EM for Bayes Nets

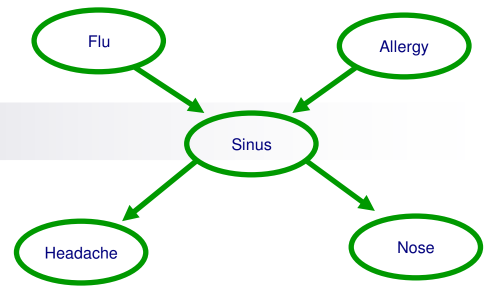
Machine Learning – 10701/15781

Carlos Guestrin

Carnegie Mellon University

April 17th, 2006

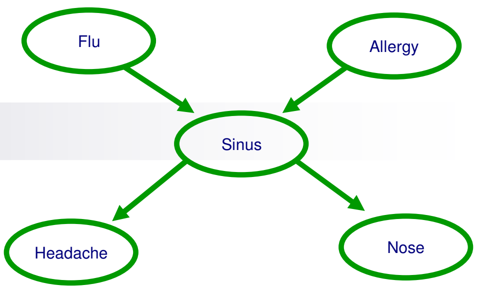
Data likelihood for BNs



- Given structure, log likelihood of fully observed data:

$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G})$$

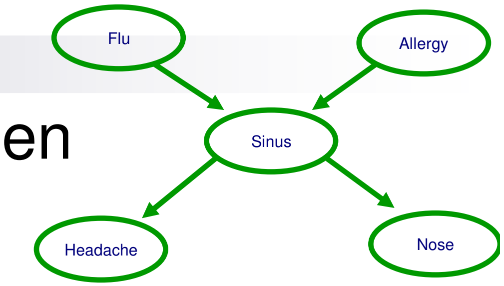
Marginal likelihood



- What if S is hidden?

$$\log P(\mathcal{D} \mid \theta_{\mathcal{G}}, \mathcal{G})$$

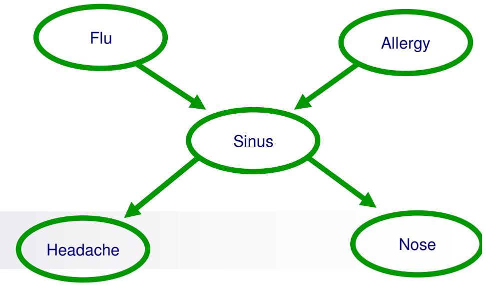
Log likelihood for BNs with hidden data



- Marginal likelihood – **O** is observed, **H** is hidden

$$\ell(\theta : \mathcal{D}) = \sum_{j=1}^{m} \log P(\mathbf{o}^{(j)} \mid \theta)$$
$$= \sum_{j=1}^{m} \log \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{o}^{(j)} \mid \theta)$$

E-step for BNs

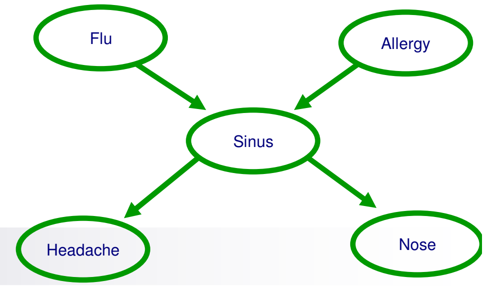


- E-step computes probability of hidden vars **h** given **o**

$$Q^{(t+1)}(\mathbf{x} \mid \mathbf{o}) \; = \; P(\mathbf{x} \mid \mathbf{o}, \theta^{(t)})$$

- Corresponds to inference in BN

The M-step for BNs

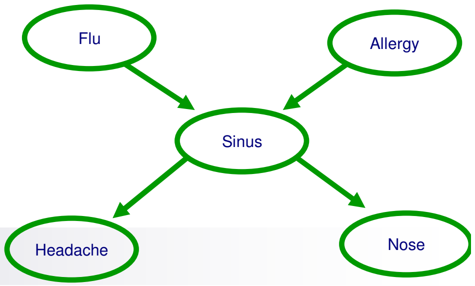


- Maximization step:

$$\theta^{(t+1)} \leftarrow \arg\max_{\theta} \sum_{\mathbf{x}} Q^{(t+1)}(\mathbf{h} \mid \mathbf{o}) \log P(\mathbf{h}, \mathbf{o} \mid \theta)$$

- Use expected counts instead of counts:
  - If learning requires Count(**h**,**o**)
  - Use $E_{Q(t+1)}$[Count(**h**,**o**)]

M-step for each CPT



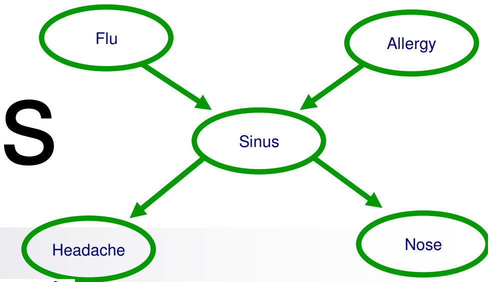
- M-step decomposes per CPT
  - Standard MLE:

$$P(X_i = x_i \mid \mathbf{Pa}_{X_i} = \mathbf{z}) = \frac{\text{Count}(X_i = x_i, \mathbf{Pa}_{X_i} = \mathbf{z})}{\text{Count}(\mathbf{Pa}_{X_i} = \mathbf{z})}$$

  - M-step uses expected counts:

$$P(X_i = x_i \mid \mathbf{Pa}_{X_i} = \mathbf{z}) = \frac{\text{ExCount}(X_i = x_i, \mathbf{Pa}_{X_i} = \mathbf{z})}{\text{ExCount}(\mathbf{Pa}_{X_i} = \mathbf{z})}$$

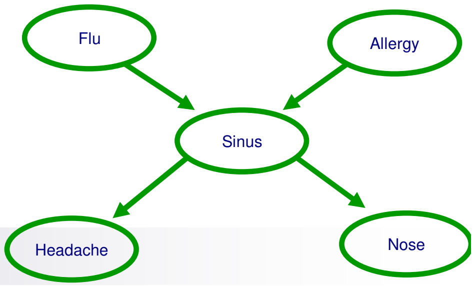
Computing expected counts



$$P(X_i = x_i \mid \mathbf{Pa}_{X_i} = \mathbf{z}) = \frac{\text{ExCount}(X_i = x_i, \mathbf{Pa}_{X_i} = \mathbf{z})}{\text{ExCount}(\mathbf{Pa}_{X_i} = \mathbf{z})}$$

- M-step requires expected counts:
  - For a set of vars **A**, must compute ExCount(**A**=**a**)
  - Some of **A** in example *j* will be observed
    - denote by $\mathbf{A_O} = \mathbf{a_O}^{(j)}$
  - Some of **A** will be hidden
    - denote by $\mathbf{A_H}$
- Use inference (E-step computes expected counts):
  - $\text{ExCount}^{(t+1)}(\mathbf{A_O} = \mathbf{a_O}^{(j)}, \mathbf{A_H} = \mathbf{a_H}) \leftarrow \text{P}(\mathbf{A_H} = \mathbf{a_H} \mid \mathbf{A_O} = \mathbf{a_O}^{(j)}, \theta^{(\mathbf{t})})$

Data need not be hidden in the same way



- When data is fully observed
  - A data point is
- When data is partially observed
  - A data point is
- But unobserved variables can be different for different data points
  - e.g.,
- Same framework, just change definition of expected counts
  - $\text{ExCount}^{(t+1)}(\mathbf{A_O} = \mathbf{a_O}^{(j)}, \mathbf{A_H} = \mathbf{a_H}) \leftarrow \text{P}(\mathbf{A_H} = \mathbf{a_H} \mid \mathbf{A_O} = \mathbf{a_O}^{(j)}, \theta^{(\mathbf{t})})$

What you need to know

- EM for Bayes Nets
- E-step: inference computes expected counts
  - Only need expected counts over X_i and $\mathbf{Pa}_{x_i}$
- M-step: expected counts used to estimate parameters
- Hidden variables can change per datapoint

- Use labeled and unlabeled data $\rightarrow$ some data points are complete, some include hidden variables

Reading:
Blum & Mitchell 1999
(see class website)

Co-Training for Semi-supervised learning

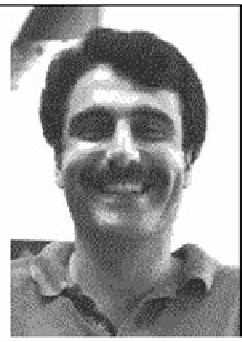
Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

April 17th, 2006

Redundant information

Professor Faloutsos

my advisor



U.S. mail address:
Department of Computer Science
University of Maryland
College Park, MD 20742
(97-99: on leave at CMU)
Office: 3227 A.V. Williams Bldg.
Phone: (301) 405-2695
Fax: (301) 405-6707
Email: christos@cs.umd.edu

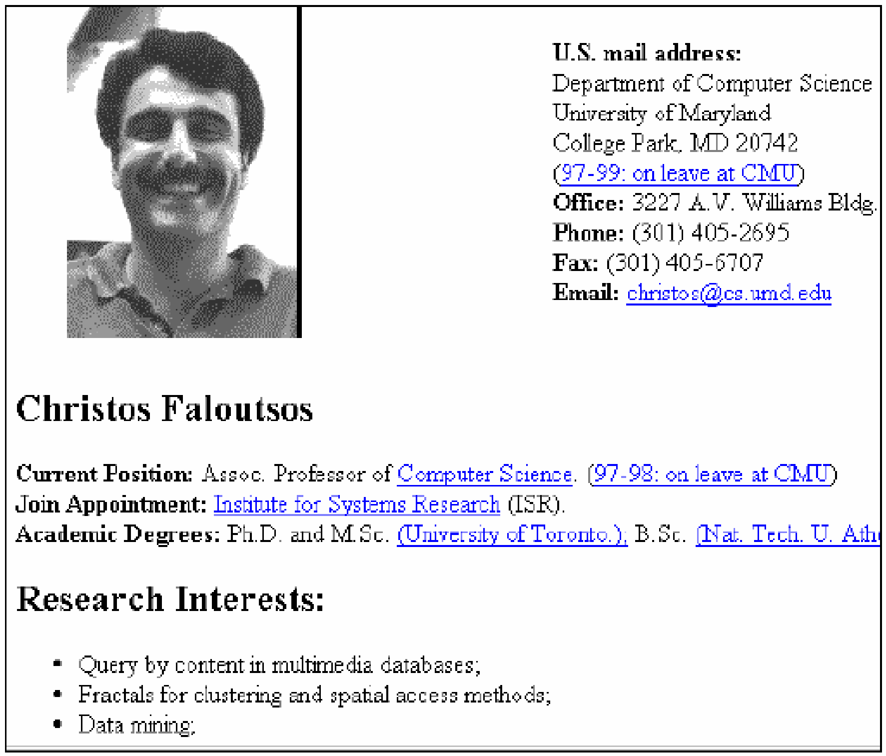
Christos Faloutsos

Current Position: Assoc. Professor of Computer Science. (97-98: on leave at CMU)
Join Appointment: Institute for Systems Research (ISR).
Academic Degrees: Ph.D. and M.Sc. (University of Toronto.), B.Sc. (Nat. Tech. U. Ath

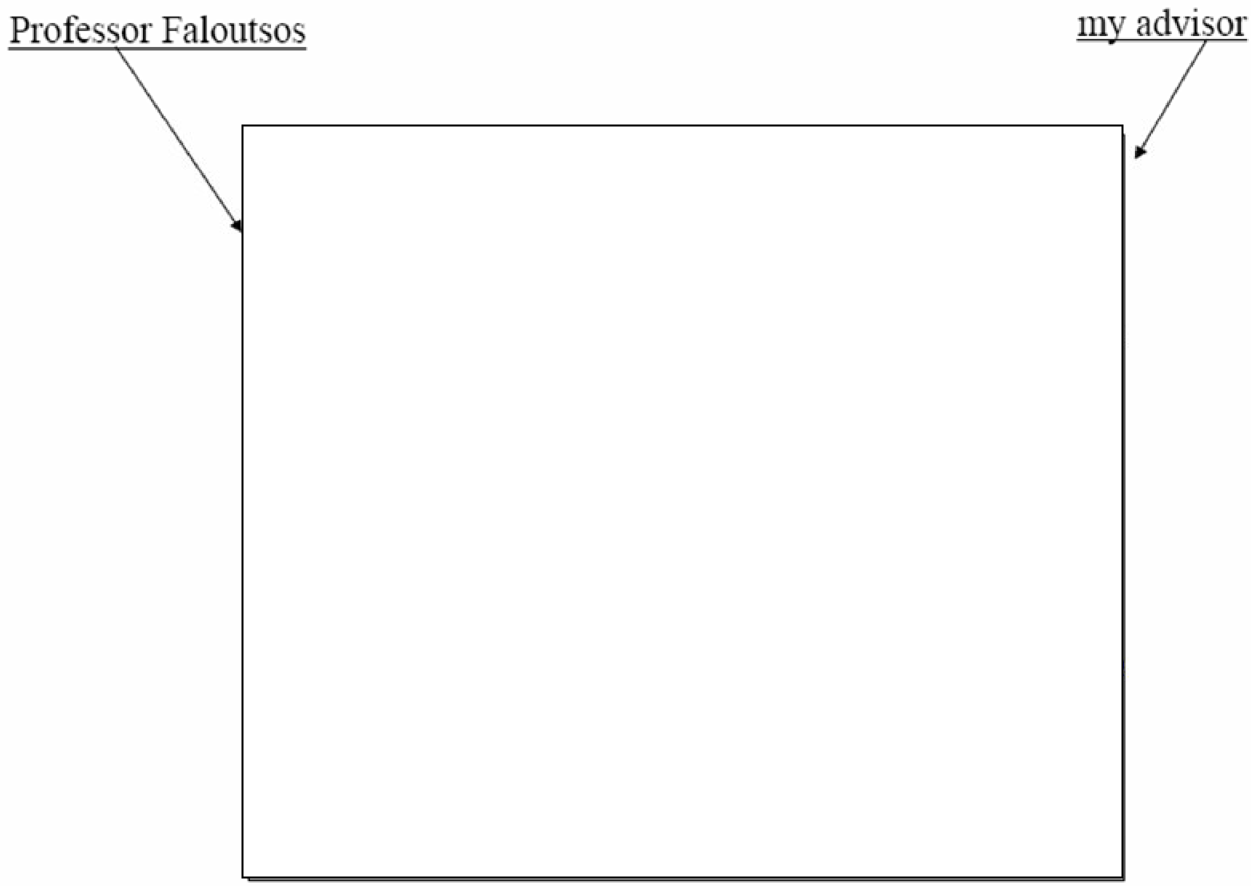
Research Interests:

- Query by content in multimedia databases;
- Fractals for clustering and spatial access methods;
- Data mining;

Redundant information – webpage text

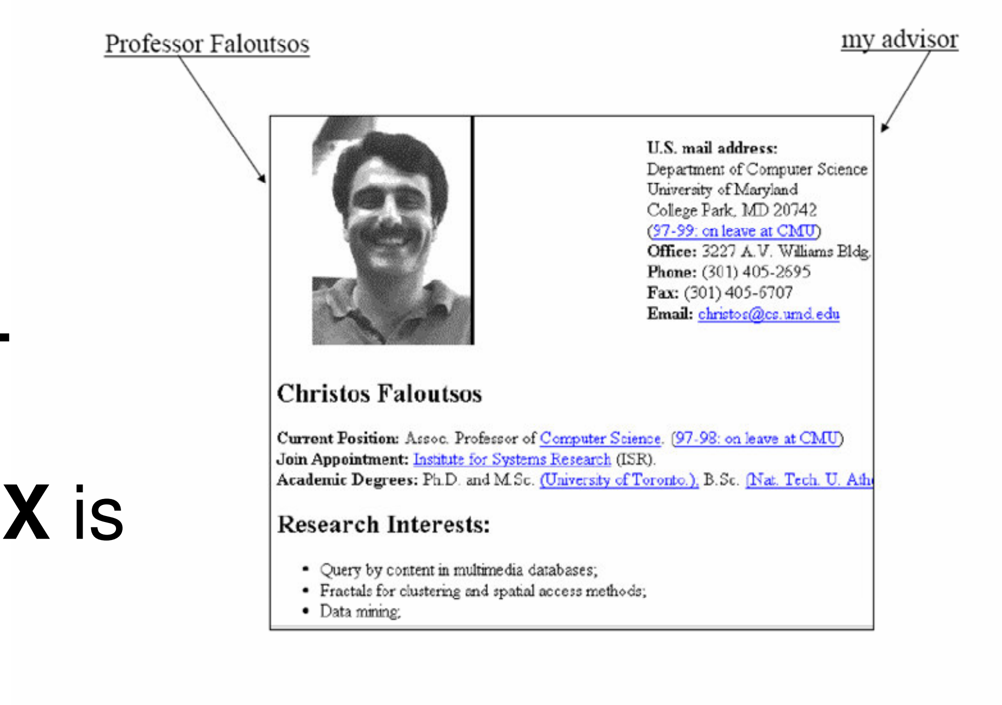


Redundant information – anchor text for hyperlinks



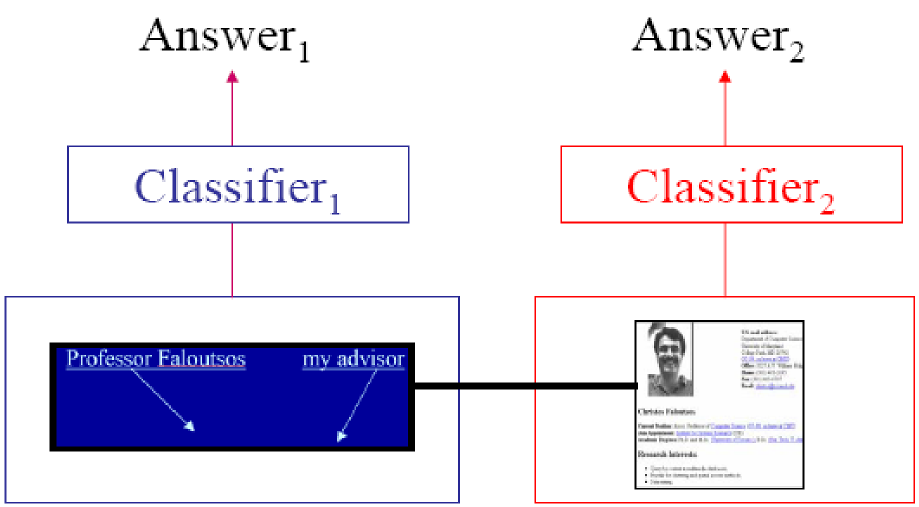
Exploiting redundant information in semi-supervised learning

- Want to predict Y from features **X**
  - f(**X**) $\mapsto$ Y
  - have some labeled data **L**
  - lots of unlabeled data **U**
- Co-training assumption: **X** is very expressive
  - **X** = (**X**$_1$,**X**$_2$)
  - can learn
    - $g_1(\mathbf{X}_1) \mapsto Y$
    - $g_2(\mathbf{X}_2) \mapsto Y$



Co-Training

- Key idea: $Classifier_1$ and $Classifier_2$ must:
  - Correctly classify labeled data
  - **Agree** on unlabeled data



Co-Training Algorithm
[Blum & Mitchell '99]

Given: labeled data L,

unlabeled data U

Loop:

Train g1 (hyperlink classifier) using L

Train g2 (page classifier) using L

Allow g1 to label *p* positive, *n* negative examps from U

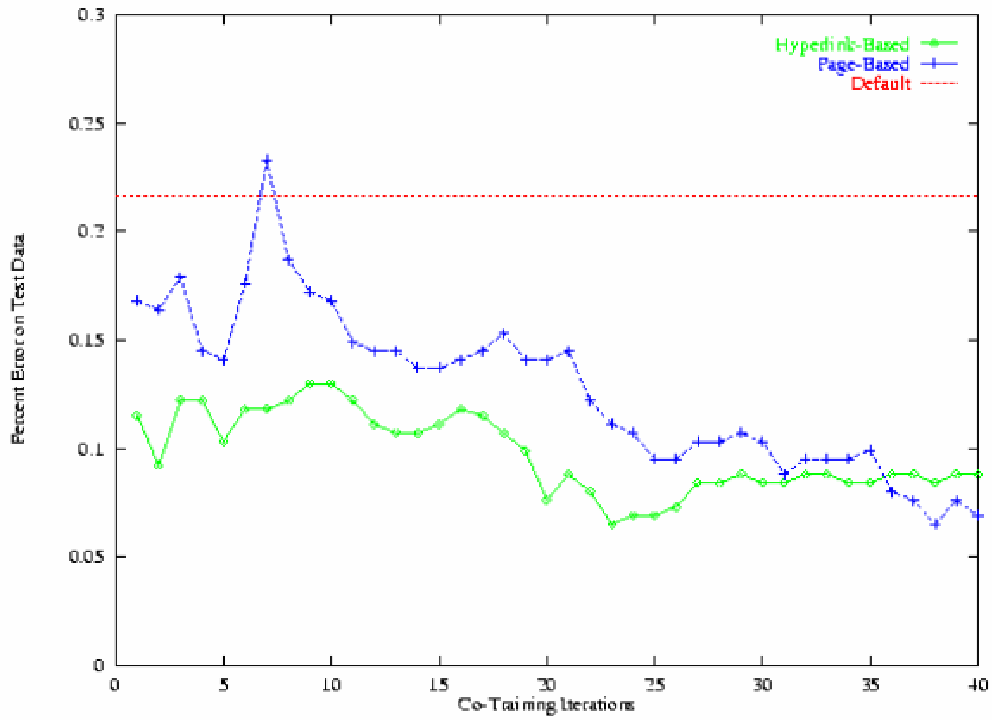
Allow g2 to label *p* positive, *n* negative examps from U

Add these self-labeled examples to L

Co-Training experimental results

- begin with 12 labeled web pages (academic course)
- provide 1,000 additional unlabeled web pages
- average error: learning from labeled data 11.1%;
- average error: cotraining 5.0%

Typical run:



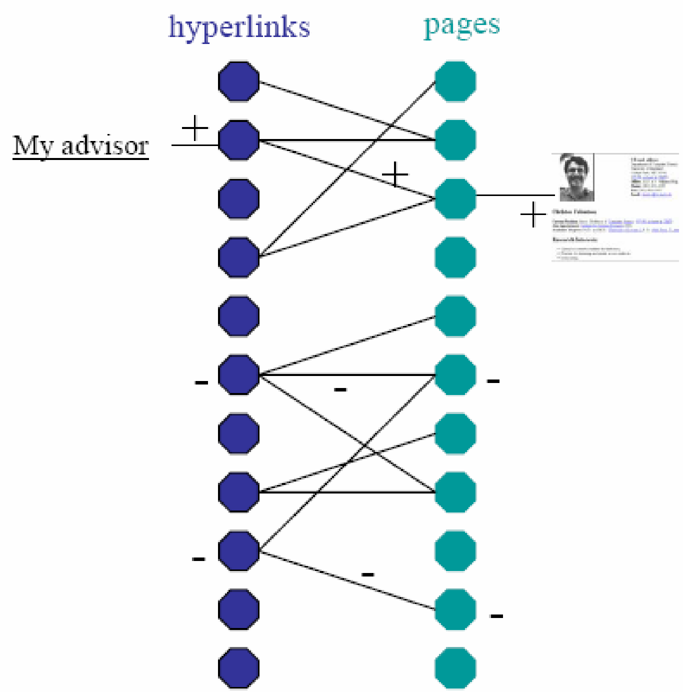
Co-Training theory

- Want to predict Y from features **X**
  - $f(\mathbf{X}) \mapsto Y$
- Co-training assumption: **X** is very expressive
  - $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$
  - want to learn $g_1(\mathbf{X}_1) \mapsto Y$ and $g_2(\mathbf{X}_2) \mapsto Y$
- *Assumption*: $\exists\ g_1, g_2, \forall\ \mathbf{x}\ g_1(\mathbf{x}_1) = f(\mathbf{x}), g_2(\mathbf{x}_2) = f(\mathbf{x})$
- Questions:
  - Does unlabeled data always help?
  - How many labeled examples do I need?
  - How many unlabeled examples do I need?

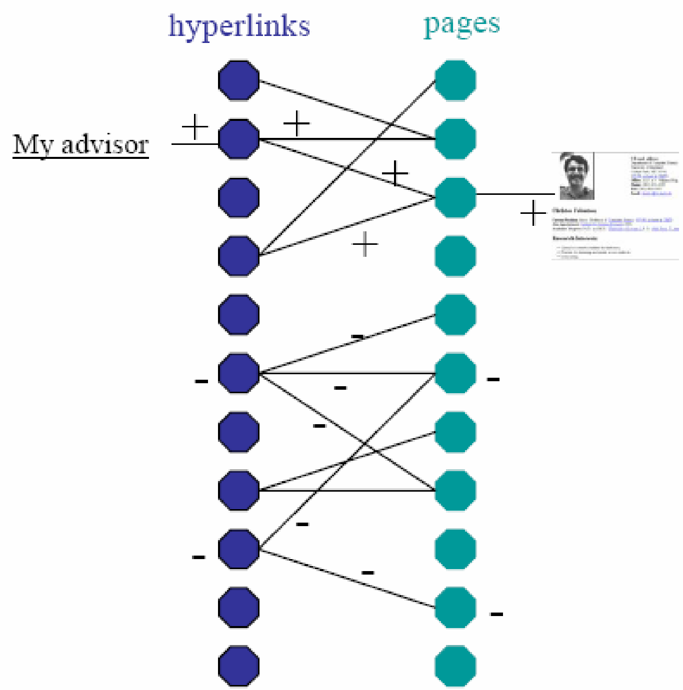
Understanding Co-Training: A simple setting

- Suppose $\mathbf{X}_1$ and $\mathbf{X}_2$ are discrete
  - $|\mathbf{X}_1| = |\mathbf{X}_2| = N$
- No label noise
- Without unlabeled data, how hard is it to learn g_1 (or g_2)?

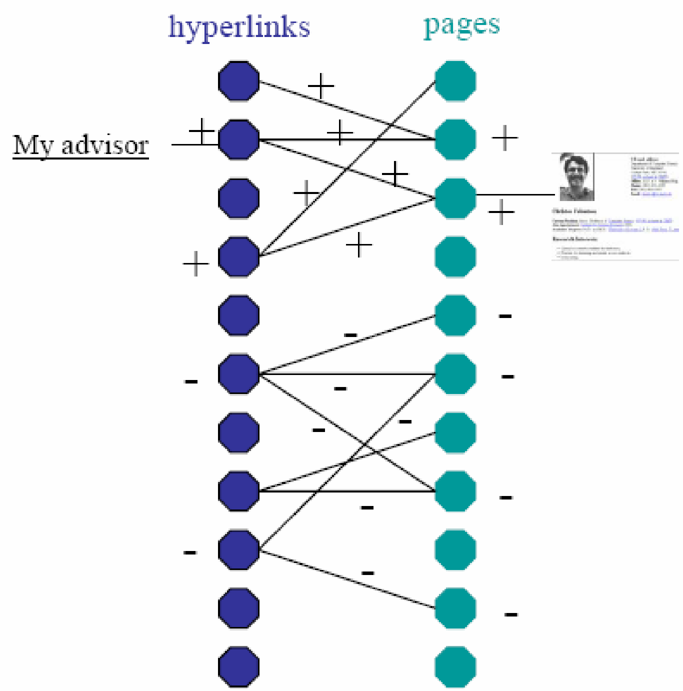
Co-Training in simple setting – Iteration 0



Co-Training in simple setting – Iteration 1

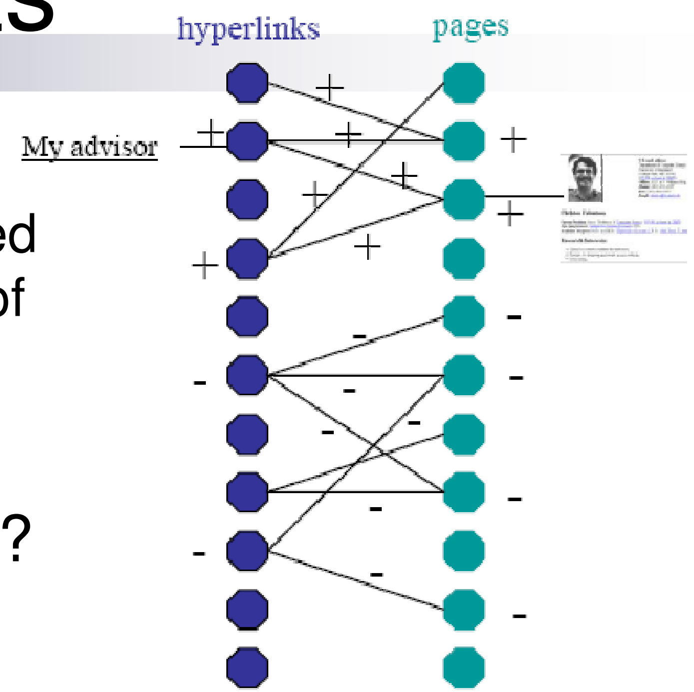


Co-Training in simple setting – after convergence



Co-Training in simple setting – Connected components

- Suppose infinite **unlabeled** data
  - Co-training must have at least one labeled example in each connected component of L+U graph



- What's probability of making an error?

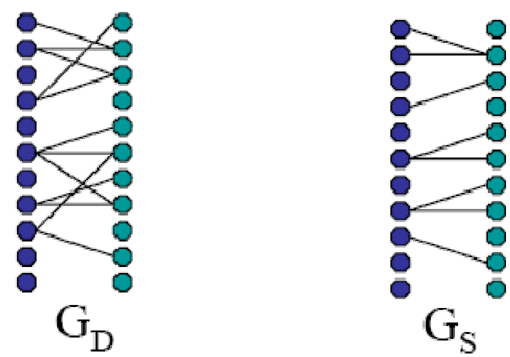
$$E[error] = \sum_j P(x \in g_j)(1 - P(x \in g_j))^m$$

Where g_j is the jth connected component of graph of L+U, m is number of labeled examples

- For k Connected components, how much labeled data?

How much unlabeled data?

Want to assure that connected components in the underlying distribution, G_D, are connected components in the observed sample, G_S



$O(\log(N)/\alpha)$ examples assure that with high probability, G_S has same connected components as G_D [Karger, 94]

N is size of G_D, α is min cut over all connected components of G_D

Co-Training theory

- Want to predict Y from features **X**
  - $f(\mathbf{X}) \mapsto Y$
- Co-training assumption: **X** is very expressive
  - $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$
  - want to learn $g_1(\mathbf{X}_1) \mapsto Y$ and $g_2(\mathbf{X}_2) \mapsto Y$
- *Assumption*: $\exists\, g_1, g_2, \forall\, \mathbf{x}\; g_1(\mathbf{x}_1) = f(\mathbf{x}), g_2(\mathbf{x}_2) = f(\mathbf{x})$
- One co-training result [Blum & Mitchell '99]
  - If
    - $(\mathbf{X}_1 \perp \mathbf{X}_2 \mid Y)$
    - g_1 & g_2 are PAC learnable from noisy data (and thus f)
  - Then
    - f is PAC learnable from weak initial classifier plus unlabeled data

What you need to know about co-training

- Unlabeled data can help supervised learning (a lot) when there are (mostly) independent redundant features
- One theoretical result:
  - If ($\mathbf{X}_1 \perp \mathbf{X}_2 \mid Y$) and g_1 & g_2 are PAC learnable from noisy data (and thus f)
  - Then f is PAC learnable from weak initial classifier plus unlabeled data
  - Disagreement between g_1 and g_2 provides bound on error of final classifier
- Applied in many real-world settings:
  - Semantic lexicon generation [Riloff, Jones 99] [Collins, Singer 99], [Jones 05]
  - Web page classification [Blum, Mitchell 99]
  - Word sense disambiguation [Yarowsky 95]
  - Speech recognition [de Sa, Ballard 98]
  - Visual classification of cars [Levin, Viola, Freund 03]

Acknowledgement

- I would like to thank Tom Mitchell for some of the material used in this presentation of co-training

Reading:
Vapnik 1998
Joachims 1999 (see class website)

Transductive SVMs

Machine Learning – 10701/15781

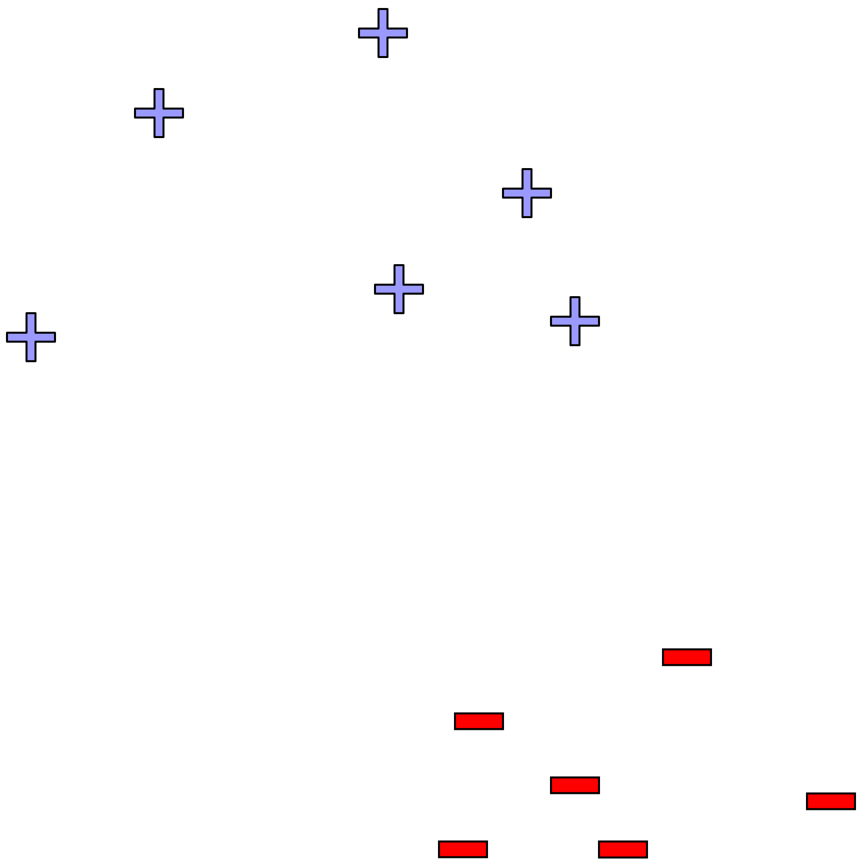
Carlos Guestrin
Carnegie Mellon University

April 17th, 2006

Semi-supervised learning and discriminative models

- We have seen semin-supervised learning for generative models
  - EM
- What can we do for discriminative models
  - Not regular EM
    - we can't compute P(x)
    - But there are discriminative versions of EM
  - Co-Training!
  - Many other tricks… let's see an example

Linear classifiers – Which line is better?



Data:

$$\left\langle x_1^{(1)}, \ldots, x_1^{(m)}, y_1 \right\rangle$$
$$\vdots$$
$$\left\langle x_n^{(1)}, \ldots, x_n^{(m)}, y_n \right\rangle$$

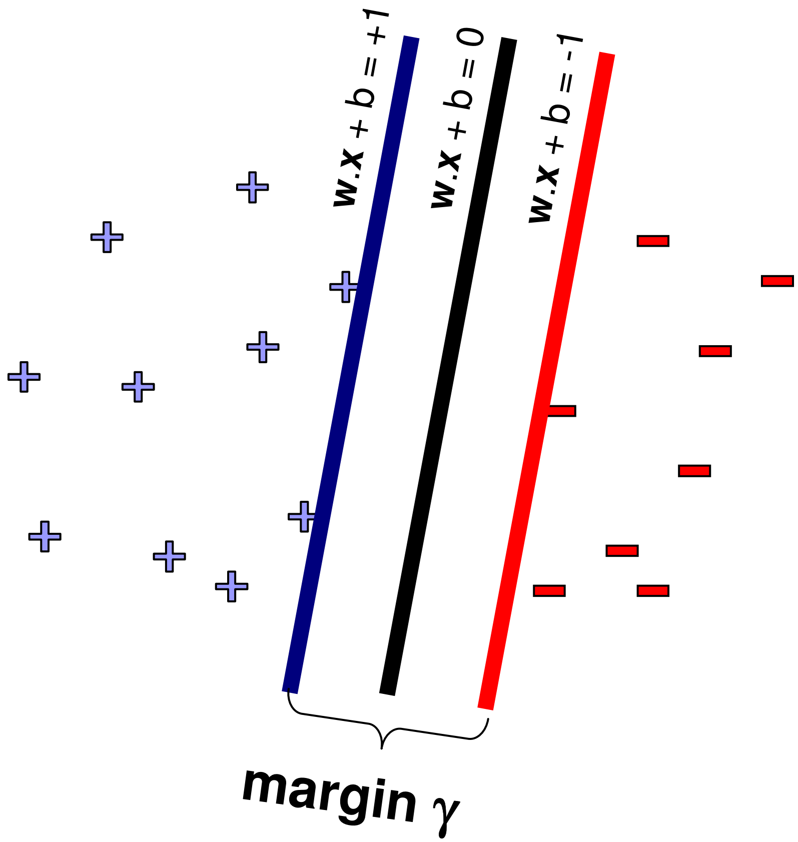
Example i:

$\left\langle x_i^{(1)}, \ldots, x_i^{(m)} \right\rangle$ — m features

$y_i \in \{-1, +1\}$ — class

$\mathbf{w}.\mathbf{x} = \sum_j w^{(j)}\ x^{(j)}$

Support vector machines (SVMs)

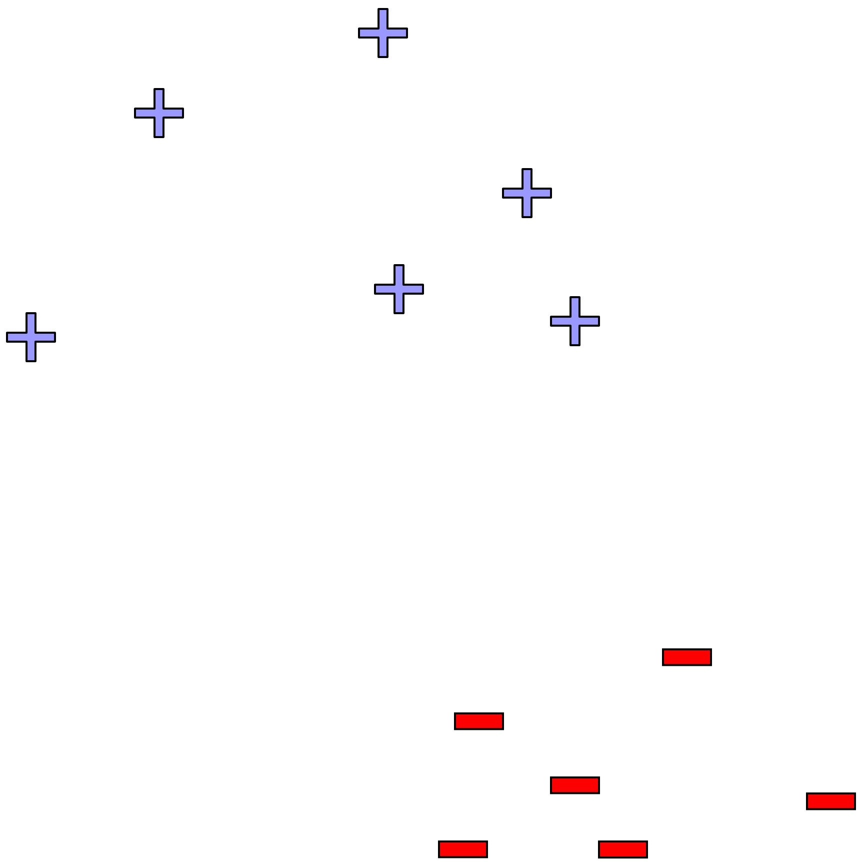


$$\text{minimize}_{\mathbf{w}} \quad \mathbf{w}.\mathbf{w}$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \forall j$$

- Solve efficiently by quadratic programming (QP)
  - Well-studied solution algorithms
- Hyperplane defined by support vectors

What if we have unlabeled data?



n_L Labeled Data:

$$\left\langle x_1^{(1)}, \ldots, x_1^{(m)}, y_1 \right\rangle$$
$$\vdots$$
$$\left\langle x_n^{(1)}, \ldots, x_n^{(m)}, y_{n_L} \right\rangle$$

Example i:

$\left\langle x_i^{(1)}, \ldots, x_i^{(m)} \right\rangle$ — m features

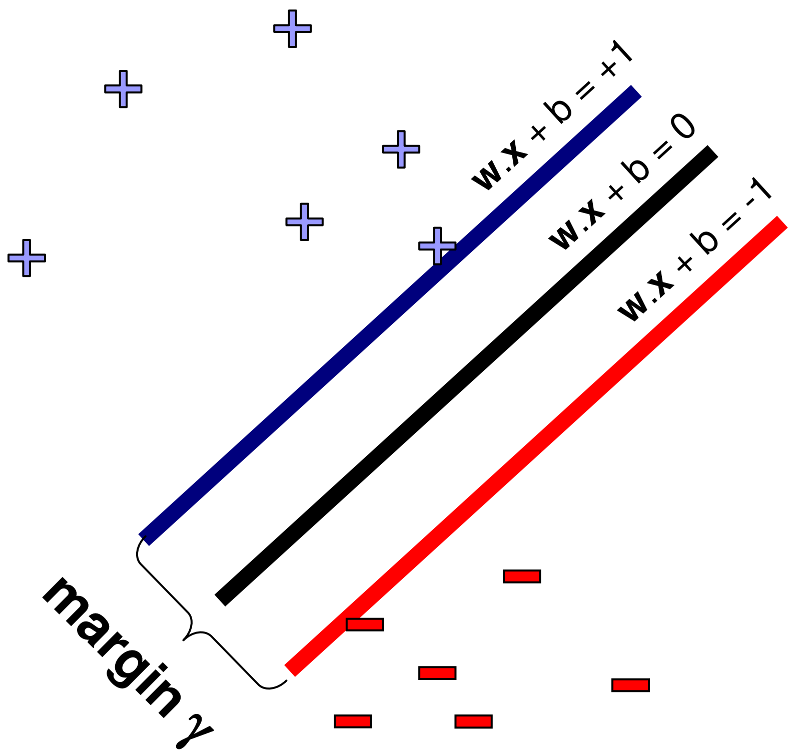
$y_i \in \{-1, +1\}$ — class

n_U Unlabeled Data:

$$\left\langle x_1^{(1)}, \ldots, x_1^{(m)}, ? \right\rangle$$
$$\vdots$$
$$\left\langle x_n^{(1)}, \ldots, x_{n_U}^{(m)}, ? \right\rangle$$

$\mathbf{w}.\mathbf{x} = \sum_j w^{(j)} x^{(j)}$

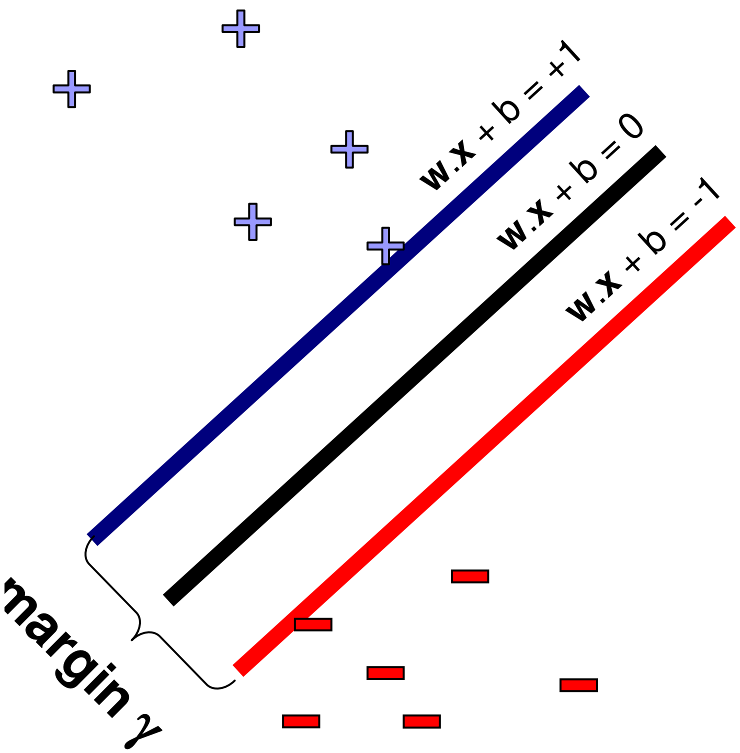
Transductive support vector machines (TSVMs)



$$\text{minimize}_{\mathbf{w}} \quad \mathbf{w}.\mathbf{w}$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \;\; \forall j$$

Transductive support vector machines (TSVMs)



$$\text{minimize}_{\mathbf{w},\{\hat{y}_1,\ldots,\hat{y}_{n_U}\}} \quad \mathbf{w}.\mathbf{w}$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \;\; \forall j = 1, \ldots, n_L$$

$$\left(\mathbf{w}.\mathbf{x}_u + b\right) \hat{y}_u \geq 1, \;\; \forall u = 1, \ldots, n_U$$

$$\hat{y}_u \in \{-1, +1\}, \;\; \forall u = 1, \ldots, n_U$$

What's the difference between transductive learning and semi-supervised learning?

- Not much, and
- A lot!!!

- Semi-supervised learning:
  - labeled and unlabeled data $\rightarrow$ learn **w**
  - use **w** on test data

- Transductive learning
  - same algorithms for labeled and unlabeled data, but…
  - unlabeled data is test data!!!

- You are learning on the test data!!!
  - OK, because you never look at the labels of the test data
  - can get better classification
  - but be very very very very very very very very careful!!!
    - never use test data prediction accuracy to tune parameters, select kernels, etc.

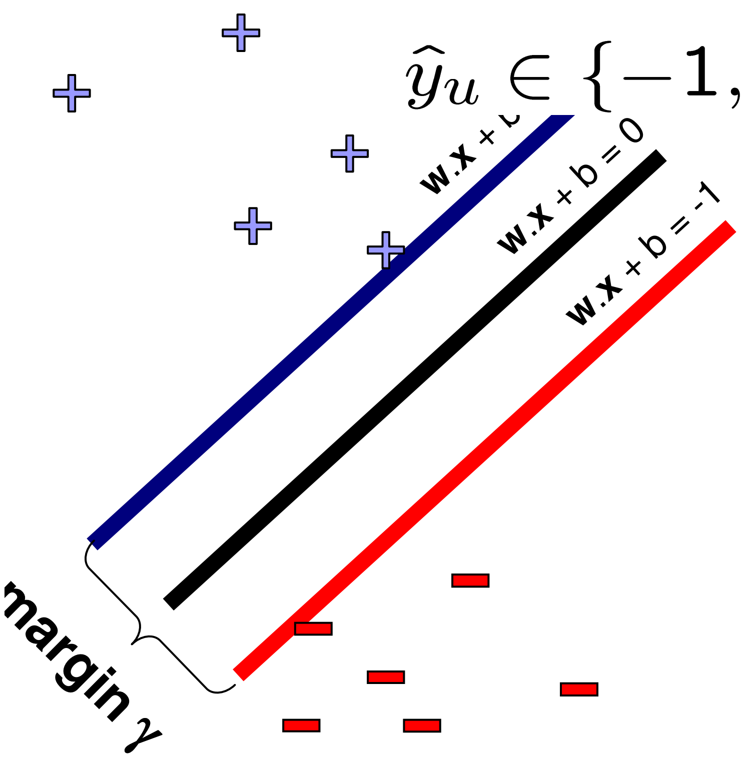
Adding slack variables

$$\text{minimize}_{\mathbf{w},\{\hat{y}_1,\ldots,\hat{y}_{n_U}\}} \quad \mathbf{w}.\mathbf{w}$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 \qquad \forall j = 1,\ldots,n_L$$

$$(\mathbf{w}.\mathbf{x}_u + b)\,\hat{y}_u \geq 1 \qquad \forall u = 1,\ldots,n_U$$

$$\hat{y}_u \in \{-1,+1\},\ \forall u = 1,\ldots,n_U$$



Transductive SVMs – now with slack variables! [Vapnik 98]

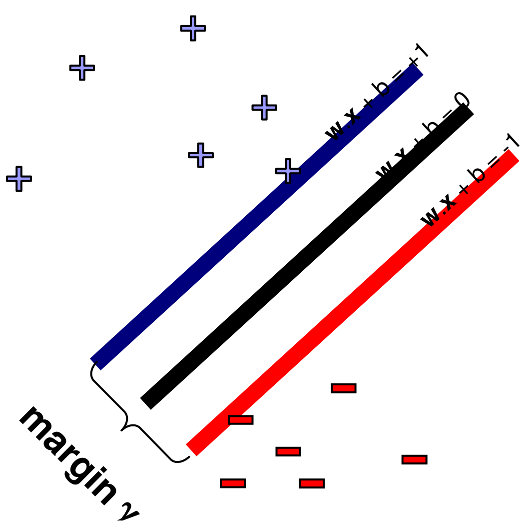
- Optimize $\mathbf{w}, \{\xi_1, \ldots, \xi_{n_L}\}, \{\widehat{y}_1, \ldots, \widehat{y}_{n_U}\}, \{\widehat{\xi}_1, \ldots, \widehat{\xi}_{n_U}\}$

minimize $\quad \mathbf{w}.\mathbf{w} + C \sum_j \xi_j + \widehat{C} \sum_u \widehat{\xi}_u$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \forall j = 1, \ldots, n_L$$

$$(\mathbf{w}.\mathbf{x}_u + b)\, \widehat{y}_u \geq 1 - \widehat{\xi}_u, \ \forall u = 1, \ldots, n_u$$

$$\widehat{y}_u \in \{-1, +1\}, \ \forall u = 1, \ldots, n_u$$



Learning Transductive SVMs is hard!

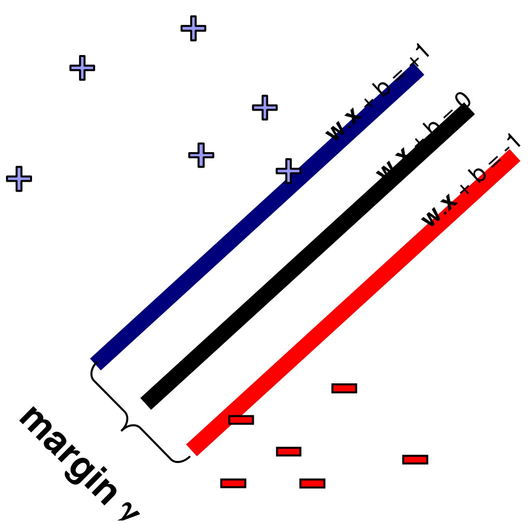
- Optimize $\mathbf{w}, \{\xi_1, \ldots, \xi_{n_L}\}, \{\hat{y}_1, \ldots, \hat{y}_{n_U}\}, \{\hat{\xi}_1, \ldots, \hat{\xi}_{n_U}\}$

minimize $\quad \mathbf{w}.\mathbf{w} + C\sum_j \xi_j + \hat{C}\sum_u \hat{\xi}_u$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j, \ \forall j = 1, \ldots, n_L$$

$$(\mathbf{w}.\mathbf{x}_u + b)\,\hat{y}_u \geq 1 - \hat{\xi}_u, \ \forall u = 1, \ldots, n_u$$

$$\hat{y}_u \in \{-1, +1\}, \ \forall u = 1, \ldots, n_u$$



- Integer Program
  - NP-hard!!!
  - Well-studied solution algorithms, but will not scale up to very large problems

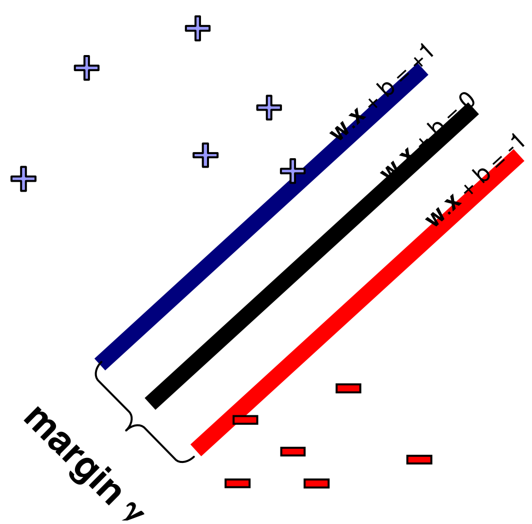
A (heuristic) learning algorithm for Transductive SVMs [Joachims 99]

minimize $\mathbf{w}.\mathbf{w} + C\sum_j \xi_j + \widehat{C}\sum_u \hat{\xi}_u$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j,\ \ \forall j = 1, \ldots, n_L$$

$$(\mathbf{w}.\mathbf{x}_u + b)\, \hat{y}_u \geq 1 - \hat{\xi}_u,\ \ \forall u = 1, \ldots, n_u$$

$$\hat{y}_u \in \{-1, +1\},\ \ \forall u = 1, \ldots, n_u$$



- If you set $\widehat{C}$ to zero $\rightarrow$ ignore unlabeled data
- Intuition of algorithm:
  - start with small $\widehat{C}$
  - add labels to some unlabeled data based on classifier prediction
  - slowly increase $\widehat{C}$
  - keep on labeling unlabeled data and re-running classifier

Some results classifying news articles – from [Joachims 99]

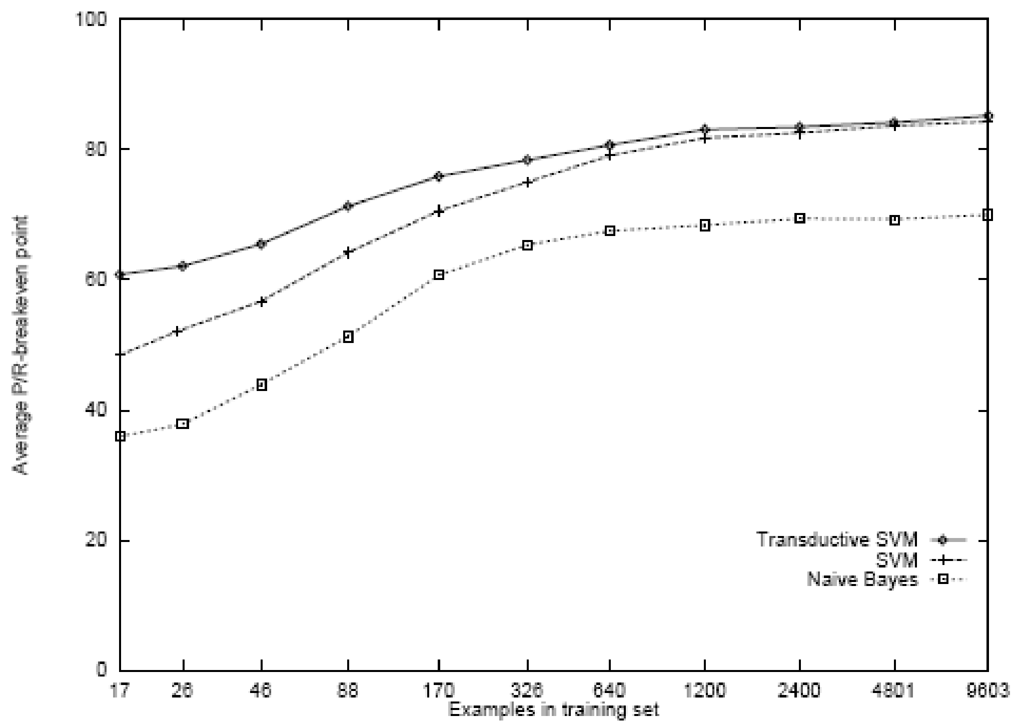


Figure 6: Average P/R-breakeven point on the Reuters dataset for different training set sizes and a test set size of 3,299.

What you need to know about transductive SVMs

- What is transductive v. semi-supervised learning
- Formulation for transductive SVM
  - can also be used for semi-supervised learning
- Optimization is hard!
  - Integer program
- There are simple heuristic solution methods that work well here