Machine Learning in Natural Language

Dan Roth
University of Illinois, Urbana-Champaign
danr@cs.uiuc.edu
http://L2R.cs.uiuc.edu/~danr

Who Are You?

- Undergrads? Ph.D students? Post Ph.D?
- Background:
  - Natural Language
  - Learning
  - Algorithms/Theory of Computation
- Native English?
- Interaction is important!
  Please, ask questions and make comments

An Owed to the Spelling Checker

I have a spelling checker, it came with my PC
It plane lee marks four my revue
Miss steaks aye can knot sea.
Eye ran this poem threw it, your sure reel glad two no.
Its vary polished in it's weigh
My checker tolled me sew.
A checker is a bless sing, it freeze yew lodes of thyme.
It helps me right awl stiles two reed
And aides me when aye rime.
Each frays come posed up on my screen
Eye trussed to bee a joule...

Ambiguity Resolution

Can I have a peace of cake ? piece

...Nissan Car and truck plant is ...
...divide life into plant and animal kingdom

Buy a car with a steering wheel (his money)

The dog bit the kid. He was taken to a veterinarian hospital

Role of Learning

- "Solving" a natural language problem requires addressing a wide variety of questions.
- Learning is at the core of any attempt to make progress on these questions.
- Learning has multiple purposes
  -Knowledge Acquisition

  -Classification/ Context sensitive disambiguation

  -integration of various knowledge sources to ensure robust behavior

Why Natural Language?

- Main channel of Communication
- Knowledge Acquisition

Important for Cognition and Engineering perspectives

- A grand application: Human computer interaction.
  - Language understanding and generation
  - Knowledge acquisition
  - NL interface to complex systems
  - Querying NL databases
  - Reducing information overload

Specific tasks include: QA, summarization, translation.

Why Learning in Natural Language?

- Challenging from a Machine Learning perspective
- Language Comprehension: a large scale phenomenon in terms of both knowledge and computation
- Requires an integrated approach: need to solve problems in learning, inference, knowledge representation…
- There is not "cheating": no toy problems.

Why Learning in Natural Language?

There is no significant aspect of natural language that can be studied without giving learning a principle role..

The role of learning in the study of natural language is significant both from the engineering point of view as it is from the cognitive point of view.

Learning is important in language acquisition, language change, language variation, language generation and language comprehension.

This Short Course

- There are many topics to cover.
- This is a very active field of research, many ideas are floating around, most of which will not stay.
- We will cover some of the main ideas and techniques.
- Attempt to abstract away from specific works and understand the main paradigms.

Examples: real problems; state of the art approaches

The Course Plan

1. Introduction
   - Why learning? Learning vs. Statistics; Motivating Example
2. Learning Paradigms
   - Pac Learning; Bayesian paradigm;
3. Algorithms
   - Algorithmic issues; Linear Algorithms
4. Bayesian
   - Maximum Likelihood; Max Entropy
5. Unsupervised.
   - EM; SemiSuperviesd learning; Active vs.
6. Inference: moving to higher level tasks
   - Markov Models; Constraint Satisfaction

Interaction is important; Problems will be "suggested"

More Detailed Plan (I)

1. Introduction to Natural Language Learning
   - Why is it difficult ?
   - Statistics vs. Learning
   - When do we need learning?
   - Examples of problems
   - Detailed example: Context Sensitive Spelling: features and an algorithm;
   - Corpus based work: data and tasks. System demo.
2. Learning Paradigms
   - PAC Learning
   - Bayesian Learning
   - Examples

More Detailed Plan (II)

3. Learning Algorithms
   - Examples
   - General Paradigm: feature based representation
   - Linear functions
   - On line algorithms: additive/multiplicative update
   - Decision Lists; TBL
   - Memory Based
4. Bayesian Methods
   - Naïve Bayes
   - HMMs (Predictions and model learning)
   - Max Entropy
   - LSQ: why do probabilistic algorithms work?

More Detailed Plan (II)

5. Relaxing Supervision
   - EM
   - Semi supervised learning: co-learning vs. selection
6. Inference: Moving to higher level tasks
   - HMMs (Predictions and model learning)
   - HMMs (with Classifiers), PMMs
   - Constraint Satisfaction/ ad hoc methods
   - Running example: part of speech tagging/ Shallow Parsing/ Parsing

Today

- Examine some of the philosophical themes and leading ideas that motivate statistical approaches to linguistics and natural language and to
- Begin exploring what can be learned by looking at statistics of texts.

Abney's Paper

Thanks for all you help

Statistics

Thanks for all you help

Thanks for all your help

Both sentences have legitimate parse, but you have chosen a reading of the sentence that gives rise to a more common one

History

- Some examples

Statistics of Text

- Some examples (words)
- Other examples (pos tags)

(try: will, win, study, can)

POS tags is an important problem that was considered difficult.
~1988: baseline is over 90%

Statistics of Text

Th_r_ _s _nly _n_ w_y t_ f_ll _n
Th_ v_w_ls _n th_s s_nt_nc_

- We rely heavily on the "statistics" of text.
- Redundancy helps communication
- Predictions are aided by some global coherency

This Course

- The course is not on statistical NLP.

We are going to discuss and gain a lot from the fact that language satisfies statistical regularities; but, the main goal is not to explore these statistical regularities but rather to exploit their existence in order to make progress towards language understanding.

- In doing that, knowledge representation is as important
(that's why it's learning rather than statistics).

Context Sensitive Text Correction

Illinois' bored of education.	board
We took a walk it the park two.	in, too
We fill it need no be this way	feel, not
The amount of chairs in the room is...	number
I'd like a peace of cake for desert	piece, dessert

Disambiguation Problems

Middle Eastern ____ are known for their sweetness

Task: Decide which of { deserts , desserts } is more likely in the given context.

Ambiguity: modeled as confusion sets (class labels C)

C={ deserts, desserts}
C={ Noun,Adj.., Verb...}
C={ topic=Finance, topic=Computing}
C={ NE=Person, NE=location}

Disambiguation Problems

- Archetypical disambiguation problem
- Data is available
- In principle, a solved problem
Golding&Roth, Mangu&Brill,...
- But
- Many issues are involved in making an "in principle" solution a realistic one

Outline

- Modeling Context Sensitive Text Correction
- A learning algorithm
- Experimental Results/ Demo
- Some questions

Learning to Disambiguate

- Given
  - a confusion set C={ deserts, desserts}
  - sentence (s)

  Middle Eastern ____ are known for their sweetness
- Map into a feature based representation
- Learn a function F_C that determines which of C={ deserts, desserts} more likely in a given context.
- Evaluate the function on future C sentences

Learning Approach: Representation

S= I don't know whether to laugh or cry

[x x x x]

Consider words, pos tags, relative location in window

Generate binary features representing presence of:

a word/pos within window around target word

conjunctions of size 2, within window of size 3

don't within +/-3 know within +/-3 Verb at -1

words: know to, to laugh

to within +/-3 Verb to, to Verb to a +1

pos words: laugh, to

Learning Approach: Representation

S= I don't know whether to laugh or cry

Is represented as a set of its active features

S= (don't at -2 , know within +/-3,... ____to Verb,...)

Label= the confusion set element that occurs in the text

Hope: S=I don't care whether to laugh or cry

has almost the same representation

This representation can be used by any propositional learning algorithm. (features, examples)

Previous works: TBL (Decision Lists) NB, SNoW, DT,...

Notes on Representation

- There is a huge number of potential features (~10^5).
- Out of these – only a small number is actually active in each example.
- The representation can be significantly smaller if we list only features that are active in each examples.
- Some algorithms can take this into account. Some cannot. (Later).

Notes on Representation (2)

- Formally:

  A feature =a characteristic function over sentences

  $$\chi : S \rightarrow \{0,1\}$$
- When the number of features is fixed, the collection of all examples is $\{(\chi_1, \chi_2, ..., \chi_n)\} \equiv \{0,1\}^n$
- When we do not want to fix the number of features (very large number, on-line algorithms,...) can work in the infinite attribute domain

  $$\{(\chi_1, \chi_2, ..., \chi_n, ...)\} \equiv \{0,1\}^\infty$$

An Algorithm

Consider all training data S: {(l, f, f,....)}

Represent as:

S={(f, #(l=0), #(l=1)} for all features

1. Choose best feature f* (and the label it suggests)
2. S ← S \ {Examples labeled in (1)}
3. GoTo 1

An Algorithm:Hypothesis

If f1 then label
Else, if f2 then label
Else…
Else default label

A decision list

Issues: How well will this do?
We train on the training data, what about new data?

Generalization

I saw the girl it the park
The from needs to be completed
I maybe there tomorrow

- New sentences you have not seen before. Can you recognize and correct it this time?
- Intuitively, there are some regularities in the language, "identified" from previous examples, which can be utilized on future examples.
- Two technical ways to formalize this intuition

1: Direct Learning

Model the problem of text correction as a problem of learning from examples.
Goal: learn directly how to make predictions.

PARADIGM

- Look at many examples.
- Discover some regularities in the data.
- Use these to construct a prediction policy.
- A policy (a function, a predictor) needs to be specific.
  [it/in] rule: if the occurs after the target ⇒in

2: Generative Model

Model the problem of text correction as that of generating correct sentences.
Goal: learn a model of the language; use it to predict.

PARADIGM

- Learn a probability distribution over all sentences

In practice: make assumptions on the distribution's type

- Use it to estimate which sentence is more likely.

Pr(I saw the girl it the park) ◇ Pr(I saw the girl in the park)

In practice: a decision policy depends on the options

Example: Model of Language

Model 1: There are 5 characters, A, B, C, D, E
At any point can generate any of them, according to:
P(A)= 0.3; P(B) =0.1; P(C) =0.2; P(D)= 0.2; P(E)= 0.1
P(END)=0.1

- Graphical representation: A sunflower model
- A sentence in the language: AAACCCDEABB.
- A less likely sentence: DEEEEBBBBEEEEBBBBEEE
- Given the model, can compute the probability of a sentence, and decide which is more likely.

Example: Model of Language

Model 2: A probabilistic finite state model.
Start: $P_s(A)=0.4$; $P_s(B)=0.4$; $P_s(C)=0.2$
From A: $P_A(A)=0.5$; $P_A(B)=0.3$; $P_A(C)=0.1$; $P_A(S)=0.1$
From B: $P_B(A)=0.1$; $P_B(B)=0.4$; $P_B(C)=0.4$; $P_B(S)=0.1$
From C: $P_C(A)=0.3$; $P_C(B)=0.4$; $P_C(C)=0.2$; $P_C(S)=0.1$

Practical issues (both paradigms):
What is the space over which we define the model?
Characters? Words? Ideas?
How do we acquire the model?

Learning Paradigms: Comments

- No difference between probabilistic/deterministic or statistical/symbolic. Both paradigms can do both.
- The difference is in the basic assumptions underlying the paradigms, and why they work.
  - 1st: uncover regularities in the past; hope they will be the in the future.
  - 2nd: Know the probabilistic model of the language.
- Direct Learning vs. Generative: major philosophical debate in learning.

Direct Learning: Formalism

- Goal: discover some regularities from examples and generalize to previously unseen data.

  What are the examples we learn from?
- Instance Space X: The space of all examples

  $X = \{0,1\}^n$ or $\{0,1\}^\infty$

  How do we represent our hypothesis?
- Hypothesis Space H: Space of potential functions

  $h : X \rightarrow \{0,1\}$
- Goal: given training data $S \subset X$, find a good $h \in H$

Why Does Learning Work?

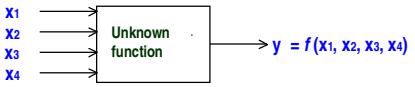
1. Learning is impossible, unless....
2. Outcome of Learning cannot be trusted, unless
3. How can we quantify the expected generalization?

   Assume h is good on the training data; what can be said on h's performance on previously unseen data?

These are some of the topics studied in Computational Learning Theory (COLT)

(notice: mode of interaction is also important)

Learning is impossible, unless…



Given:

Training examples (x,f(x)) of unknown function f

Find:

A good approximation to f

Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Why Does Learning Work (2)?

Complete Ignorance: There are 2^{16} = 56536 possible functions over four input features.

We can't figure out which one is correct until we've seen every possible input-output pair.

- Even after seven examples we still have 2^9 possibilities for f
- Is Learning Possible?

Example	x_1	x_2	x_3	x_4	y
	0	0	0	0	?
	0	0	0	1	?
	0	0	1	0	0
	0	0	1	1	1
	0	1	0	0	0
	0	1	0	1	0
	0	1	1	0	0
	0	1	1	1	?
	1	0	0	0	?
	1	0	0	1	1
	1	0	1	0	?
	1	0	1	1	?
	1	1	0	0	0
	1	1	0	1	?
	1	1	1	0	?
	1	1	1	1	?

Hypothesis Space

Simple Rules: There are only 16 simple conjunctive rules of the form $y=x_i \wedge x_j \wedge x_k$

Try to learn a function of this form that explains the data. (try it: there isn't).

m-of-n rules: There are 29 possible rules of the form "y = 1 if and only if at least m of the following n variables are 1" (try it, there is).

Bias

- Learning requires guessing a good, small hypothesis class.
- We can start with a very small class and enlarge it until it contains an hypothesis that fits the data. (model selection)
- We could be wrong !

Can We Trust the Hypothesis?

There is a hidden conjunction the learner is to learn

$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

How many examples are needed to learn it ? How ?

Protocol: Some random source (e.g., Nature) provides training examples;
Teacher (Nature) provides the labels (f(x))

<(1,1,1,1,1,1,...,1,1), 1>
<(1,1,1,1,1,0,...0,1,1), 1>
<(1,1,1,1,1,0,...0,0,1), 1>
<(1,1,1,1,1,1,...,0,1), 1>

<(1,1,1,0,0,0,...,0,0), 0>
<(1,0,1,1,1,0,...0,1,1), 0>
<(1,0,1,0,0,0,...0,1,1), 0>
<(0,1,0,1,0,0,...0,1,1), 0>

Learning Conjunction

- Algorithm: Elimination

Start with the set of all literals as candidates
Eliminate a literal if not active (0) in a positive example

<(1,1,1,1,1,1,...,1,1), 1> $f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$
<(1,1,1,0,0,0,...,0,0), 0> learned nothing
<(1,1,1,1,1,0,...0,1,1), 1> $f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{99} \wedge x_{100}$
<(1,0,1,1,0,0,...0,0,1), 0> learned nothing
<(1,1,1,1,1,0,...0,0,1), 1> $f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$
<(1,0,1,0,0,0,...0,1,1), 0>
<(1,1,1,1,1,1,...,0,1), 1>
<(0,1,0,1,0,0,...0,1,1), 0> Final: $f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

Prototypical Learning Scenario

- Instance Space: X
- Hypothesis Space: H (set of possible hypotheses)
- Training instances S:
  positive and negative examples of the target f

S: sampled according to a fixed, unknown, probability distribution D over X

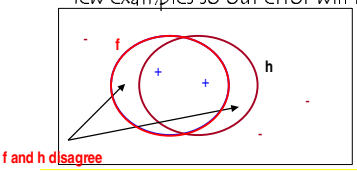
- Determine: A hypothesis $h \in H$ such that
  $h(x) = f(x)$ for all $x \in S$?
  $h(x) = f(x)$ for all $x \in X$?

Evaluated on future instances sampled according to D

$f = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$

PAC Learning: Intuition

- Have seen many examples (drawn according to D)
- Since in all the positive examples x_1 was active, it is likely to be active in future positive examples
- If not, in any case, in D is active only in a relatively few examples so our error will be small.



$Error_D = Pr_{x \in D}[f(x) \neq h(x)]$

Error can be bounded Via Chernoff bounds

A distribution free notion!

Learning Theory: Setting

- Labeled observations $S = \{(x,l)\}_{i=1}^{m}$
  sampled according to a distribution D on $X \times \{0,1\}$
- Goal: to compute a hypothesis $h \in H$ that performs well on future, unseen observations.
- Assumption: test examples are also sampled according to D (label is not observed)

Learning Theory:Why does it work?

- Look for $h \in H$ that minimizes the true error

$$Err_D(h) = Pr_{(x,l)\in D}[h(x) \neq l]$$

- All we get to see is the empirical error

$$Err_S(h) = |\{x \in S \mid h(x) \neq l\}| / |S|$$

- Basic theorem: With probability at least $(1-\delta)$

$$Err_D(h) < Err_S(h) + \sqrt{[kVC(H) + \ln(1/\delta))]/m}$$

Practical Lesson

- Use Hypothesis Space with small expressivity
- E.g. prefer to use a function that is linear in the feature space, over higher order functions

$$f(x) = \sum_i c_i \chi_i(x)$$

- VC dimension of a linear function of dimension N: is N+1
- Sparsity: If there are a maximum of k active in each example then VC dimension is k+1
- Algorithmic issues: There are good algorithms for linear function; learning higher order functions is computationally hard.

2: Generative Model

Model the problem of text correction as that of generating correct sentences.

Goal: learn a model of the language; use it to predict.

PARADIGM

- Learn a probability distribution over all sentences

In practice: make assumptions on the distribution's type

- Use it to estimate which sentence is more likely.

Pr(I saw the girl it the park) ◇ Pr(I saw the girl in the park)

In practice: a decision policy depends on the options

Error Driven Learning

Consider a distribution D over space XxY
Think about X as the instance space, Y the set of labels.

- Given a sample $\{(x,y)\}_1^m$, and a loss function L(x,y)
- Find $h \in H$ that minimizes $\sum_1^m L(h(x), y)$
-
- L can be: L(a,b)=1, $a \neq b$, o/w L(a,b) = 0 (0-1 loss)

$L(a,b) = (a-b)^2$, (L_2)

Need to find an algorithms that minimizes average loss; know that things will be okay (as a function of H).

Basics of Bayesian Learning

- Goal: find the best hypothesis from some space H of hypotheses, given the observed data D.
- Define best to be: most probable hypothesis in H
- In order to do that, we need to assume a probability distribution over the class H.
- In addition, we need to know something about the relation between the data observed and the hypotheses (E.g., a coin problem.)

Basics of Bayesian Learning

- P(h) - the prior probability of a hypothesis h
  Reflects background knowledge; before data is observed. If no information - uniform distribution.
- P(D) - The probability that this sample of the Data is observed.(No knowledge of the hypothesis)
- P(D|h): The probability of observing the sample D, given that the hypothesis h holds
- P(h|D): The posterior probability of h. The probability h holds, given that D has been observed.

Bayes Theorem

$$P(h \mid D) = P(D \mid h)\, P(h) / P(D)$$

- P(h|D) increases with P(h) and with P(D|h)
- P(h|D) decreases with P(D)

Learning Scenario

$$P(h \mid D) = P(D \mid h)\, P(h) / P(D)$$

- The learner considers a set of candidate hypotheses H, and attempts to find the most probable one h ∈H, given the observed data.
- Such maximally probable hypothesis is called maximum a posteriori hypothesis (MAP); Bayes theorem is used to compute it:

$$h_{MAP} = \mathrm{argmax}_{h \in H} P(h \mid D) = \mathrm{argmax}_{h \in H} P(D \mid h)\, P(h) / P(D)$$
$$= \mathrm{argmax}_{h \in H} P(D \mid h) P(h)$$

Learning Scenario (2)

$$h_{MAP} = \mathrm{argmax}_{h \in H} P(h \mid D) = \mathrm{argmax}_{h \in H} P(D \mid h) P(h)$$

- We may assume that a priori, hypotheses are equally probable $P(h_i) = P(h_j), \forall h_i, h_j \in H$
- We get the Maximum Likelihood hypothesis:

$$h_{ML} = \mathrm{argmax}_{h \in H} P(D \mid h)$$

- Here we just look for the hypothesis that best explains the data

Examples

$$h_{MAP} = \mathrm{argmax}_{h \in H} P(h \mid D) = \mathrm{argmax}_{h \in H} P(D \mid h) P(h)$$

- A given coin is either fair or has a 60% bias in favor of Head.
- Goal: decide what is the bias of the coin

Basics of Bayesian Learning

- P(h) - the prior probability of a hypothesis h Reflects background knowledge; before data is observed. If no information - uniform distribution.
- P(D) - The probability that this sample of the Data is observed.(No knowledge of the hypothesis)

- P(D|h): The probability of observing the sample D, given that the hypothesis h holds
- P(h|D): The posterior probability of h. The probability h holds, given that D has been observed.

Example

- Two hypotheses: h_1: P(H)=0.5; h_2: P(H)=0.6

- P(h): $P(h_1)=0.75$ $P(h_2)=0.25$
- Now we need Data; Experiment. 1st coin toss is H.

- P(D|h): $P(D|h_1)=0.5$; $P(D|h_2)=0.6$
- P(D): $P(D)=P(D|h_1)P(h_1)+P(D|h_2)P(h_2)=$
  $0.5 \bullet 0.75 + 0.6 \bullet 0.25 = 0.525$
- P(h|D):
  $P(h_1|D)=P(D|h_1)P(h_1)/P(D)=0.5 \bullet 0.75/0.525=0.714$
  $P(h_2|D)=P(D|h_2)P(h_2)/P(D)=0.6 \bullet 0.25/0.525=0.286$

Example (2)

- Two hypotheses: h_1: P(H)=0.5; h_2: P(H)=0.6
- P(h): $P(h_1)$=0.75 $P(h_2)$=0.25
- Now we need Data; Experiment. 1st coin toss is H.
- We still think that the coin is more likely to be fair
- If we were to use Maximum Likelihood approach (i.e., assume equal priors) we would think otherwise. The data supports the biased coin better.

Try: 100 coin tosses; 70 heads.
Now you will believe that the coins is biased.

Learning a Concept Class

- Assume that we are given a concept class C.
- Given a collection of examples (x,f(x)),
- for $f \in C$, we try to identify f.
- What will the Bayesian approach tell us ?

Learning a Concept Class

- P(h): the prior probability of a hypothesis h:
- p(h) = 1/|H| for all h in H
- P(D|h): Let d=(x,l) be the observed labeled example
  P((x,l)|h)= 1 (h(x)=l), 0 (w/o)
- P(D): For a set of examples $\mathbf{P(D) = \sum_{h_i \in H} P(D \mid h_i) P(h_i) = \frac{|H_{CON}|}{|H|}}$
- where H_{CON} is the set of hypotheses in Hwhich are consistent with the sample D
- P(h|D): via Bayes rule

$$\mathbf{P(h \mid D) = P(D \mid h) P(h) / P(D) = \begin{cases} 1/|H_{CON}| & \text{if h is consistent with D} \\ 0 & \text{othewise} \end{cases}}$$

Maximum Likelihood Estimate

- Assume that you toss a (p,1-p) coin m times and get k Heads, m-k Tails. What is p?

Maximum Likelihood Estimate

- Assume that you toss a (p,1-p) coin m times and get k Heads, m-k Tails. What is p?
- If p is the probability of Tail, the probability of the data observed is: $\mathbf{P(D \mid p) = p^k (1-p)^{m-k}}$

Maximum Likelihood Estimate

- Assume that you toss a (p,1-p) coin m times and get k Heads, m-k Tails. What is p?
- If p is the probability of Tail, the probability of the data observed is: $\mathbf{P(D \mid p) = p^k (1-p)^{m-k}}$
- The Log Likelihood:

$$\mathbf{\log(L(D \mid p)) = \log(P(D \mid p)) = \log p^k (1-p)^{m-k} = k \log p + (m-k) \log(1-p)}$$

Maximum Likelihood Estimate

- Assume that you toss a (p,1-p) coin m times and get k Heads, m-k Tails. What is p?
- If p is the probability of Tail, the probability of the data observed is: $P(D|p) = p^k(1-p)^{m-k}$
- The Log Likelihood:

$$\log(L(D|p)) = \log(P(D|p)) = \log p^k(1-p)^{m-k} =$$
$$= k \log p + (m-k)\log(1-p)$$

- To maximize, set the derivative w.r.t. p equal to 0:

$$\frac{d(\log L(D|p))}{dp} = \frac{k}{p} - \frac{m-k}{1-p} = 0$$

Maximum Likelihood Estimate

- Assume that you toss a (p,1-p) coin m times and get k Heads, m-k Tails. What is p?
- If p is the probability of Tail, the probability of the data observed is: $P(D|p) = p^k(1-p)^{m-k}$
- The Log Likelihood:

$$\log(L(D|p)) = \log(P(D|p)) = \log p^k(1-p)^{m-k} =$$
$$= k \log p + (m-k)\log(1-p)$$

- To maximize, set the derivative w.r.t. p equal to 0:

$$\frac{d(\log L(D|p))}{dp} = \frac{k}{p} - \frac{m-k}{1-p} = 0$$

- Solving this for p, gives: p=k/m

Bayes Optimal Classifier

- How should we use the general formalism?
- What should H be?

- H can be a collection of functions. Given the training data, choose an optimal function. Then, given new data, evaluate the selected function on it.
- H can be a collection of possible predictions. Given the data, try to directly choose the optimal prediction.

- Could be different!

Disambiguation

He was reluctant to [accept/except] it.

The first formalisms suggest to learn a good hypothesis:

$$h_{MAP} = \mathbf{argmax}_{h \in H} P(h|D) = \mathbf{argmax}_{h \in H} P(D|h)P(h)$$

And use it.
(Language modeling, grammar learning, etc. are here)

The second one suggests to directly choose a hypothesis – [accept or except]: study of naïve Bayes.

Learning Paradigms

We have looked at two learning paradigms – direct learning and the Bayesian learning approach.

Next, we will establish some more relations between them and use it to motivate some algorithmic approaches.

Statistical Methods

Decision are derived by
- assuming a generative language model for the data

$$x = (t_1, t_2, ..., t_n)$$

- estimate its parameters

$$Pr(x) = Pr(t_1, t_2, ..., t_n) = \prod_{i=1}^{n} Pr(t_i | t_1, t_2, ..., t_{i-1}) =$$
$$= \prod_{i=1}^{n} Pr(t_i | h_{i-1})$$

Difficulty: estimating the terms

Statistical Methods

Middle Eastern ____ are known for their sweetness

Estimating

Pr(deserts | context) and Pr(desserts | context)

Statistical Methods

[x x x x] Middle Eastern ____ are known for their sweetness

Estimating

Pr(context | deserts) and Pr(context | desserts)

A representation-less procedure

Probabilistic assumptions are typically wrong

Requires making some probabilistic assumptions

Typically, Markov or other independence assumptions

Otherwise - impossible to get robust estimation

Thesis: Why does it work ?

Statistical methods learn simple and robust hypotheses that fit the data

- Representation:
  Decisions surface of low expressive power. (linear)
- Algorithm: LSQ
  Coefficients are estimated using statistical queries
- Methodology (learning protocol): Fits Data
  Features are chosen so that the linear classifier fits the training data Learning Theory

Learning Theory: Setting

- Labeled observations $S = \{(x,l)\}_{i=1}^{m}$ sampled according to a distribution D on $X \times \{0,1\}$
- Goal: to compute a hypothesis $h \in H$ that performs well on future, unseen observations.
- Assumption: test examples are also sampled according to D (label is not observed)

Learning Theory: Why does it work?

- Look for $h \in H$ that minimizes the true error

$$Err_D(h) = Pr_{(x,l)\in D}[h(x) \neq l]$$

- All we get to see is the empirical error

$$Err_S(h) = |\{x \in S \mid h(x) \neq l\}| / |S|$$

- Basic theorem: With probability at least $(1-\delta)$

$$Err_D(h) < Err_S(h) + \sqrt{[kVC(H) + \ln(1/\delta)]/m}$$

Linear Statistical Queries Hypothe

The STAT NLP hypotheses: Representation and Algorithm

eses with good properties and additional robustness

- Representation:
  Linear over a commonly used set of features
- Algorithm:
  Coefficients are statistics on the training sample

Linear Statistical Queries Hypotheses

- Restricted class of hypotheses with good generalization properties and additional robustness
- Representation:
  Linear over a commonly used set of features
- Algorithm:
  Coefficients are statistics on the training sample

LSQ: Representation

- Features are indicator functions $\chi : X \to \{0,1\}$
  - define subsets of the instance space

Eastern___: Eastern occurs before target word

$$x = (x_1, x_2, ..., x_n) = ((w_1, t_1), (w_2, t_2), ..., (w_n, t_n))$$

χ_1 : the condition $\exists i (w_i = \text{good}, w_{i+1} = \text{talk})$
is active ($\chi_1(x) = 1$) in x= "is this a good talk"

χ_2 : the condition $\exists i (w_i = \text{talk}, t_i = \text{verb})$
is active ($\chi_2(x) = 1$) in x= "It's good to talk to you"

LSQ: Representation (2)

- Features are indicator functions $\chi : X \to \{0,1\}$
  - define subsets of the instance space

The collection **Z** of features maps the instance space into a feature space:

$$x = (x_1, x_2, ..., x_n) \to (\chi_1, \chi_2, ... \chi_{|Z|}) \in \{0,1\}^{|Z|}$$

The **LSQ** hypothesis is linear in the feature space:

$$f(x) = \sum_Z c_i \chi_i(x)$$

LSQ: Algorithm

- The coefficients of: $f(x) = \sum c_i \chi_i(x)$
  are computed via Statistical Queries (Kearns'93)
- Statistical query: $[\chi, l, \tau]$: A τ-good estimate of
  $\Pr^D_{[\chi,l]} = \Pr_D\{(x,i) \mid \chi(x) = 1 \wedge i = l\}$
  computed on the sample S
- An algorithm is said to be a good **LSQ** learning algorithm if it produces an **LSQ** hypothesis with small empirical error using a polynomial size sample.

Example

- Given a sample: $S \subseteq \{0,1\}^n \times \{0,1\}$
- Let the n+1 features be: $\chi_0(x) = 1, \chi_i(x) = x_i, \quad i = 1,...,n$
- Set the coefficients for labels l=0,1:

$$c_{[\chi_0,l]} = \log \hat{Pr}^D_{[\chi_0,l]} = \log |\{(x,j) \mid j = l\}| / |S|$$

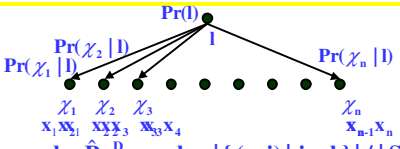
$$c_{[\chi,l]} = \log \hat{Pr}^D_{[\chi,l]} / \log \hat{Pr}^D_{[\chi_0,l]} = \log \frac{|\{(x,j) \mid \chi(x) = 1 \wedge j = l\}|}{|\{(x,j) \mid j = l\}|}$$

$$\text{prediction}(x) = \text{argmax}_{\{l=0,1\}} \sum_{i=0}^{n} c_{[x_i,l]} \chi_i(x)$$

Basic Probability

- Product Rule: $P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$
  If A and B are independent
  $P(A \wedge B) = P(A)P(B)$; $P(A|B) = P(A)$, $P(A|B,C) = P(A|C)$
- Sum Rule: $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
- Bayes Rule: $P(A|B) = P(B|A)\,P(A)/P(B)$
- $A_1, A_2, ..., A_i$ mutually exclusive with $\sum_i P(A_i) = 1$
- Total Probability:
  $$P(B) = \sum_i P(B \wedge A_i) = \sum_i P(B \mid A_i) P(A_i)$$
- Total Conditional Probability:
  $$P(B \mid C) = \sum_i P(B \wedge A_i \mid C) = \sum_i P(B \mid A_i, C) P(A_i \mid C)$$

Example: naive Bayes



$$c_{[\chi_0,l]} = \log \hat{Pr}^D_{[\chi_0,l]} = \log |\{(x,j) \mid j = l\}| / |S|$$

$$c_{[\chi,l]} = \log \hat{Pr}^D_{[\chi,l]} / \log \hat{Pr}^D_{[\chi_0,l]} = \log \frac{|\{(x,j) \mid \chi(x) = 1 \wedge j = l\}|}{|\{(x,j) \mid j = l\}|}$$

$$prediction(x) = \text{argmax}_{\{l=0,1\}} \sum_{i=0}^{n} c_{[x_i,l]} \chi_i(x)$$

Linear Statistical Queries

Generalization:

The VC dimension of the hypotheses class is bounded by the number of features used.

$$Err_D(h) < Err_S(h) + \sqrt{[kVC(H) + \ln(1/\delta)]/m}$$

Robustness:

A good **LSQ** learning algorithm produced for **D**, is also a good learning algorithm over distribution **D'**.

d(D,D') is related to the algorithm's tolerance.

If trained on **D**, can safely test on **D'**.

Robustness of LSQ Hypotheses

Distributional Distance:

$$d_X(D,D') = \max_{\chi \in X} |D(\chi) - D'(\chi)| \leq \max_{A \subseteq X} |D(A) - D'(A)|$$

Quality of learned hypotheses:

$$|\hat{P}^D_\chi - P^{D'}_\chi| \leq |\hat{P}^D_\chi - P^D_\chi| + |P^D_\chi - P^{D'}_\chi| \leq (\tau - \gamma) + \gamma = \tau$$

Robustness:

Depends on Tolerance

- size of sample
- expressiveness of the features used.

Example: Markov Models

Input: $x = ((w_1 : t_1), \ldots (w_{i-1} : t_{i-1}), (w_i : ?), (w_{i+1} : t_{i+1}) \ldots)$

States: t_{i-1} t t_{i+1} T

Observations: w_{i-1} w_i w_{i+1} W

Assumptions: $Pr(w_i \mid t_i) = Pr(w_i \mid x)$

$Pr(t_{i+1} \mid t_i) = Pr(t_{i+1} \mid t_i, t_{i-1}, \ldots, t_1)$

Prediction: predict $t \in T$ that maximizes

$$Pr(t \mid t_{i-1}) \bullet Pr(w_i \mid t) \bullet Pr(t_{i+1} \mid t)$$

Example: Markov Models

Input: $x = ((w_1 : t_1), \ldots (w_{i-1} : t_{i-1}), (w_i : ?), (w_{i+1} : t_{i+1}) \ldots)$

States: t_{i-1} t t_{i+1} T

Observations: w_{i-1} w_i w_{i+1} W

LSQ Algorithm: features are pairs and singletons of t's, w's

$$c_{[t_1 \times t_2, t_1]} = \log \hat{Pr}^D_{[t_1 \times t_2, t_1]} / \log \hat{Pr}^D_{[1,t_1]} \quad c = \log \frac{|\{(t,t') \mid t = t_1 \wedge t' = t_2\}|}{|\{(t,t') \mid t = t_1\}|} \quad \text{Otherwise } c = 0$$

$$prediction(x) = \text{argmax}_{[t \in T]} \sum_{\chi \in X} c_{[x,t]} \chi(x)$$

Only 3 active features

Learning with Probabilistic Classifiers

- Learning Theory $Err_S(h) = |\{x \in S \mid h(x) \neq l\}| / |S|$
- Linear Statistical Queries Models
- Properties: Generalization+Robustness
  - naïve Bayes
  - Markov Models

⇨ Is this enough to explain it all?

Learning Protocol

- **LSQ** hypotheses are computed directly, w/o assumptions on the underlying distribution:
  - Choose features
  - Compute coefficients

◇ Is there a reason to believe that an **LSQ** hypothesis minimizes the empirical error on the sample?

◇ In General, No.

(Unless it corresponds to some probabilistic assumptions that hold).

Example: Conjunctions with naive Bayes

- **D** – a distribution over $\{0,1\}^{100}$

  Examples are labeled using a simple conjunction:

$$x_1 \wedge x_2 \wedge x_{28}$$

◇ Learning: **LSQ** /naïve Bayes

- **D** can be defined so that the empirical error is as close to **1/2** as you can.

Learning Protocol: Practice

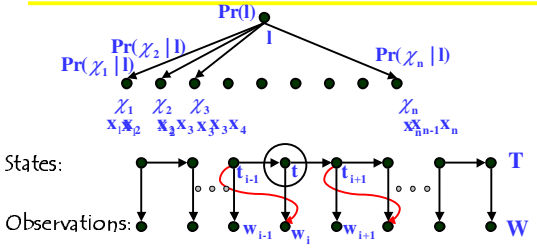
- **LSQ** hypotheses are computed directly:
  - Choose features
  - Compute coefficients

◇ If hypothesis does not fit the training data –

– Augment set of features

(Forget your original assumption)

Example: probabilistic classifiers



Features are pairs and singletons of t's, w's

Additional features are included

Learning Protocol: Practice

- **LSQ** hypotheses are computed directly:
  - Choose features
  - Compute coefficients

◇ If hypothesis does not fit the training data –

– Augment set of features so it does

◇ Learning Theory explains good behavior

◇ Generalization and Robustness of **LSQ** across distributions come into play.

Happens regardless of whether the probabilistic assumptions hold

Not the whole Story: Robustness of Probabilistic Predictors

◇ Consider all distributions with the same marginals

$$\Pr(\chi_i \mid l)$$

(E.g, a naïve Bayes classifier will predict the same regardless of which distribution generated the data.)

(Garg&Roth ECML'01):

In most cases (i.e., for most such distributions), the resulting predictor's error is close to optimal classifier (that if given the correct distribution)

Summary: Probabilistic Modeling

- Classifiers derived from probability density estimation models were viewed as **LSQ** hypotheses.

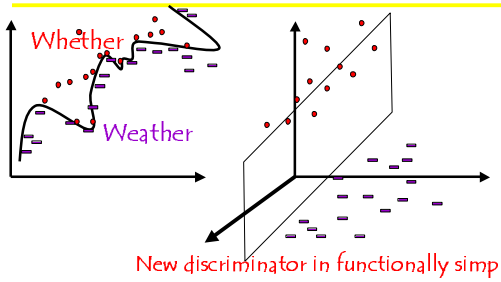
$$\sum c_i \chi_i(x) = \sum c_i \prod x_{i_1} x_{i_2} \ldots x_{i_k}$$

- Probabilistic assumptions:
  - \+ Guiding feature selection but also –
  - – Not allowing the use of more general features.

Plan

- Learning with probabilistic classifiers
  - Why does it work ?
  - A Learning Theory Account
  - Linear Statistical Queries Models
- A unified framework for Learning in NL
  - Algorithmic Approaches

Embedding



New discriminator in functionally simpler

$x_1 x_2 \bar{x}_3 \vee \bar{x}_1 x_4 \bar{x}_3 \vee x_3 \bar{x}_2 x_5 \qquad y_1 \vee y_4 \vee y_5$

A Unified Approach

- Most methods blow up original feature space.

$$X(x_1, x_2, x_3, \ldots x_k) \rightarrow X(\chi_1(x), \chi_2(x), \chi_3(x) \ldots \chi_n(x)) \qquad n >> k$$

- And make predictions using a linear representation over the new feature space

$$\arg\max_j \sum_i c_i^j \chi_i(x)$$

Note: Methods do not have to actually do that; But: they produce same decision as a hypothesis that does that. (Roth 98; 99,00)

A Unified Approach

- Most methods blow up original feature space.

$$X(x_1, x_2, x_3, \ldots x_k) \rightarrow X(\chi_1(x), \chi_2(x), \chi_3(x) \ldots \chi_n(x)) \qquad n >> k$$

- And make predictions using a linear representation over the new feature space

$$\arg\max_j \sum_i c_i^j \chi_i(x)$$

- Probabilistic Methods
- Rule based methods (TBL; decision lists; exponentially decreasing weights)
- Linear representation (SNoW; Perceptron; SVM; Boosting)
- Memory Based Methods (subset features)

A Unified Approach

- Most methods blow up original feature space.

$$X(x_1, x_2, x_3, \ldots x_k) \rightarrow X(\chi_1(x), \chi_2(x), \chi_3(x) \ldots \chi_n(x)) \qquad n >> k$$

- And make predictions using a linear representation over the new feature space

$$\arg\max_j \sum_i c_i^j \chi_i(x)$$

Q 1: How are weights determined?
Q 2: How is the new feature-space determined?
Implications? Restrictions?

Algorithmic Approaches

- Focus: Two families of algorithms
  (will discuss the on-line representative)
- Additive update algorithms: Perceptron

  SVM (not on-line, but a close relative of Perceptron)
- Multiplicative update algorithms: Winnow

  SNoW

  Close relatives: Boosting; Max Entropy

Algorithm Descriptions

Examples : $x \in \{0,1\}^n$; Hypothesis : $w \in R^n$

Prediction is 1 iff $w \bullet x \geq \theta$

- Additive weight update algorithm
  (Perceptron, Rosenblatt, 1958. Variations exist)

If Class = 1 but $w \bullet x \leq \theta$, $w_i \leftarrow w_i + 1$ (if $x_i = 1$) (promotion)
If Class = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i - 1$ (if $x_i = 1$) (demotion)

- Multiplicative weight update algorithm
  (Winnow, Littlestone, 1988. Variations exist)

Relative Entropy based

If Class = 1 but $w \bullet x \leq \theta$, $w_i \leftarrow 2w_i$ (if $x_i = 1$) (promotion)
If Class = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ (if $x_i = 1$) (demotion)

How to Compare?

- Generalization
  (since the representation is the same)
  How many examples are needed
  to get to a given level of accuracy?
- Efficiency
  How long does it take to evaluate
  a hypothesis?
- Robustness; Adaptation to a new domain, ...

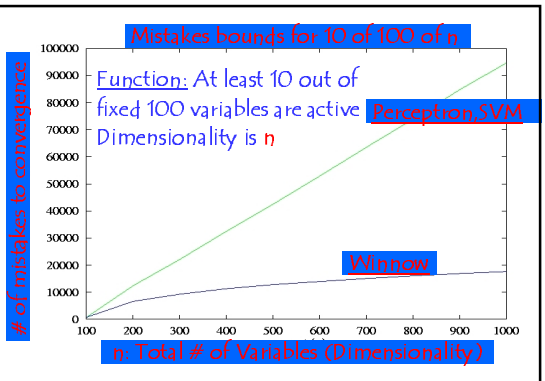
Learning in NLP: Characteristics

- The number of potential features is very large
- The instance space is sparse
- Decisions depend on a small set of features (sparse)
- Want to learn from a number of examples that is small relative to the dimensionality

Generalization

- Dominated by the sparseness of the function space
  Most features are irrelevant

  Advantage multiplicative: # of examples required depends mostly on # of relevant features
  (Generalization bounds depend on $\|w\|_1$)
- Lesser issue: Sparseness of features space:
  Advantage additive. Generalization depend on $\|x\|$
  (Kivinen/Warmuth 95)



Efficiency

- Dominated by the size of the feature space
- Most features are functions (e.g., n-grams) of raw attributes

$$X(x_1, x_2, x_3, \ldots x_k) \rightarrow X(\chi_1(\mathbf{x}), \chi_2(\mathbf{x}), \chi_3(\mathbf{x}) \ldots \chi_n(\mathbf{x})) \qquad \mathbf{n >> k}$$

- Additive algorithms allow the use of Kernels
  No need to explicitly generate the complex features

$$\sum c_i K(\mathbf{x}, \mathbf{x}_i)$$

- Could be more efficient since work is done in the original feature space.

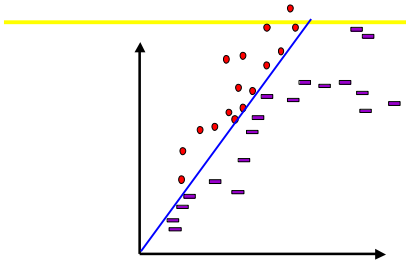
Multiplicative vs. Additive

- The kernel idea is unlikely to be available in multiplicative update algorithms.
- On the other hand, Kernels help only computationally, and do not, in general, help Generalization.
- Generalization still depends on the blown up dimensionality.
- [Khardon, Roth Servedio, 2001]
- Next, some intuition for these algorithms.

Linear Functions

$$f(x) = \begin{cases} 1 & \text{if } w_1 x_1 + w_2 x_2 + \ldots w_n x_n >= \theta \\ 0 & \text{Otherwise} \end{cases}$$

- Disjunctions: $y = x_1 \vee x_3 \vee x_5$
  $y = (1 \cdot x_1 + 1 \cdot x_3 + 1 \cdot x_5 >= 1)$
- At least m of n: $y =$ at least 2 of $\{x_1, x_3, x_5\}$
  $y = (1 \cdot x_1 + 1 \cdot x_3 + 1 \cdot x_5 >= 2)$
- Exclusive-OR: $y = (x_1 \wedge \overline{x_2}) \vee (\overline{x_1} \wedge x_2)$
- Non-trivial DNF: $y = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$



Learning Linear Threshold Functions

- Perceptron
- Winnow
- Both can be viewed as learning a continuous functions, but also as learning a Boolean function (symbolic learning…)

Perceptron for Boolean Functions

- How many mistakes will the Perceptron algorithms make when learning a k-disjunction?
- Find a sequence of examples that will cause Perceptron to make O(n) mistakes on k-disjunction on n attributes.

Winnow Algorithm

Initialize : $\theta = n$; $w_i = 1$
Prediction is 1 iff $w \bullet x \geq \theta$
If no mistake : do nothing
If f(x) = 1 but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ **(if** $x_i = 1$**) (promotion)**
If f(x) = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ **(if** $x_i = 1$**) (demotion)**

- The Winnow Algorithm learns Linear Threshold Functions.
- For the class of disjunction, instead of demotion we can use elimination.

Winnow - Example

$f = x_1 \vee x_2 \vee x_{1023} \vee x_{1024}$
Initialize : $\theta = 1024$; $w = (1,1,...,1)$
< (1,1,...,1),+ > $w \bullet x \geq \theta$ $w = (1,1,...,1)$ ok
< (0 ,0,...,0),- > $w \bullet x \leq \theta$ $w = (1,1,....,1)$ ok
< (0,0,111,,,,0),– > $w \bullet x \leq \theta$ $w = (1,1,....,1)$ ok
< (1,0,0,...,0),+ > $w \bullet x \leq \theta$ $w = (2,1,....,1)$ mistake
< (1,0,1,1,0..,0),+ > $w \bullet x \leq \theta$ $w = (4,1,2,2...,1)$ mistake
< (1,0,1,0,0...1),+ > $w \bullet x \leq \theta$ $w = (8,1,4,2...,2)$ mistake
..... log(n/2) (for each good variable)
$w = (512,1,256,256,....,256)$

< (1,0,1,0...,1),+ > $w \bullet x \geq \theta$ $w = (512,1,256,256,....,256)$ ok
< (0,0,1,0.111..,0),– > $w \bullet x \geq \theta$ $w = (512,1,0,..0,...,256)$ mistake **(eliminati on version)**
.....
$w = (1024,1024,0,0,0,1,32,...,1024,1024)$ **(final hypothesis)**

Winnow - Example

$f = x_1 \vee x_2 \vee x_{1023} \vee x_{1024}$
Initialize : $\theta = 1024$; $w = (1,1,...,1)$
< (1,1,...,1),+ > $w \bullet x \geq \theta$ $w = (1,1,...,1)$ ok
< (0 ,0,...,0), - > $w \bullet x \leq \theta$ $w = (1,1,....,1)$ ok
< (0,0,111 , , , ,0),– > $w \bullet x \leq \theta$ $w = (1,1,....,1)$ ok
< (1,0,0,.... 0),+ > $w \bullet x \leq \theta$ $w = (2,1,....,1)$ mistake
< (1,0,1,1,0.., 0),+ > $w \bullet x \leq \theta$ $w = (4,1,2,2....,1)$ mistake
< (1,0,1,0,0...,1),+ > $w \bullet x \leq \theta$ $w = (8,1,4,2...., 2)$ mistake
.... log(n/2) (for each good variable)
$w = (512,1,256,256,....., 256)$

< (1,0,1,0....,1),+ > $w \bullet x \geq \theta$ $w = (512,1,256,256,....., 256)$ ok
< (0,0,1,0.111 .., 0),– > $w \bullet x \geq \theta$ $w = (512,1,0,.. 0,...., 256)$ mistake **(eliminati on version)**
....
$w = (1024,1024,0,0,0,1,32,...,1024,1024)$ **(final hypothesis)**

- Notice that the same algorithm will learn a conjunction over these variables (w=(256,256,0,…32,…256,256))

Winnow - Mistake Bound

Claim: Winnow makes O(k log n) mistakes on k-disjunctions

Initialize : $\theta = n$; $w_i = 1$
Prediction is 1 iff $w \bullet x \geq \theta$
If no mistake : do nothing
If f(x) = 1 but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ **(if** $x_i = 1$**) (promotion)**
If f(x) = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ **(if** $x_i = 1$**) (demotion)**

u - # of mistakes on positive examples (promotions)
v - # of mistakes on negative examples (demotions)

Winnow - Mistake Bound

Claim: Winnow makes O(k log n) mistakes on k-disjunctions

Initialize : $\theta = n$; $w_i = 1$
Prediction is 1 iff $w \bullet x \geq \theta$
If no mistake : do nothing
If f(x) = 1 but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ **(if** $x_i = 1$**) (promotion)**
If f(x) = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ **(if** $x_i = 1$**) (demotion)**

u - # of mistakes on positive examples (promotions)
v - # of mistakes on negative examples (demotions)

1. u < k log(2n)

Winnow - Mistake Bound

Claim: Winnow makes O(k log n) mistakes on k-disjunctions

Initialize : $\theta = n$; $w_i = 1$
Prediction is 1 iff $w \bullet x \geq \theta$
If no mistake : do nothing
If f(x) = 1 but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ **(if** $x_i = 1$**) (promotion)**
If f(x) = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ **(if** $x_i = 1$**) (demotion)**

u - # of mistakes on positive examples (promotions)
v - # of mistakes on negative examples (demotions)

1. u < k log(2n)
A weight that corresponds to a good variable is only promoted.
When these weights get to n there will no more mistakes on positives

Winnow - Mistake Bound

Initialize : $\theta = n$; $w_i = 1$
Prediction is 1 iff $w \bullet x \geq \theta$
If no mistake : do nothing
If f(x) = 1 but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ (if $x_i = 1$) (promotion)
If f(x) = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ (if $x_i = 1$) (demotion)

u - # of mistakes on positive examples (promotions)
v - # of mistakes on negative examples (demotions)

2. $v < 2(u + 1)$

Winnow - Mistake Bound

Initialize : $\theta = n$; $w_i = 1$
Prediction is 1 iff $w \bullet x \geq \theta$
If no mistake : do nothing
If f(x) = 1 but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ (if $x_i = 1$) (promotion)
If f(x) = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ (if $x_i = 1$) (demotion)

u - # of mistakes on positive examples (promotions)
v - # of mistakes on negative examples (demotions)

2. $v < 2(u + 1)$
Total weight: TW=n initially

Winnow - Mistake Bound

Initialize : $\theta = n$; $w_i = 1$
Prediction is 1 iff $w \bullet x \geq \theta$
If no mistake : do nothing
If f(x) = 1 but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ (if $x_i = 1$) (promotion)
If f(x) = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ (if $x_i = 1$) (demotion)

u - # of mistakes on positive examples (promotions)
v - # of mistakes on negative examples (demotions)

2. $v < 2(u + 1)$
Total weight: TW=n initially
Mistake on positive: TW(t+1) < TW(t) + n

Winnow - Mistake Bound

Initialize : $\theta = n$; $w_i = 1$
Prediction is 1 iff $w \bullet x \geq \theta$
If no mistake : do nothing
If f(x) = 1 but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ (if $x_i = 1$) (promotion)
If f(x) = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ (if $x_i = 1$) (demotion)

u - # of mistakes on positive examples (promotions)
v - # of mistakes on negative examples (demotions)

2. $v < 2(u + 1)$
Total weight TW=n initially
Mistake on positive: TW(t+1) < TW(t) + n
Mistake on negative: TW(t+1) < TW(t) - n/2

Winnow - Mistake Bound

Initialize : $\theta = n$; $w_i = 1$
Prediction is 1 iff $w \bullet x \geq \theta$
If no mistake : do nothing
If f(x) = 1 but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ (if $x_i = 1$) (promotion)
If f(x) = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ (if $x_i = 1$) (demotion)

u - # of mistakes on positive examples (promotions)
v - # of mistakes on negative examples (demotions)

2. $v < 2(u + 1)$
Total weight TW=n initially
Mistake on positive: TW(t+1) < TW(t) + n
Mistake on negative: TW(t+1) < TW(t) - n/2
$0 < TW < n + u\,n - v\,n/2 \Rightarrow v < 2(u+1)$

Winnow - Mistake Bound

Initialize : $\theta = n$; $w_i = 1$
Prediction is 1 iff $w \bullet x \geq \theta$
If no mistake : do nothing
If f(x) = 1 but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ (if $x_i = 1$) (promotion)
If f(x) = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ (if $x_i = 1$) (demotion)

u - # of mistakes on positive examples (promotions)
v - # of mistakes on negative examples (demotions)

2. $v < 2(u + 1)$
Total weight TW=n initially
Mistake on positive: TW(t+1) < TW(t) + n
Mistake on negative: TW(t+1) < TW(t) - n/2
$0 < TW < n + u\,n - v\,n/2 \Rightarrow v < 2(u+1)$

Winnow - Mistake Bound

Initialize : $\theta = n$; $w_i = 1$

Prediction is 1 iff $w \bullet x \geq \theta$

If no mistake : do nothing

If f(x) = 1 but $w \bullet x < \theta$, $w_i \leftarrow 2w_i$ (if $x_i = 1$) (promotion)

If f(x) = 0 but $w \bullet x \geq \theta$, $w_i \leftarrow w_i/2$ (if $x_i = 1$) (demotion)

u - # of mistakes on positive examples (promotions)

v - # of mistakes on negative examples (demotions)

# of mistakes: u+ v < 3u + 2 = O(k log n)

Winnow - Extensions

- This algorithm learns monotone functions
- For the general case:
  - Duplicate variables
    For the negation of variable x, introduce a new variable y.
    Learn monotone functions over 2n variables
  - Balanced version:

Keep two weights for each variable; effective weight is the difference

Update Rule :

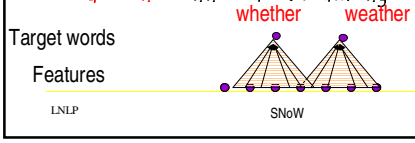
If $f(x) = 1$ but $(w^+ - w^-) \bullet x \leq \theta$, $w_i^+ \leftarrow 2w_i^+$ $w_i^- \leftarrow \frac{1}{2} w_i^-$ where $x_i = 1$ (promotion)

If $f(x) = 0$ but $(w^+ - w^-) \bullet x \geq \theta$, $w_i^+ \leftarrow \frac{1}{2} w_i^+$ $w_i^- \leftarrow 2w_i^-$ where $x_i = 1$ (demotion)

SNoW: has a version of Winnow that supports the general case.

SNoW Learning Architecture

- The most successful approach tried on several NLP problems
- A learning architecture tailored for high dimensional problems
- A network of linear representations
- Here: One for each word in a confusion set (target).
- Prediction: Winner Take All among relevant targets.

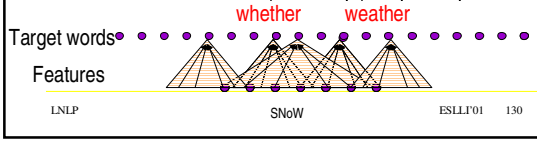


SNoW Learning Architecture

- Several update algorithms are available
- Most successful – a multiplicative update algorithm
- a variation of Winnow (Littlstone'88)
- Learning protocol:

  on-line, mistake driven

  1-vs. all within a confusion set



SNoW

- Update rule: Multiplicative – feature efficient
- Feature space: Infinite Attribute Space $\{0,1\}^\infty$
  - examples of variable size: only active features
  - determined in a data driven way
- Multi Class Learner
- Makes Possible:

  Generation of many complex/relational types of features

  Only a small fraction is actually represented

  Computationally efficient (on-line!)

Work Done

- Context Sensitive Text Correction
  (peace;piece) (among;between)
- Prepositional Phrase Attachment
  (car with..., buy with...)
- Part of Speech Tagging
  (Verb, Noun, Adj,...)
- Shallow Parsing Tasks
  (noun phrases; subject-verb)
- Comparable or Superior in performance and efficiency

  http://L2R.cs.uiuc.edu/~danr/snow.html

Summary (So Far)

- Problems in NLP: modeling, some characteristics
- Learning paradigm: PAC (direct learning; Bayesian)
- Why things work?
- Algorithms:
- Rules, naïve Bayes; Perceptron; Winnow; SNoW
- All these pertain to the supervised learning protocol.

- In some cases, supervision is expensive. What can be done?

Semi-Supervised Learning

- Consider the problem on Prepositional Phrase Attachment.

- There are several ways to generate features. Given the limited representation, we can assume that all possible conjunctions of the 4 attributes are used. (15 feature in each example).
- See other possibilities in [Krymolovsky, Roth 98]
- Assume we will use naïve Bayes for learning to decise between [n,v]
- Examples are: $(x_1,x_2,\ldots x_n,[n,v])$

Using naïve Bayes

- To use naïve Bayes, we need to use the data to estimate: $P(n)$ $P(v)$
  $P(x_1|n)$ $P(x_1|v)$
  $P(x_2|n)$ $P(x_2|v)$
  ……
  $P(x_n|n)$ $P(x_n|v)$
- Then, given an example $(x_1,x_2,\ldots x_n,?)$, compare:
  $P_n(x)=P(n)\ P(x_1|n)\ P(x_2|n)\ldots\ P(x_n|n)$

and

  $P_v(x)=P(v)\ P(x_1|v)\ P(x_2|v)\ldots\ P(x_n|v)$

Using naïve Bayes

- After seeing 10 examples, we have:
- $P(n)=0.5$; $P(v)=0.5$

$P(x_1|n)=0.75; P(x_2|n)=0.5; P(x_3|n)=0.5; P(x_4|n)=0.5$

$P(x_1|v)=0.25; P(x_2|v)=0.25; P(x_3|v)=0.75; P(x_4|v)=0.5$

- Then, given an example (1000), we have:
  $P_n(x)=0.5\ 0.75\ 0.5\ 0.5\ 0.5 = 3/64$
  $P_v(x)=0.5\ 0.25\ 0.75\ 0.25\ 0.5=3/256$

Now, assume that in addition to the 10 labeled examples, we also have 100 unlabeled examples.

Using naïve Bayes

- For example, what can be done with (1000?) ?
- We can guess the label of the unlabeled example…

- But, can we use it to improve the classifier? (that is, the estimation of the probabilities?)
- We can assume the example x=(1000) is a
  - n example with probability $P_n(x)/(P_n(x) + P_v(x))$
  - v example with probability $P_v(x)/(P_n(x) + P_v(x))$

- Estimation of probabilities does not require work with integers!

Using Unlabeled Data

The discussion suggests several algorithms:

1. Use a threshold. Chose examples labeled with high confidence. Labeled them [n,v]. Retrain.

2. Use fractional examples. Label the examples with fractional labels [p of n, (1-p) of v]. Retrain.

Comments on Unlabeled Data

- Both algorithms suggested can be used iteratively.
- Both algorithms can be used with other classifiers, not only naïve Bayes. The only requirement – a robust confidence measure in the classification.
- E.g.: Brill, ACL'01: uses all three algorithms in SNoW for studies of these sort.
- And, what happens if instead of 10 labeled examples we start with 0 labeled examples?
- Make a Guess; continue as above; a version of EM

EM

- EM is a class of algorithms that is used to estimate a probability distribution in the presence of missing attributes.
- Using it, requires an assumption on the underlying probability distribution.
- The algorithm can be very sensitive to this assumption and to the starting point (that is, the initial guess of parameters.
- In general, known to converge to a local maximum of the maximum likelihood function.
- Some more details.

Inference

- An attempt to move into higher level tasks in NLP typically requires the use of several different classifiers.
- In most cases, though, these classifiers are not independent.
- Instead - their outcomes need to satisfy some domain dependent or task dependent constrains.
- We will use Shallow parsing as an example.
- (POS – too simple).

Syntactic Analysis: History

- Parsing unrestricted language at reasonable accuracies became possible only in the last 5 years using statistical learning methods
- Shallow Parsing: Advocated as a more plausible partial solution that could be more robust.
- Became feasible using statistical learning methods
- In both cases: lexical information is used

Identifying Phrase Structure

[$_{NP}$ He] [$_{VP}$ reckons] [$_{NP}$ the current account deficit] [$_{VP}$ will narrow] [$_{PP}$ to] [$_{NP}$ only # 1.8 billion] [$_{PP}$ in] [$_{NP}$ September]

- Classifiers
  1. Recognizing "The beginning of NP"
  2. Recognizing "The end of NP"
  3. Also for other kinds of phrases…
- Some Constraints
  1. Phrases do not overlap
  2. Order of phrases
  3. Length of phrases
- Use classifiers to infer a coherent set of phrases

Phrase Identification Problem

- General Paradigm: Inference with Classifiers

❖Applications:

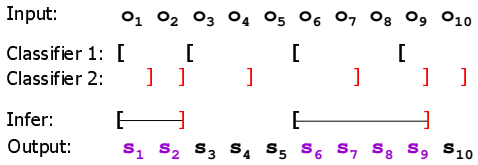
- Shallow parsing (chunking)
- Identifying document structure
- Computational Biology: Detecting Splice Sites
- Multi-Modality: Identifying Actions of Interest

Outline

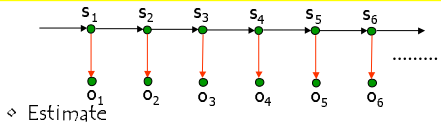
- Phrase Identification Problem
- Markovian Formalism
  - Hidden Markov Models (HMM)
  - Projection based Markov Model (PMM)
- Constraint Satisfaction Based Formalism
- Robustness of Shallow Parsing

Phrase Identification Problem

- Use classifiers' outcomes to identify phrases
- Phrase structure needs to satisfy some constraints

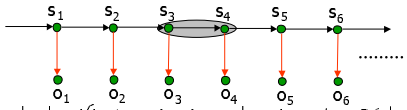


Hidden Markov Model



- Estimate
  - Initial state probability $P_1(s)$
  - Transition probability $P(s|s')$
  - Observation probability $P(o|s)$
- Goal Not exactly what we want
  - $\text{argmax}_{s,S} P(S|O)$
  - Can use dynamic programming (Viterbi)

HMM with Classifiers



- Each classifier's output can be viewed as $P(s|o)$

$$P_t(o|s) = \frac{P_t(s|o)P_t(o)}{P_t(s)}$$

Constant at time t

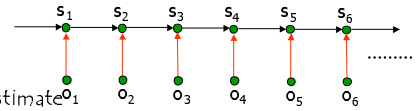
$$P_t(s) = \sum_{s'} P_t(s|s')P_{t-1}(s')$$

➔ Constraints are incorporated via the transition probability

Conditional Models

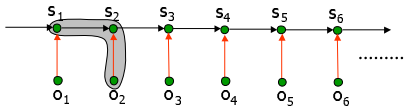
- Model States directly
- Directly incorporate the previous states in term of features
- Train many classifiers each of which is projected on a previous state
  - More classifiers, but *simpler*

Projection-based Markov Model



- Estimate
  - Initial state probability $P_1(s|o)$
  - Transition probability $P(s|s',o)$
- Goal
  - $\text{argmax}_{s,S} P(S|O)$
  - $\text{argmax}_{s,S} \Pi_{t=2..n} [P(s_t|s_{t-1}, o_t)] P_1(s_1|o_1)$
  - Can use dynamic programming (Viterbi)

PMM with Projected Classifiers



- $P_1(s|o)$ - the classifier projected on the first symbol of sequences
- $P(s|s',o) = P_{s'}(s|o)$ – the classifiers projected on each previous state

➔ Constraints are incorporated via the transition probability

The Cost Function

- Markovian Method
  - Maximize the probability over the sequence
- The True Cost Function
  - Maximize the number of correct phrases
  - Minimize the number of wrong phrases

Constraint Satisfaction (CSCL)

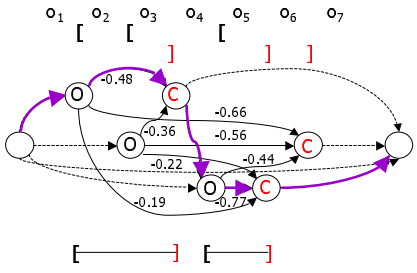
- We extend the Boolean Constraint Satisfaction formalism to handle variables that are outcomes of classifiers
  - V - set of variables; Clauses: model constraints
  - f - A CNF, the CSP problem.
  - Satisfying assignment: τ: $V \rightarrow \{0,1\}$
  - Cost: c: $V \rightarrow \Re$
  - Find the solution τ that minimizes the cost

$$c(\tau) = \sum_{i=1..n} \tau(v_i)c(v_i)$$

Modeling Constraints

- Let V be the set of all possible phrases
- $f = \bigwedge_{v_i \text{ overlaps } v_j} (\neg v_i \vee \neg v_j)$
- c = 1-P(O)P(C) ↘ -P(O)P(C)
  - P(O) and P(C) are supplied by classifiers
- CSP in general is hard
- Structure of the constraints yields a problem that can be solved by shortest path algorithm

Constraint Solution



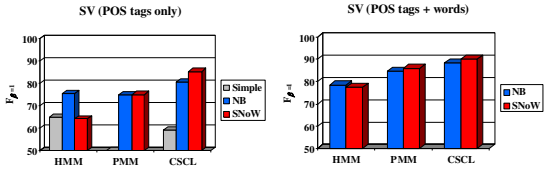
Experiment (Shallow Parsing)

- Methodology
  - Standard (WSJ) Data Set
    - baseNP: 9k (2k) sentences 54k (12k) patterns
      - Short and fairly easy to recognize
    - SV: 16k (2k) sentences 25k (3k) patterns
      - Longer and harder
  - Recall: (Correct proposed / Correct patterns)
  - Precision: (Correct proposed/ All proposed)
  - $F_{\beta=1} = RP/(R + P)$

Experiment (Shallow Parsing)

- Classifiers
  - Simple: Estimate P(o|s) directly from data
  - Naive Bayes
  - SNoW: A Winnow-based learning architecture
    Both SNoW and NB use same wider context features

Experimental Results



- Identifying SV is harder
  - differences more evident
- CSCL performs better
  - Handles better Longer patterns
  - Better cost function

Standard (WSJ) Data Set

- baseNP: 54k (12k) patterns
- SV: 25k (3k) patterns

Final Thoughts

- Learning methods are becoming prime in the area of natural language processing.

Our goal:

- Survey the principles of machine learning in the context of natural learning.
- Study the main learning paradigms and some of the important algorithms, to support a knowledgeable selection of methods, critical thinking and ability to develop new ideas.
- Lecture notes will be available (see below)