Experiments

- (Sarkar, 2001) applied co-training to statistical parsing
- a feature split was obtained by using a statistical parser and a HMM-based SuperTagger
- both learners had to label sentences using the same set of complex lexical descriptions to each word (trees from a Tree-adjoining grammar)
- some knowledge about the unseen trees (a grammar) was assumed in the experiment due to the small size of the seed set
- co-training outperformed simply using the labeled data

Co-training Experiments

- (Steedman et. al., 2003a) also applied co-training to statistical parsing
- a feature split was obtained by using one view as the Collins parser (a CFG-based model) while the other view was an Tree-adjoining grammar statistical parser; the two views were experimentally shown to be distinct
- experiments showed improvement of 2.5% for co-training vs. a decrease of 0.1% for self-training with a 500 word seed set
- the same experimental setup was used for domain adaptation: a small seed set of 100 WSJ sentences was added to a larger (1K) set of Brown corpus sentences and after co-training on WSJ, the parsers were tested on WSJ data
- in this setting, co-training was able to improve f-score from 75.4 to 78.2

Co-training Experiments

- (Steedman et. al., 2003b) is about the method for selection of examples for each view for inclusion into the labeled data
- this paper considers alternative methods for selecting such examples:
  - **above-*n***: the score of an example for each view is greater than or equal to *n*
  - **difference-*n***: score for an example of one view is greater than score of the other by some threshold *n* **(difference-10% performed the best)**
  - **intersection-*n***: an example is in the bottom *n* percent of one view is in the set of the other view's *n* percent highest scoring sentences
- the parameter *n* controls the amount of newly labeled data in each iteration; can be used to deal with noise entering the data
- as in many other co-training papers, an active learning component was added to correct some of the co-training labeled examples added to the labeled set

Co-training Experiments

- (Müller et. al., 2002) apply co-training to the task of co-reference resolution
- base learners used were C4.5 decision trees
- co-reference chains were split up into individual binary classification decisions
- it is not clear if there is a natural feature split that could be exploited in this setting
- (mostly) negative results for co-training

Co-training Experiments

- (Callison-Burch, 2002) proposed a co-training algorithm for statistical machine translation
- the idea is to use multiple languages A, B, C, D each of which translate into English E
- in addition, A, B, C and D are sentence aligned with each other, so that if a sentence from C is found to be accurately translated into English then the corresponding sentences in A, B, and D now have a new labeled parallel text
- expts used the EU corpus and word error rate (WER) improvement was highest for German to English (2.5%)
- noise injected into the labeled set was a problem when large amounts of co-trained data was added

Co-training Experiments

- (Clark, Curran & Osborne, 2003) report on co-training expts for part-of-speech tagging
- using two previously built taggers: TnT and a MaxEnt tagger
- performs an explicit greedy **agreement based** co-training algorithm
- naive co-training (using the whole cache)
- a held-out set was used to measure agreement on addition of newly labeled example(s)
- agreement-based selection is more picky and so can reduce noise in the newly labeled data
- with small seed sets (~500) there was significant improvement, but no significant improvement was seen with large seed sets (all Treebank)

Dealing with Noise

- One common issue that crops up in co-training expts is the issue of noise when a large number of newly labeled examples are added into the training set
- (Goldman & Zhou, 2000) deal with this issue by using hypothesis testing to check if each example if added to the labeled data is likely improve accuracy for the classifier
- (Zhou & Goldman, 2004) use an ensemble of three or more classifiers to vote on whether a newly labeled example should be added into the labeled set

Dealing with Noise

- Note that in the Yarowsky algorithm, we can choose to relabel any unlabeled example or even drop a previously labeled example from U altogether (in some versions of the algorithm)
- Most research in co-training has focused on keeping the best examples from the unlabeled data rather than re-labeling previously seen examples
- This is mainly due to the computational inefficiency caused by re-labeling the entire unlabeled set in each iteration

Data Based Methods

- From bootstrapping methods we now move to methods that use some inherent geometry in the unlabeled data. We call these methods: data based methods
- For many of these methods we represent learning as trying to maximize some objective function
- This maximum is found using standard methods such as gradient descent
- A good objective function tries to minimize error on training data using a loss function
- And tries to find a small model so that it can generalize and so that it does not *overfit*: this is done by adding an additional factor to the function called regularization.

Loss Function

- Goal: finding a *good* classifier, i.e. one which has the minimum expected loss or risk

- Often we build classifiers based on functions
  - Example of a 2-class classifier:

$$c(x) = 1 \quad \text{if} \quad f(x) > 0$$
$$c(x) = -1 \quad \text{otherwise}$$

Loss Function

- *loss(c(x), y, x)* : The penalty induced by assigning class *c(x)* instead of the true class *y* to the instance *x*

- 0-1 loss: $\text{loss}(c(x), y, x) = 1 \quad \text{if } c(x) \neq y$
  $\text{loss}(c(x), y, x) = 0 \quad \text{otherwise}$

- Negative log loss :
  $\text{loss}(c(x), y, x) = -\log P(y \mid c(x), x)$

Regularization

- Expected risk or loss:

$$r = \int loss(f(x), y, x)\, P(x)\, dx$$

- But often P(x) is unknown. Moreover, *y* is not given for all *x*

- The expected risk is upper-bounded by empirical risk plus a regularization term (or complexity term)

Regularization

- Prefers simple functions over complex ones

$$r \leq \Sigma_i \, loss(f(x_i) , y_i , x_i) + \lambda \, \Omega(f)$$

Empirical risk + regularization parameter * Complexity term

- Goal : Find *f* which minimizes the upper-bound expression
- Often *f* is searched in a function class

Example

- Gaussian regularization for a log-linear model

$$\sum_{i=1}^{n} \mathbf{W} \cdot \phi(x_i, y_i) - \log Z - \sum_{k=1}^{m} \frac{\mathbf{W_k}^2}{2\sigma^2}$$

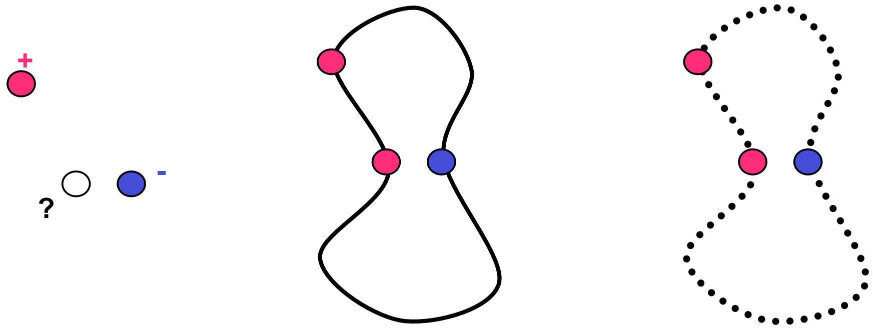
Log likelihood of the log linear model

Gaussian regularization

  - **W** is the parameter vector, and Φ(x,y) is the feature vector
  - Penalizes large weights **W**

- Bayesian interpretation of the Gaussian regularization: It puts a Gaussian prior over the weights **W** and combines it with the log-likelihood to compute the maximum of the posterior distribution.

Data Manifold



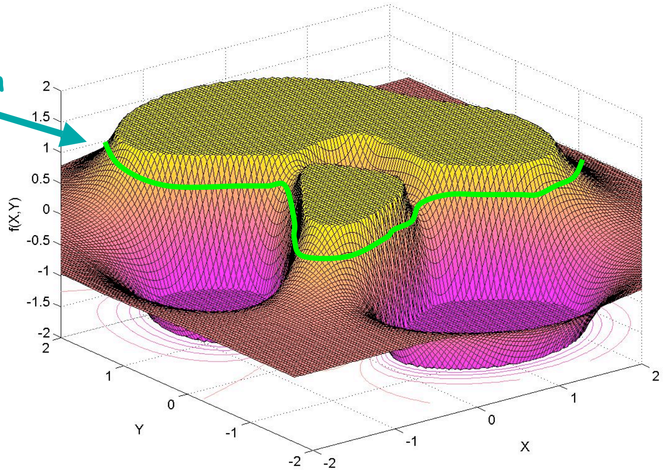
- What is the label?
- Knowing the geometry affects the answer.
  - Geometry changes the notion of similarity.
  - Assumption: Data is distributed on some low dimensional manifold.
- Unlabeled data is used to estimate the geometry.

Smoothness assumption

- Desired functions are smooth with respect to the underlying geometry.
  - Functions of interest do not vary much in high density regions or clusters.
    - Example: The constant function is very smooth, however it has to respect the labeled data.

- The probabilistic version:
  - Conditional distributions $P(y|x)$ should be smooth with respect to the marginal $P(x)$.
    - Example: In a two class problem $P(y=1|x)$ and $P(y=2|x)$ do not vary much in clusters.

A Smooth Function



The decision boundary

- Cluster assumption: Put the decision boundary in low density area.
  - – A consequence of the smoothness assumption.

What is smooth? (Belkin & Niyogi)

- Let $f : \mathcal{M} \rightarrow R$. Penalty at $x \in \mathcal{M}$:

$$\frac{1}{\delta^k} \int_\Delta (f(x) - f(x+\delta))^2 p(x)\, d\delta \approx ||\nabla f||^2 p(x)$$

- Total penalty:

$$\int_\mathcal{M} ||\nabla f||^2 p(x)\, dx$$

- p(x) is unknown, so the above quantity is estimated by the help of unlabeled data:

$$\sum_{i,j} (f(x_i) - f(x_j))^2 W_{ij}$$

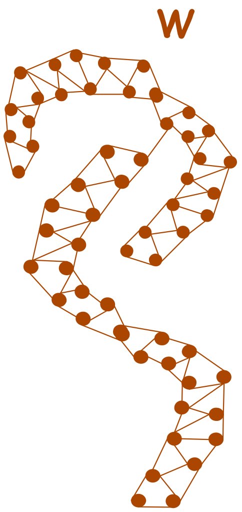


Figure from (Krishnapuram, et. al., 2005)

Manifold Regularization

(Belkin et al, 2004)

Data dependent regularization

$$f^{opt} = \arg\min_{f\in H} \lambda_I ||f||_I^2 + \lambda_k ||f||_k^2 + \frac{1}{l}\sum_i^l (f(x_i) - y_i)^2$$

Smoothness term: Unlabeled data

Function complexity: Prior belief

Fitness to Labeled data

- Where:
  - H is the RKHS associated with kernel k(.,.)
  - Combinatorial laplacian can be used for smoothness term:

$$||f||_I^2 = f^T \cdot (D - W) \cdot f = \sum_{i,j} (f(x_i) - f(x_j))^2 W_{ij}$$

The Representer Theorem

- The Representer theorem guarantees the following form for the solution of the optimization problem:

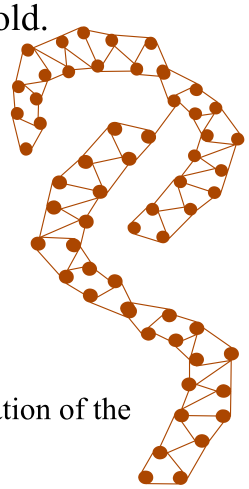
$$f^{opt}(.) = \sum_{i=1}^{l+u} \alpha_i k(x_i, .)$$

Return to SSL for structured…

Harmonic Mixtures

(Zhu & Lafferty 2005)

- Data is modeled by a mixture of Gaussians.
  - Assumption: Look at the mean of Gaussian components, they are distributed on a low dimensional manifold.



- Maximize the objective function:

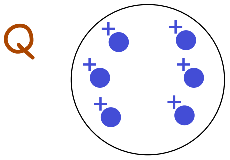
$$\lambda L(\theta) - (1-\lambda)\varepsilon(\theta)$$

  - θ includes mean of the Gaussians and more.
  - $L(\theta)$ is the likelihood of the data.
  - $\varepsilon(\theta)$ is taken to be the combinatorial laplacian.
    - Its interpretation is the energy of the current configuration of the graph.

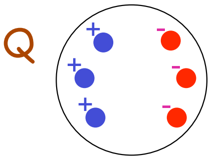
Mutual Information

- Gives the amount of variation of y in a local region Q:

$$I_Q(x,y) = \sum_y \int_Q p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$



- I(x,y) = 0
- Given the label is +, we cannot guess which (x,+) has been chosen (independent).



- I(x,y) = 1
- Given the label is +, we can guess which (x,+) has been chosen.

Information Regularization

(Szummer & Jaakkola 2002)

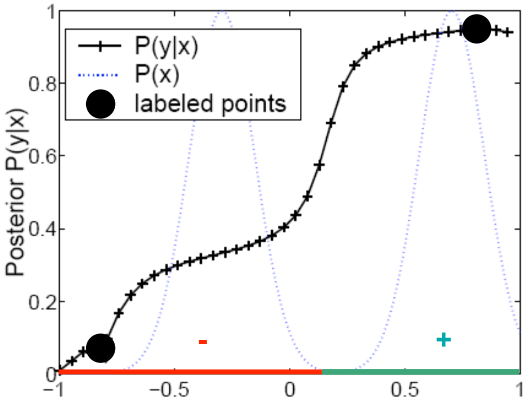
- We are after a good conditional P(y|x).
  - Belief: Decision boundary lays in low density area.
  - P(y|x) must not vary so much in high density area.

- Cover the domain with local regions, the resulting maximization problem is:

$$p^{opt}(y|x) = \arg\max_{p(y|x)}$$

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

Example



- A two class problem (Szummer&Jaakkola)

Return to smoothness

Predict Classifier Structure

(Ando & Zhang 2005)

- Often semi-supervised learning algorithms are not reliable
  - They improve the performance when the labeled data is small but may degrade the performance when the labeled data is large
  - This method does *not* have this deficiency

- The usual approach is to consider a distance measure in the input space, enforce the smoothness of functions w.r.t. the underlying geometry
  - But, what is a good metric?

More intuition on multi-task..

Structural Learning

- First learn the common predictive structure, and then learn the final classifier
  - Example: Classifiers are linear, and share parameter θ:

$$f(x) = (w + \theta \cdot v)^T \cdot x$$

- Generate several auxiliary problems from unlabeled data, learn the corresponding classifiers, and discover the common structure

$$\arg\min_{\theta, f_i} \sum_{i=1}^{m} \left(\sum_{(x,y)\in S_i} \frac{loss(f_i(x),y)}{|S_i|} + r(f_i) \right) + r_0(\theta)$$

Fitness to labeled data

Risk of the classifier, for the linear example: $||w||^2$

Prior belief

Example: Linear Classifiers

- Parameters are found by optimizing the following objective function:

$$\underset{\theta,\{w_i,v_i\}}{\arg\min} \sum_{i=1}^{m}\left(\sum_{j\in S_i}\frac{loss((w_i+\theta^T\cdot v_i)^T\cdot x_j^i, y_j^i)}{|S_i|}\right)+\lambda_i\,||\,w_i\,||$$

Parameters of the auxiliary problems

Common structure parameter vector

Risk of the classifier

- Recall: $f(x) = (w + \theta \cdot v)^T \cdot x$

More on Multi-Task

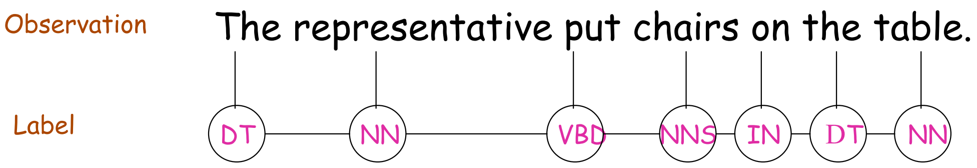
SSL for Structured Labels

- Generative model based (already covered with EM):
  - Hidden Markov Model (HMM)
  - Stochastic Context Free Grammar (SCFG)

- Discriminative model based:
  - Co-Hidden Markov Perceptron
  - Semi-Structured Support Vector Machines (SSVM)
  - Co-Structured SVM (Co-SSVM)
  - Semi-Kernel Conditional Random Fields (KCRF)

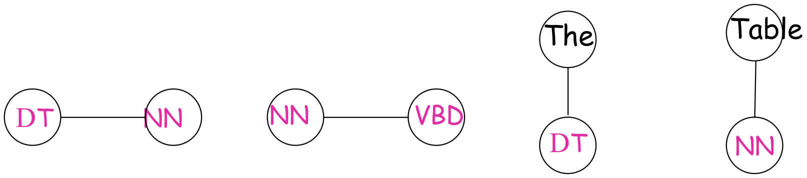
Structured Prediction

- Example: Part-of-speech tagging:

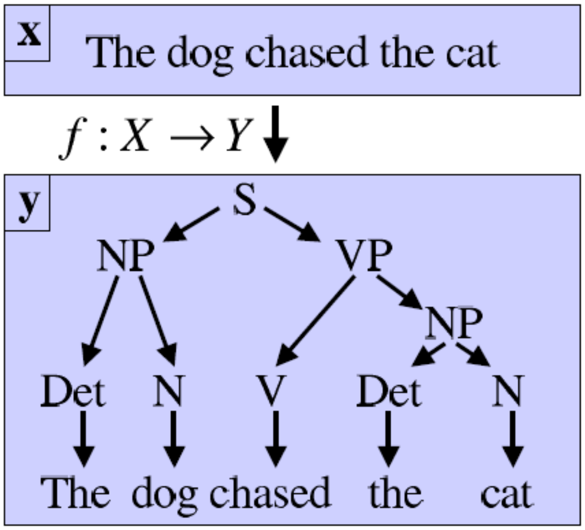

Observation The representative put chairs on the table.

Label DT NN VBD NNS IN DT NN

- The input is a complex object as well as its label.
  - Input-Output pair (x,y) is composed of simple parts.
  - Example: Label-Label and Obs-Label edges:



Structured Prediction: Parsing


x The dog chased the cat

$f : X \rightarrow Y$

y

Figure from (Tsochantaridis et al 2004)

Scoring Function

- For a given x, consider the set of all its candidate labelings as Y_x
  - For a sentence x, consider all of the parse trees Y_x which have x at their leaves
- We are interested in a function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- Instead, learn a scoring function $S : \mathcal{X} \times \mathcal{Y} \rightarrow R$ over the input-output space

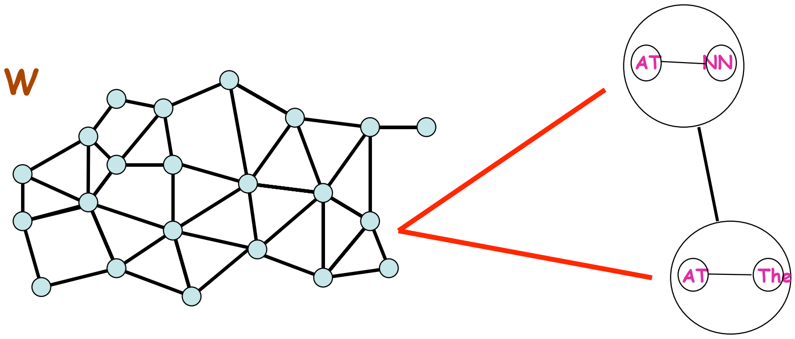
$$y = \arg\max_{y' \in Y_x} S(x, y')$$

- In general, decoding (doing the above argmax) is **intractable** except for special cases

More on Scoring Function

Manifold of "simple parts"

(Altun et al, 2005)



- Construct d-nearest neighbor graph on all parts seen in the sample.
  - For unlabeled data, put all parts for each candidate.
- Belief: f(.) is smooth on this graph (manifold).

More on Discriminative Structured…

SSL for Structured Labels: Semi-KCRF and Semi-SSVM

- The final maximization problem:

Data dependent regularization

$$\arg\min_{f\in\mathcal{H}} \sum_i loss(f(x_i), y_i) + \lambda_k ||f||_k + \lambda_I \sum_{r,r'} W_{r,r'}(f(r) - f(r'))^2$$

Fitness to Labeled data

Function complexity: Prior belief

Smoothness term: Unlabeled data

- The Representer theorem:

$$f(.) = \sum_{r\in R(S)} \alpha_r k(r,.)$$

  - R(S) is all the simple parts of labeled and unlabeled instances in the sample.
  - Note that f(.) is related to $\alpha = (\alpha_1, ..., \alpha_{R(S)})$

Modified problem

- Plugging the form of the best function in the optimization problem gives:

$$\arg\min_\alpha \sum_i loss(f_\alpha(x_i), y_i) + \alpha^T \cdot Q \cdot \alpha$$

  - Where Q is a constant matrix.
  - By introducing slack variables ε_i:

$$\arg\min_\alpha \sum_i \varepsilon_i + \alpha^T \cdot Q \cdot \alpha$$

Subject to

$$\forall i, loss(f_\alpha(x_i), y_i) \leq \varepsilon_i$$

Modified problem(cont'd)

- Loss function: $\arg\min_\alpha \sum_i \varepsilon_i + \alpha^T \cdot Q \cdot \alpha$

  Subject to

  $\forall i, loss(f_\alpha(x_i), y_i) \leq \varepsilon_i$

  - SVM: $loss(f_\alpha(x), y) = \max_{y' \in Y_x} \Delta(x, y, y') + S_\alpha(x, y') - S_\alpha(x, y)$

    Hamming distance

  - CRF: $loss(f_\alpha(x), y) = -S_\alpha(x, y) + \log \sum_{y' \in Y_x} \left(\exp S_\alpha(x, y') \right)$

  - Note that an α vector gives the f(.) which in turn gives the scoring function S(x,y). We may write $S_\alpha(x,y)$.

More ...

Conclusions

- We reviewed some important recent works on SSL:
- Classifier based methods
  - EM, Stable mixing of Complete and Incomplete Information
  - Self-training, The Yarowsky Algorithm, Co-training
- Data based methods
  - Manifold Regularization, Harmonic Mixtures, Information Regularization
  - Learning Predictive Structure from Multiple Tasks
- SSL for structured prediction
  - EM for HMMs and PCFGs } **Generative Models**
  - Semi-KCRF, Semi-SSVM
  - Co-SSVM, Co-HM Perceptron } **Discriminative Models**

Conclusions

- Different learning methods for SSL are based on different assumptions
  - Cluster assumption, View-independence assumption, ...
  - Fulfilling these assumptions is crucial for the guaranteed success of the methods

- SSL for structured domains is an exciting area of research

- SSL is related to Active Learning and Sample Selection methods

Thank You

Some Preliminaries

Generative Models:

$$P(C,X \mid \theta) = P(X \mid \theta) \cdot P(C \mid X,\theta)$$

Example of a Generative Model: Naive Bayes

$$P(C = c_k \mid X = \mathbf{x}) = \frac{P(c_k)P(\mathbf{x} \mid c_k)}{P(\mathbf{x})}$$

$$P(\mathbf{x} \mid c_k) = \prod_{x_j \in \mathbf{x}} P(x_j \mid c_k)$$

Discriminative Models:

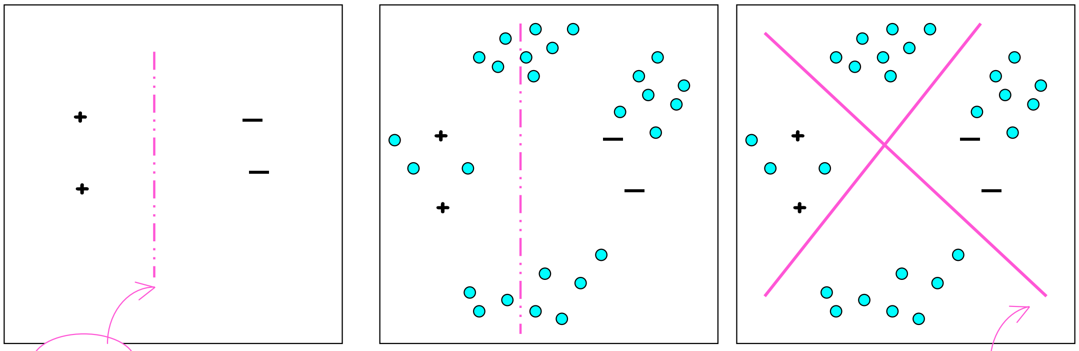
$$P(C,X \mid \theta) = P(X) \cdot P(C \mid X,\theta)$$

Back ...

Example of a Discriminative Model: SVM

Intuition in Large-Margin Setting

(Zhang & Oles, 2000)



SVM

Labeled data only

Right vs. Wrong Margin

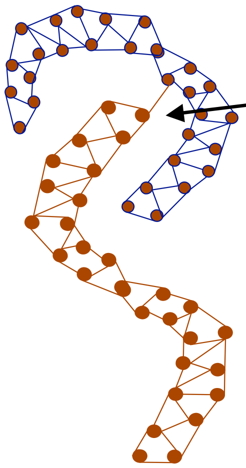
Training a Support Vector Machine (SVM), input *x*, label = {+,-}

Back to slide ...

Graph Mincuts

Labeled Examples: +1

(source)



use edge weights to compute a min-cut

Labeled Examples: -1

(sink)

Back ...

Weak Classifier Dependence

- Classifier independence

$P(F = u \mid G = v, Y = y) = P(F = u \mid Y = y)$

- d_y: Deviation from Classifier independence:

$$d_y = (1/2)\ (\ |P(F = u \mid G = v, Y = y) - P(F = u \mid Y = y)| \\ + |P(F = v \mid G = u, Y = y) - P(F = v \mid Y = y)|\ )$$

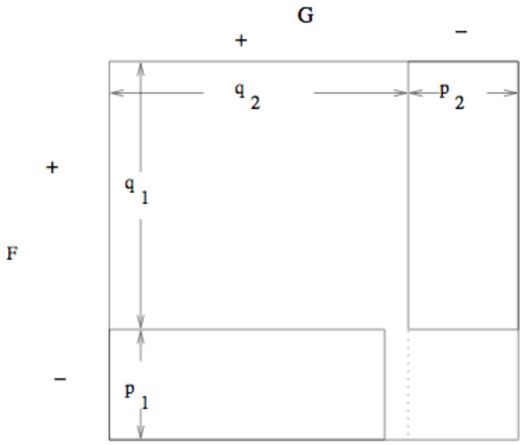
- (Abney, 2002) showed that instead of classifier independence we can use a weaker version to show the same upper bound on error:

$P(F \neq Y) \leq P(F \neq G)$ or $P(F' \neq Y) \leq P(F \neq G)$

Back ...

Weak Classifier Dependence

(Abney, 2002)



Positive Correlation, Y = +

$p_1 = \min_u P(F = u \mid Y = y)$

$p_2 = \min_u P(G = u \mid Y = y)$

- By non-triviality, p1 and p2 must be > 0.5
- $q_1 = 1 - p_1$

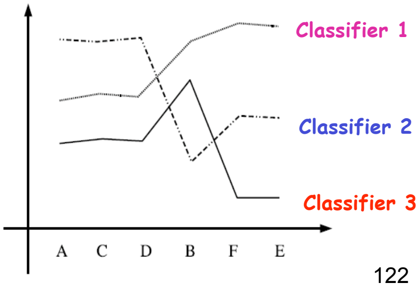
- F and G are weakly dependent if:

$$d_y \leq p_2 \frac{q_1 - p_1}{2 p_1 q_1}$$

Finding Good Structure

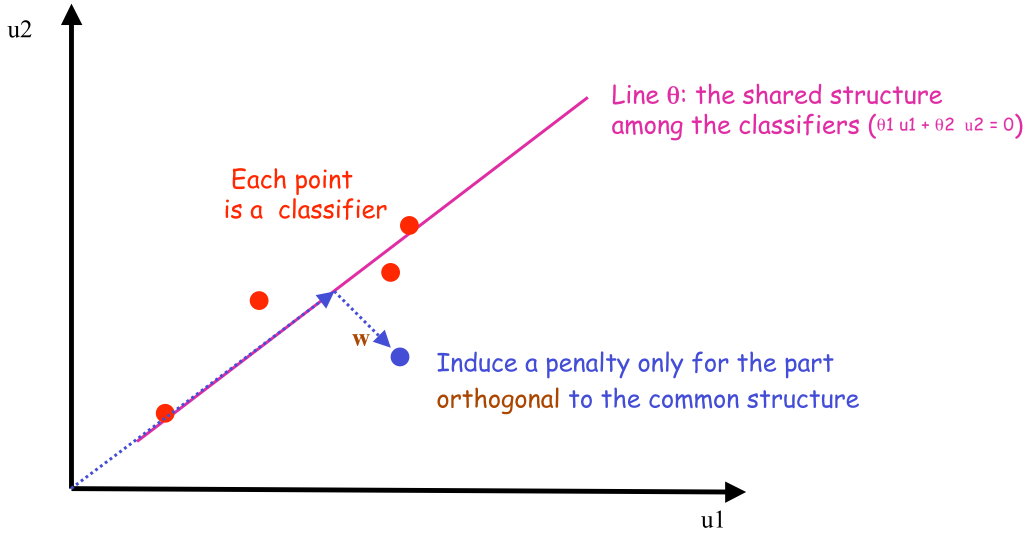
- There are several relevant learning tasks (assumption)
  - Good classifiers are similar to each other, or share a common predictive structure.

- Example: Without knowing the metric inherent to the input space, a good classifier should have:
  - Similar values at **A,C,D**
  - Similar values at **F,E**



Example from
(Ando & Zhang 2005)

Intuition in the Classifier Space



Line θ: the shared structure among the classifiers (θ1 u1 + θ2 u2 = 0)

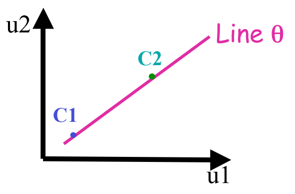
Each point is a classifier

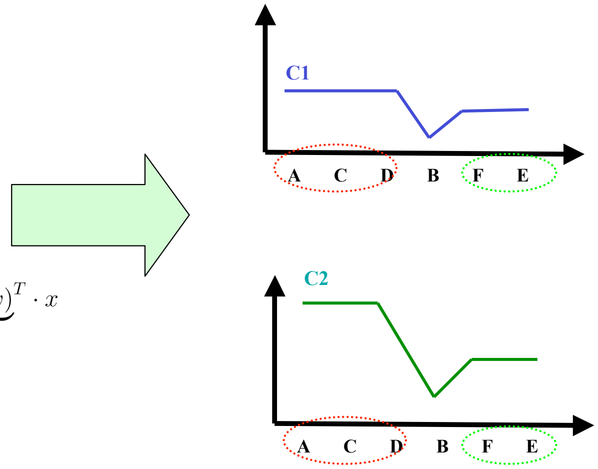
Induce a penalty only for the part orthogonal to the common structure

Linear classifier: $f(x) = \underbrace{(w + \theta \cdot v)}_{u}^{T} \cdot x$

Classifier Space vs Input Space

- Key idea: Classifiers close to the shared structure, partition (cluster) the input space roughly the same





Linear classifier: $f(x) = \underbrace{(w + \theta \cdot v)}_{u}^{T} \cdot x$

Return to structural learning

Structural Learning for NLP

- In Named Entity Chunking task, unlabeled data can be used to generate lots of auxiliary problems: Predict the current word based on other features in the current position
  - Auxiliary problem 1: predict word1
  - Auxiliary problem 2: predict word2
  - …
- (Ando & Zhang 2005) have done several experiments on text chunking and the results are promising.
  - It shows improvement on CoNLL'00 syntactic chunking and CoNLL'03 named entity chunking tasks compared with the best previous systems

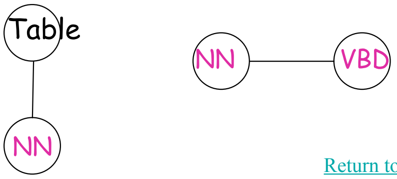
Return to structured label..

Scoring Function

- Assume S(x,y) can be written as the sum of scores for each simple part:

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

  - R(x,y) is the set of simple parts for (x,y).
- Tagging example: Total score is the sum of scores of label-label and observation-label interactions (parts)



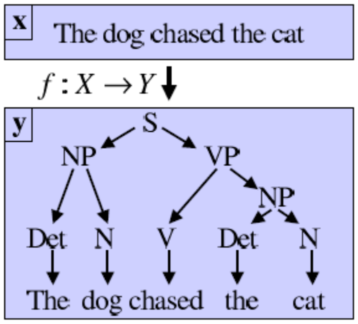
- How to find f(.)?

Return to manifold of parts

Input-Output Feature Space

- Φ(x,y) maps an input-output pair to a fixed dimensional feature space R^d (d is the dimension)

- Parsing example:



$$\Phi(\mathbf{x},\mathbf{y}) = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \\ \vdots \\ 0 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{array}{c} S \rightarrow NP\, VP \\ S \rightarrow NP \\ NP \rightarrow Det\ N \\ VP \rightarrow V\ NP \\ \\ Det \rightarrow dog \\ Det \rightarrow the \\ N \rightarrow dog \\ V \rightarrow chased \\ N \rightarrow cat \end{array}$$

Example from
Tsochantaridis et al 2004

  – Here d is the number of rules in the grammar

HM Perceptron

(Altun et al 2003, Collins 2002)

- Primal formulation:

$$S(x,y) = \mathbf{w} \cdot \Phi(x,y)$$

- Dual formulation:

Training points

$$S(x,y) = \sum_i \sum_{y'} \alpha_i(y') \times [\Phi(x_i, y') \cdot \Phi(x,y)]$$

  – Kernel functions can be used to compute the inner products without explicitly mapping points to the feature space

- Training:
  – If the prediction is $\hat{y}$ instead of the correct y_i do the following until convergence

$$\alpha_i(y_i) = \alpha_i(y_i) + 1 \qquad \alpha_i(\hat{y}) = \alpha_i(\hat{y}) - 1$$

SSL for Structured Labels: Co-HM Perceptron

- We are looking for a good linear scoring function which *separates* the correct label from wrong labels for each training example
  - Here there is not any notion of margin, just a separator!

$$S(x, y) = \mathbf{w} \cdot \Phi(x, y)$$

$$y = \arg\max_{y' \in Y_x} S(x, y')$$

- In the training phase, each classifier uses the prediction of the other view for each unlabeled instance.

Co-HM Perceptron

(Brefeld et al 2005)

- In each view, use the prediction of the classifier in the other view for each unlabeled instance

- Training in view 1:
  - Labeled instance (x_i,y_i) has caused a mistake:

$$\alpha_i^1(y_i) = \alpha_i^1(y_i) + 1 \qquad \alpha_i^1(\hat{y}) = \alpha_i^1(\hat{y}) - 1$$

  - Unlabeled instance x_i has caused a disagreement:

$$\alpha_i^1(\hat{y}^2) = \alpha_i^1(\hat{y}^2) + C_u \qquad \alpha_i^1(\hat{y}) = \alpha_i^1(\hat{y}) - C_u$$

  - $\hat{y}^2$ is the prediction of the view 2.
  - $0 \leq C_u \leq 1$ controls the influence of an unlabeled instance

- Training in view 2 is similar to the above

Experiments

- (Brefeld & Scheffer 2006) applied Co-SSVM to the named entity recognition (NER) and parsing tasks.
- (Brefeld et al 2005) applied Co-SSVM and Co-HM Perceptron to the named entity recognition task.
- In the above papers, random splitting of features into two views results in a good performance for NER.
- Better performance of Co-SSVM and Co-HM Perceptron compared to their single view counterparts comes at the cost of longer training time
  - Co-SSVM scales quadratically and Co-HM Perceptron scales linearly in the number of unlabeled instances

Back ...

SSL for Structured Labels: Co-Structured SVM

- Multi-view learning methods naturally allow the inclusion of unlabeled data in discriminative learning
- In Co-SSVM, there are two views each of which has its own way of converting Input-Output (x,y) to features: $\Phi^1(x,y)$ and $\Phi^2(x,y)$
  - The scoring function is a linear model for each view:

$$S(x,y) = \mathbf{w} \cdot \Phi(x,y)$$

$$y = \arg\max_{y' \in Y_x} S(x,y')$$

SSL for Structured Labels: Co-Structured SVM

- In each view an SSVM is learnt, the goal is to maximize the *agreement* between these two SSVMs

- The final scoring function:

$$S(x,y) = \underbrace{S_1(x,y)}_{\text{View 1}} + \underbrace{S_2(x,y)}_{\text{View 2}}$$

Structured SVM with 0/1 loss

- Optimization problem of SSVM with hard constraints:

$$\min_{\mathbf{w}} ||\mathbf{w}||^2$$

Subject to :

$$\forall_{i=1}^{n}, \forall y' \neq y_i : \mathbf{w} \cdot \Phi(x_i, y_i) - \mathbf{w} \cdot \Phi(x_i, y') \geq 1$$

- Optimization problem of SSVM with soft constraints:

Complexity term

Upper bound on Empirical error

$$\min \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \varepsilon_i$$

Subject to :

$$\forall_{i=1}^{n}, \forall y' \neq y_i : \mathbf{w} \cdot \Phi(x_i, y_i) - \mathbf{w} \cdot \Phi(x_i, y') \geq 1 - \varepsilon_i$$

$$\forall_{i=1}^{n} : \varepsilon_i \geq 0$$

SSVM with arbitrary loss

(Tsochantaridis et al 2004)

$$\min \tfrac{1}{2}||\mathbf{w}||^2 + C\textstyle\sum_{i=1}^{n} \varepsilon_i$$

Complexity term

Upper bound on empirical error

Subject to :

$$\forall_{i=1}^{n}, \forall y' \neq y_i : \mathbf{w} \cdot \Phi(x_i, y_i) - \mathbf{w} \cdot \Phi(x_i, y') \geq 1 - \tfrac{\varepsilon_i}{\Delta(y_i, y')}$$

Loss function

$$\forall_{i=1}^{n} : \varepsilon_i \geq 0$$

- However in max-margin markov networks (M^{3}N), the margin is rescaled: (Taskar et al 2003)

$$\forall_{i=1}^{n}, \forall y' \neq y_i : \mathbf{w} \cdot \Phi(x_i, y_i) - \mathbf{w} \cdot \Phi(x_i, y') \geq \Delta(y_i, y') - \varepsilon_i$$

Measuring the Agreement

- Agreement of the two SSVMs on an unlabeled point is expressed by:

$$S^v(x_i, \hat{y}_i^{\bar{v}}) - \max_{y' \neq \hat{y}_i^{\bar{v}}} S^v(x_i, y') = \gamma_i^v \geq 1$$

Prediction of the other view

- We pretend the prediction of the SSVM in the other view is correct, and
- Expect the SSVM in the current view to produce the same label with enough confidence.

Co-Structured SVM

(Brefeld et al 2006)

$$\min \frac{1}{2}||\mathbf{w}||^2 + C\left[\sum_{i=1}^{n} \varepsilon_i + C_u \sum_{j=n+1}^{n+m} \min\{1, \gamma_i^{\bar{v}}\}\varepsilon_i\right]$$

Subject to :

Labeled : $\forall_{i=1}^{n}, \forall y' \neq y_i : \mathbf{w} \cdot \Phi(x_i, y_i) - \mathbf{w} \cdot \Phi(x_i, y') \geq 1 - \frac{\varepsilon_i}{\Delta(y_i, y')}$

Unlabeled : $\forall_{i=n+1}^{n+m}, \forall y' \neq y_i^{\bar{v}} : \mathbf{w} \cdot \Phi(x_i, y_i^{\bar{v}}) - \mathbf{w} \cdot \Phi(x_i, y') \geq 1 - \frac{\varepsilon_i}{\Delta(y_i^{\bar{v}}, y')}$

$\forall_{i=1}^{n+m} : \varepsilon_i \geq 0$

Back ...