Overfitting, PAC Learning, VC Dimension, VC Bounds, Mistake Bounds, Semi-Supervised Learning

Yi Zhang

10-701, Machine Learning, Spring 2011

March 22nd, 2011

Outline

- **Overfitting**
  - **True, training, testing errors, and overfitting**
- PAC learning (finite hypothesis space)
  - Consistent learner case, and agnostic case
- PAC learning (infinite hypothesis space)
  - VC dimension, VC bounds, structural risk minimization
- Mistake bounds
  - Find-S, Halving algorithm, weighted majority algorithm
- Semi-supervised learning
  - The general idea, EM, co-training, NELL

Training error and true error

True error of hypothesis h with respect to c

- How often $h(x) \neq c(x)$ over future instances drawn at random from $\mathcal{D}$

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$

Probability distribution P(x)

Training error of hypothesis h with respect to target concept c

- How often $h(x) \neq c(x)$ over training instances D

$$error_{train}(h) \equiv \Pr_{x \in \mathsf{D}}[c(x) \neq h(x)] \equiv \frac{\sum_{x \in \mathsf{D}} \delta(c(x) \neq h(x))}{|\mathsf{D}|}$$

training examples

Training error and true error

- Is $error_{train}(h)$ an unbiased approximation to the true error $error_{\mathcal{D}}(h)$? No !
  - Training error is an approximation to the true error
  - Key: *h* ***is selected*** using training examples
  - On *h*, it is likely to be an ***underestimate***

Training error of hypothesis h with respect to target concept c

- How often $h(x) \neq c(x)$ over training instances D

$$error_{train}(h) \equiv \Pr_{x \in D}[c(x) \neq h(x)] \equiv \frac{\sum_{x \in D} \delta(c(x) \neq h(x))}{|D|}$$

training examples

True error of hypothesis h with respect to c

Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

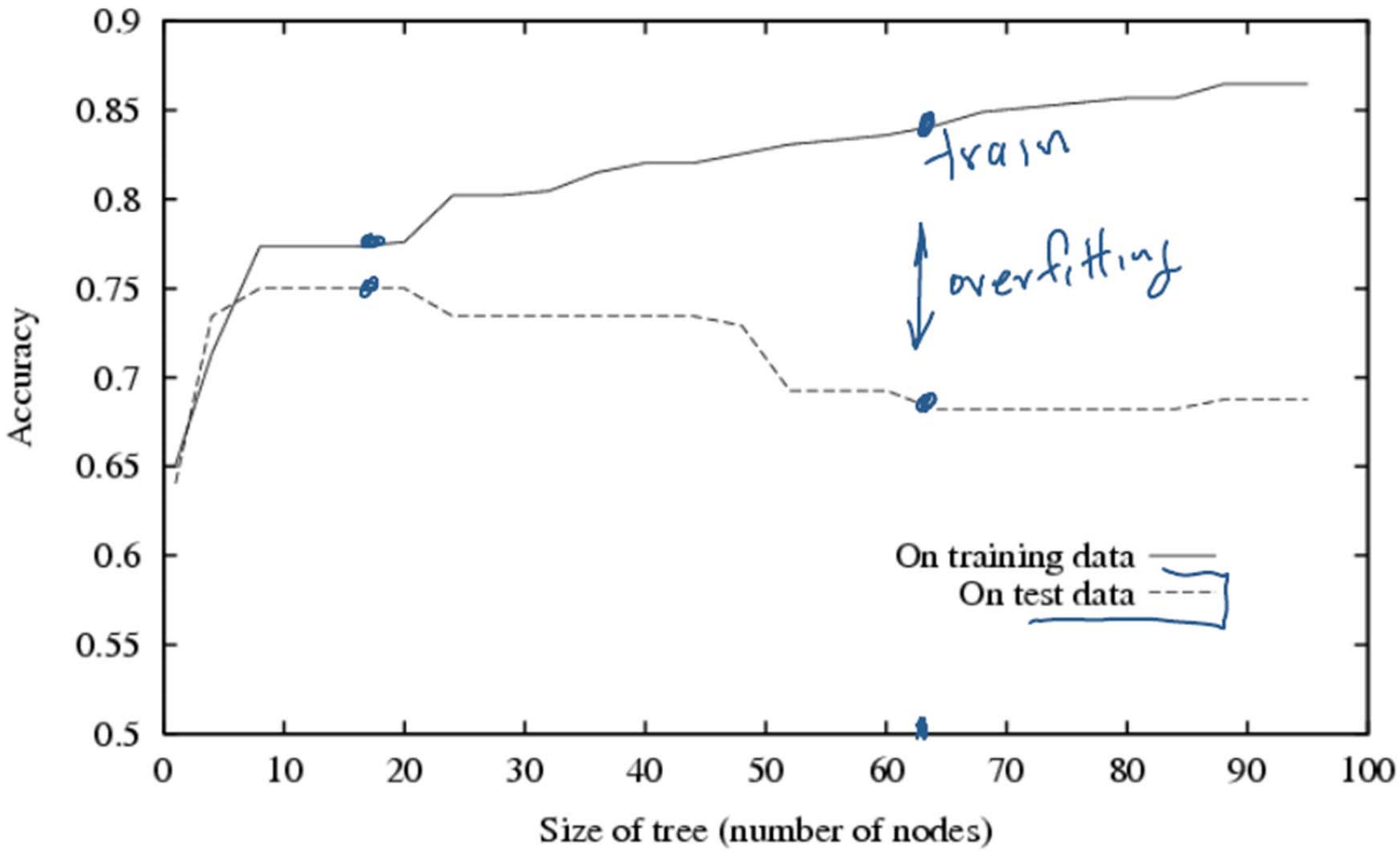
and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Testing error and true error

- Testing error is an unbiased approximation to the true error
  - as the testing set are i.i.d. samples draw from the true distribution ***independently*** of *h*

An example of overfitting



- What if the training set → infinite?

Outline

- Overfitting
  - True, training, testing errors, and overfitting
- **PAC learning (finite hypothesis space)**
  - **Consistent learner case, and agnostic case**
- PAC learning (infinite hypothesis space)
  - VC dimension, VC bounds, structural risk minimization
- Mistake bounds
  - Find-S, Halving algorithm, weighted majority algorithm
- Semi-supervised learning
  - The general idea, EM, co-training, NELL

PAC learning: finite hypothesis space

- Training error ***underestimates*** the true error !
- In PAC learning, we seek theory to relate:
  - The number of training samples: m
  - The gap between training and true errors

    $$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \epsilon$$

  - Complexity of the hypothesis space: |H|
  - Confidence of this relation: at least (1-δ)

A special case: training error is 0

- In PAC learning, we seek theory to relate:
  - The number of training samples: m
  - The gap between training (0) and true errors

$$error_{true}(h) \leq error_{train}(h) + \epsilon$$

$$\downarrow$$

$$error_{true}(h) \leq \epsilon$$

  - Complexity of the hypothesis space: |H|
  - Confidence of this relation: at least $(1-\delta)$
- What is the probability that there exists consistent hypothesis with true error > ϵ?
  - i.e., represent δ using other quantities

Derivation …

let $\underbrace{h_1, \ldots h_k}$ be the hyps $\in H$ with true error $\underline{> \epsilon}$

Prob that h_1 will be consistent with first training example

$\leq (1-\epsilon)$

" h_1 will be cons. w/ m indep drawn examps?

$\leq (1-\epsilon)^m$

" that at least of $h_1 \cdots h_k$ will be consist w/ m ~~ex~~ ?

$\leq k(1-\epsilon)^m$

$k \leq |H|$

$\leq |H|(1-\epsilon)^m$

$$\boxed{\leq |H| e^{-\epsilon m}}$$

if $0 \leq \epsilon \leq 1$
then $(1-\epsilon) \leq e^{-\epsilon}$

Bounds for finite hypothesis space

$$\Pr[(\exists h \in H) s.t. (error_{train}(h) = 0) \wedge (error_{true}(h) > \epsilon)] \;\leq\; |H|e^{-\epsilon m}$$

Suppose we want this probability to be at most δ

1. How many training examples suffice?

$$m \geq \frac{1}{\epsilon}(\ln|H| + \ln(1/\delta))$$

2. If $error_{train}(h) = 0$ then with probability at least (1-δ):

$$error_{true}(h) \leq \frac{1}{m}(\ln|H| + \ln(1/\delta))$$

Agnostic learning

- Training error is ***not*** 0
- In PAC learning, we seek theory to relate:
  - The number of training samples: m
  - The gap between training and true errors

    $$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \epsilon$$
  - Complexity of the hypothesis space: |H|
  - Confidence of this relation: at least $(1-\delta)$

Agnostic learning

- In PAC learning, we seek theory to relate:
  - The number of training samples: m
  - The gap between training and true errors

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \epsilon$$

  - Complexity of the hypothesis space: |H|
  - Confidence of this relation: at least (1-δ)
- The bound on δ

$$\Pr[(\exists h \in H) error_{true}(h) > error_{train}(h) + \epsilon] \leq |H| e^{-2m\epsilon^2}$$

  - Derived from Hoeffding bounds

Agnostic learning

- The bound on δ

$$\Pr[(\exists h \in H) error_{true}(h) > error_{train}(h) + \epsilon] \leq |H| e^{-2m\epsilon^2}$$

  - Derived from Hoeffding bounds

- Also

$$m \geq \frac{1}{2\epsilon^2}(\ln|H| + \ln(1/\delta))$$

$$error_{true}(h) \leq error_{train}(h) + \sqrt{\frac{\ln|H| + \ln\frac{1}{\delta}}{2m}}$$

PAC learnable

Consider a class C of possible target concepts defined over a set of instances X of length n, and a learner L using hypothesis space H.

Definition: C is **PAC-learnable** by L using H if for all $c \in C$, distributions $\mathcal{D}$ over X, ϵ such that $0 < \epsilon < 1/2$, and δ such that $0 < \delta < 1/2$,
learner L will with probability at least $(1 - \delta)$ output a hypothesis $h \in H$ such that $error_{\mathcal{D}}(h) \leq \epsilon$, in time that is polynomial in $1/\epsilon$, $1/\delta$, n and $size(c)$.

Sufficient condition:

Holds if learner L requires only a polynomial number of training examples, and processing per example is polynomial

Outline

- Overfitting
  - True, training, testing errors, and overfitting
- PAC learning (finite hypothesis space)
  - Consistent learner case, and agnostic case
- **PAC learning (infinite hypothesis space)**
  - **VC dimension, VC bounds, structural risk minimization**
- Mistake bounds
  - Find-S, Halving algorithm, weighted majority algorithm
- Semi-supervised learning
  - The general idea, EM, co-training, NELL

PAC learning: *infinite* hypothesis space

- Bounds for ***finite*** hypothesis space

$$m \geq \frac{1}{2\epsilon^2}(\ln|H| + \ln(1/\delta))$$

$$error_{true}(h) \leq error_{train}(h) + \sqrt{\frac{\ln|H| + \ln\frac{1}{\delta}}{2m}}$$

$$\Pr[(\exists h \in H) error_{true}(h) > error_{train}(h) + \epsilon] \leq |H| e^{-2m\epsilon^2}$$

Question: If H = {h | h: X → Y} is infinite, what measure of complexity should we use in place of |H| ?

VC dimension

- VC(H): size of the largest sample set that can be ***shattered*** by H

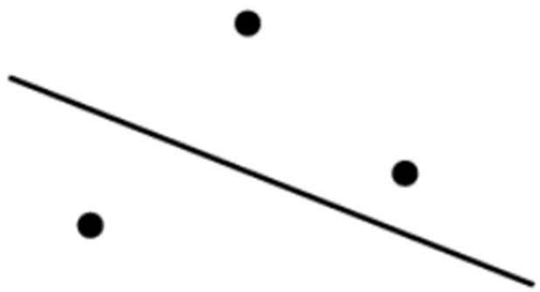
Definition: The **Vapnik-Chervonenkis dimension**, $VC(H)$, of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite sets of X can be shattered by H, then $VC(H) \equiv \infty$.

  - Shatter: correctly classify regardless of the labelings

VC dimension: an example

What is VC dimension of

- $H_2 = \{ ((w_0 + w_1x_1 + w_2x_2)>0 \rightarrow y=1) \}$
  - $VC(H_2)=3$
- For H_n = linear separating hyperplanes in n dimensions, $VC(H_n)=n+1$



VC dimension: an example

2. [3 pts] Consider a decision tree learner applied to data where each example is described by 10 boolean variables $\langle X_1, X_2, \ldots X_{10} \rangle$. What is the VC dimension of the hypothesis space used by this decision tree learner?

VC dimension: an example

2. [3 pts] Consider a decision tree learner applied to data where each example is described by 10 boolean variables $\langle X_1, X_2, \ldots X_{10} \rangle$. What is the VC dimension of the hypothesis space used by this decision tree learner?

★ **SOLUTION:** The VC dimension is 2^{10}, because we can shatter 2^{10} examples using a tree with 2^{10} leaf nodes, and we cannot shatter $2^{10} + 1$ examples (since in that case we must have duplicated examples and they can be assigned with conflicting labels).

VC(H) vs. |H|

- Any relation between VC(H) and |H| ?

$VC(H) = k$

→ shatter k examps

→ 2^k labelings of them

→ $|H| \geq 2^k$

$k \leq \log_2 |H|$

VC bounds

- Bound on m using other quantities

$$m \geq \frac{1}{\epsilon}(4\log_2(2/\delta) + 8VC(H)\log_2(13/\epsilon))$$

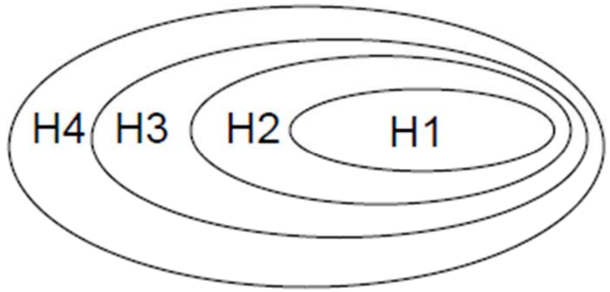
- Bound on error using other quantities

$$error_{true}(h) < error_{train}(h) + \sqrt{\frac{VC(H)(\ln\frac{2m}{VC(H)} + 1) + \ln\frac{4}{\delta}}{m}}$$

Structural risk minimization

Which hypothesis space should we choose?

- Bias / variance tradeoff



SRM: choose H to minimize bound on expected true error!

$$error_{true}(h) < error_{train}(h) + \sqrt{\frac{VC(H)(\ln \frac{2m}{VC(H)} + 1) + \ln \frac{4}{\delta}}{m}}$$

Outline

- Overfitting
  - True, training, testing errors, and overfitting
- PAC learning (finite hypothesis space)
  - Consistent learner case, and agnostic case
- PAC learning (infinite hypothesis space)
  - VC dimension, VC bounds, structural risk minimization
- **Mistake bounds**
  - **Find-S, Halving algorithm, weighted majority algorithm**
- Semi-supervised learning
  - The general idea, EM, co-training, NELL

Mistake bounds

- Consider the following setting:
  - Instances draw randomly from X according to the data distribution P(X)
  - The learner must classify each instance x before knowing its label
  - How many mistakes before the learner converges to the correct concept?

Mistake bounds

- Consider the following setting:
  - Instances draw randomly from X according to the data distribution P(X)
  - The learner must classify each instance x before knowing its label
  - How many mistakes before the learner converges to the correct concept?
- Analogy: given a pool of "experts", how many mistakes before we find the "true expert"?
- Difference from the PAC learning bound
  - Do not care about how many samples we see
  - Care about how many mistakes we make

Mistake Bounds: Find-S

$x = \langle x_1, x_2 \dots x_n \rangle$ ← boolean $y \in \{0,1\}$

e.g. $h = \underbrace{(x_2 = 1)}_{= l_2} \wedge \underbrace{(x_n = 0)}_{\neg l_n} \rightarrow y = 1$

$\rightarrow y = 1$

Consider Find-S when H = conjunction of boolean literals

Find-S:

- Initialize h to the most specific hypothesis $l_1 \wedge \cancel{\neg l_1} \wedge l_2 \wedge \cancel{\neg l_2} \dots \cancel{l_n} \wedge \neg l_n$
- For each positive training instance x
  - Remove from h any literal that is not satisfied by x
- Output hypothesis h.

Start with $2n$ lits.

mistake 1: remove n

= first + example

Mistake 2: remove 1 or more

⋮

k: 1

How many mistakes before converging to correct h? $\leq n+1$

Halving algorithm

- Start from a hypothesis space H
- Given each new instance x
  - Majority voting from all h in H to classify x
  - Obtain the label of x
  - Remove from H those misclassify x
- Bound the number of mistakes K?

initial size of VS = $|H|$

after 1 mistake $\leq |H| \frac{1}{2}$

K mistakes $\leq \boxed{|H| (\frac{1}{2})^K} \rightarrow K \leq \lfloor \log_2 |H| \rfloor$

Optimal Mistake Bounds

Let $M_A(C)$ be the max number of mistakes made by algorithm A to learn concepts in C. (maximum over all possible $c \in C$, and all possible training sequences)

$$M_A(C) \equiv \max_{c \in C} M_A(c)$$

Definition: Let C be an arbitrary non-empty concept class. The **optimal mistake bound** for C, denoted $Opt(C)$, is the minimum over all possible learning algorithms A of $M_A(C)$.

$$Opt(C) \equiv \min_{A \in learning\ algorithms} M_A(C)$$

$$\boxed{VC(C) \leq Opt(C) \leq M_{Halving}(C) \leq log_2(|C|).}$$

Weight Majority Algorithm

- What is there is no "perfect" function h in the hypothesis space H?
- Can we design an algorithm using H, such that #mistakes is "close" to using the best h in H?
- Yes! Weighted majority algorithm:
  - Assign initial weight one to each h in H
  - Make prediction by weighted majority voting
  - Update the weight of each h in H

Weighted Majority Algorithm

a_i denotes the i^{th} prediction algorithm in the pool A of algorithms. w_i denotes the weight associated with a_i.

- For all i initialize $\boxed{w_i \leftarrow 1}$
- For each training example $\langle x, c(x)\rangle$
  * Initialize q_0 and q_1 to 0
  * For each prediction algorithm a_i
    - If $a_i(x) = 0$ then $q_0 \leftarrow q_0 + w_i$

      If $a_i(x) = 1$ then $q_1 \leftarrow q_1 + w_i$
  * If $q_1 > q_0$ then predict $c(x) = 1$

    If $q_0 > q_1$ then predict $c(x) = 0$

    If $q_1 = q_0$ then predict 0 or 1 at random for $c(x)$
  * For each prediction algorithm a_i in A do

    If $a_i(x) \neq c(x)$ then $w_i \leftarrow \beta w_i$

when β=0, equivalent to the Halving algorithm…

$\beta = 0.5$

Weighted Majority

Even algorithms that learn or change over time...

[Relative mistake bound for WEIGHTED-MAJORITY] Let D be any sequence of training examples, let A be any set of n prediction algorithms, and let k be the minimum number of mistakes made by any algorithm in A for the training sequence D. Then the number of mistakes over D made by the WEIGHTED-MAJORITY algorithm using $\beta = \frac{1}{2}$ is at most

$$2.4(k + \log_2 n) \geq \text{\# mistakes by Wtd Maj}$$

let M be # of mistakes made by Wtd Maj. Alg using n algs.

K # " " by best $a_i \in A$

$W = \sum_i w_i$

What is final wt of alg a_i? $\left(\frac{1}{2}\right)^K$

What is final $\sum_{j=1}^{n} w_j$

What is initial $W = n$

after mistake #1, $W \le \frac{3}{4} n$

after mistake M $\rightarrow$ $W \le \left(\frac{3}{4}\right)^M n$

$\boxed{w_i \le W}$ $\left(\frac{1}{2}\right)^K \le W \le \left(\frac{3}{4}\right)^M n$

$$\boxed{\left(\frac{1}{2}\right)^K \le \left(\frac{3}{4}\right)^M n}$$

Outline

- Overfitting
  - True, training, testing errors, and overfitting
- PAC learning (finite hypothesis space)
  - Consistent learner case, and agnostic case
- PAC learning (infinite hypothesis space)
  - VC dimension, VC bounds, structural risk minimization
- Mistake bounds
  - Find-S, Halving algorithm, weighted majority algorithm
- **Semi-supervised learning**
  - **The general idea, EM, co-training, NELL**

Semi-supervised learning

Consider problem setting:

- Set *X* of instances drawn from unknown distribution *P(X)*
- Wish to learn target function *f: X*→ *Y* (or, P(Y|X))
- Given a set *H* of possible hypotheses for *f*

Given:

- i.i.d. labeled examples $L = \{\langle x_1, y_1 \rangle \ldots \langle x_m, y_m \rangle\}$
- i.i.d. unlabeled examples $U = \{x_{m+1}, \ldots x_{m+n}\}$

Semi-supervised learning

Consider problem setting:

- Set *X* of instances drawn from unknown distribution *P(X)*
- Wish to learn target function *f: X*→ *Y* (or, P(Y|X))
- Given a set *H* of possible hypotheses for *f*

Given:

- i.i.d. labeled examples $L = \{\langle x_1, y_1 \rangle \ldots \langle x_m, y_m \rangle\}$
- i.i.d. unlabeled examples $U = \{x_{m+1}, \ldots x_{m+n}\}$

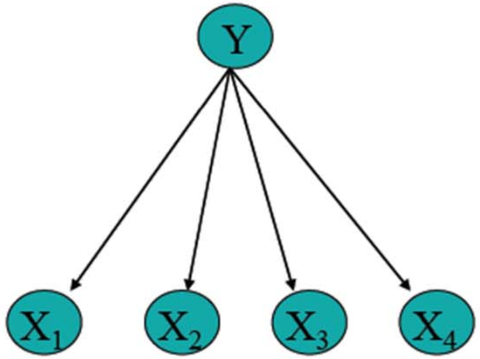
- Why do we care?
  - Unlabeled data is much easier to obtain!

- How can we use unlabeled data to help?

EM

Using Unlabeled Data to Help Train Naïve Bayes Classifier

Learn P(Y|X)

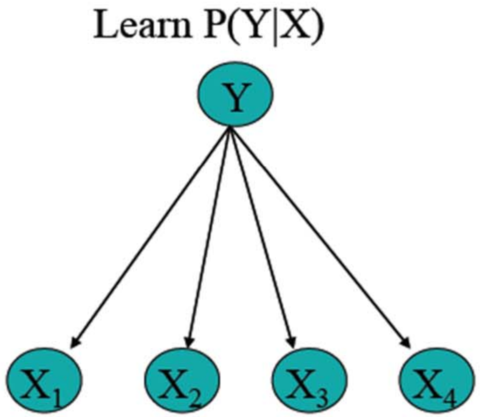


Y	X1	X2	X3	X4
1	0	0	1	1
0	0	1	0	0
0	0	0	1	0
E[Y] ?	0	1	1	0
E[Y] ?	0	1	0	1

- Learn the initial model using a few labeled data
- Iterate:
  - Use the model to "guess" unknown labels
  - Re-learn the model using labeled + unlabeled data

EM

Using Unlabeled Data to Help Train
Naïve Bayes Classifier

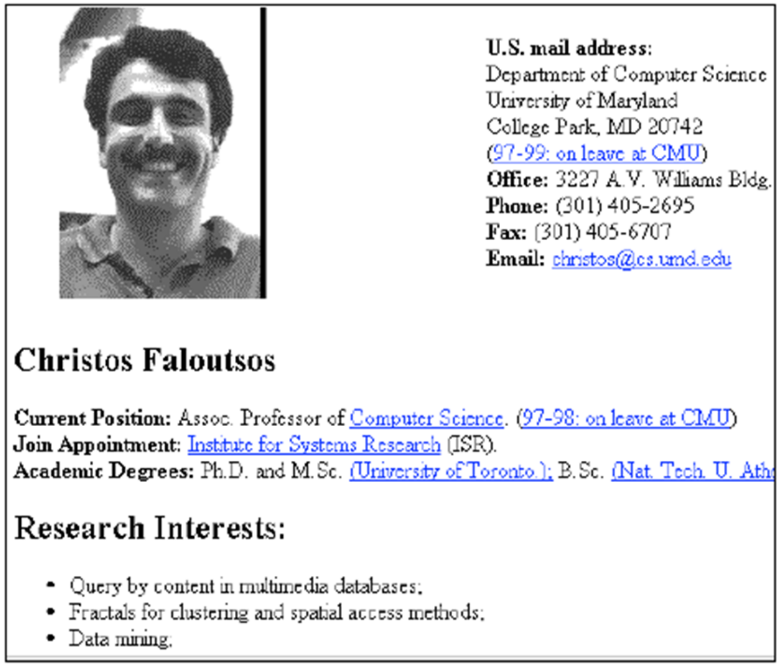


Y	X1	X2	X3	X4
✓ 1	0	0	1	1
✓ 0	0	1	0	0
✓ 0	0	0	1	0
E[Y] → ?	0	1	1	0
E[Y] → ?	0	1	0	1

- Any problem?
  - The initial model can be inaccurate
  - The “guess” on unknown labels may be inaccurate
  - Model re-learned using inaccurate information

Co-training and multi-view learning

- Features in X can be split into multiple views
- Ideally, each view is sufficient to predict Y
- Ideally, views are conditionally independent given Y
- Example: hyperlink view + page view → prof. or not?


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. Wiliams 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;

CoTraining Algorithm #1

[Blum&Mitchell, 1998]

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

- Difference to EM
  - Directly assign labels instead of estimating expectation
  - Use two (or more) models from different views !
- Potential problem? Self-labeling noise?

CoTraining Algorithm #1

[Blum&Mitchell, 1998]

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

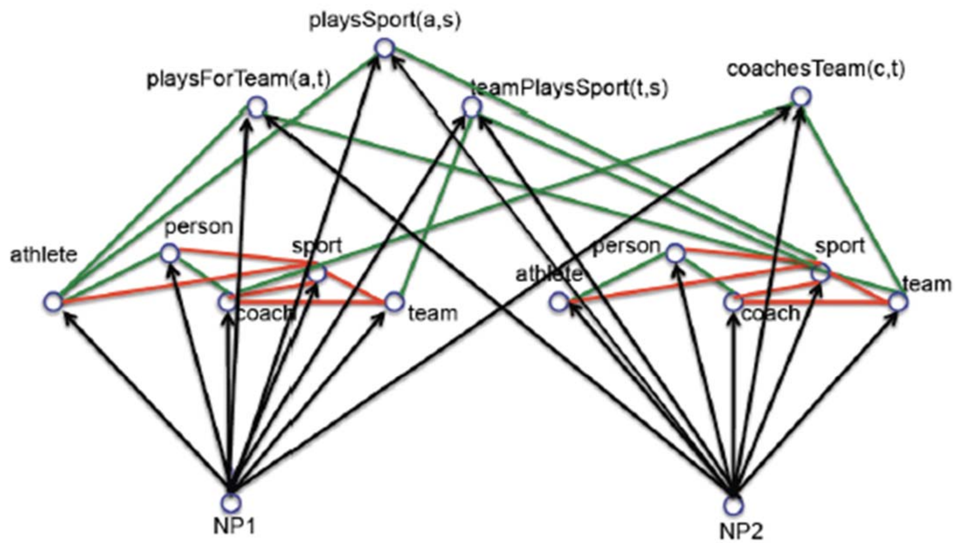
Add these self-labeled examples to L

- Idea for dealing with self-labeling noise?
- Last step:
  - Add only ***consistent*** self-labeled examples to L?

Semi-supervised Learning in NELL

- NELL (never-ending language learning)
- Coupled semi-supervised learning

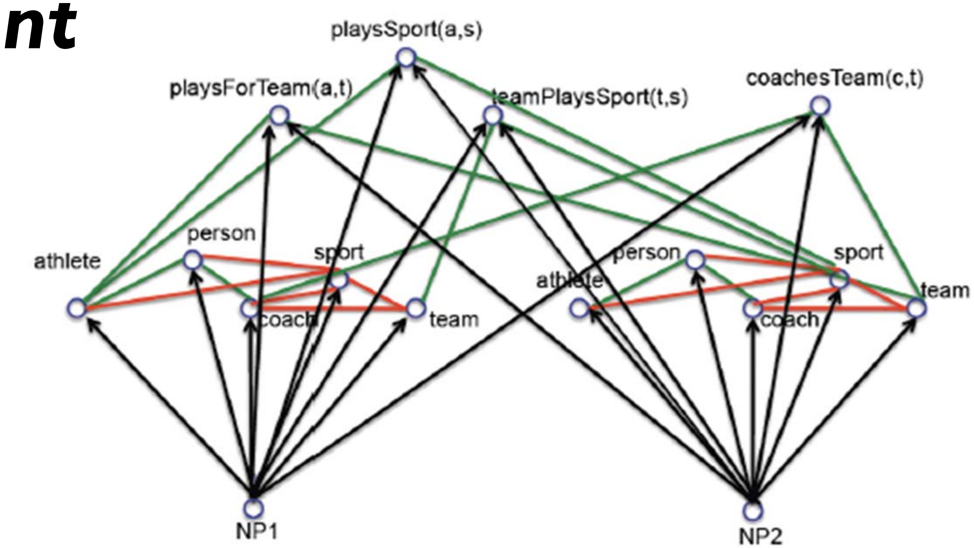
hard
(underconstrained)
semi-supervised
learning problem



much easier (more constrained)
semi-supervised learning problem

Coupled semi-supervised learning

- Given: labeled set L, unlabeled set U
- Loop
  - For each task i, learn the classifier f_i using L
  - For each task i, use f_i to label samples in U
  - Add self-labeled examples to L if labels from all f_i are ***consistent***



Semi-supervised learning

- Self labeling is only one way for SSL
- Many many other ways …
- See:
  - Xiaojin Zhu. Semi-Supervised Learning Literature Survey.

Questions?