Lecture notes by Edward Loper

Course: CIS 630 (Machine Learning Seminar)
Professor: Fernando Pererez
Institution: University of Pennsylvania

1 Logistics

Office hours: wed before class

1.1 Project

write a summary/somehting that will stand the test of time; or write a good implementation of something; or design a testing framework for a domain; etc.

2 Models for Machine Learning

2.1 Learning Tasks

- Classification (documents, configs)
- Segmentation/tagging/extraction
- Parsing
- Inducing representations (unsupervised)

First few classes – document classification.

2.2 Questions

- Generative or discriminitive?
- Handling small sample sizes – sparse data problem
- sequences: local or global methods?
- Does unsupervised learning help?

2.3 Generative Models

Destimate p(x,y)

Easy to train; robust; probablistic → gives you a way to think about it.

2.4 Discriminitive Models

minimize $\sum[[f(x_1) \neq y]]$ i.e., try to build a function f predicting y given x, and minimize the numer of training errors.

Estimate p(y|x)

- Focus modeling resources on instance-to-label mapping.
- Avoid restrictive assumptions (probablistic). No need for explicit model of the domains. In particular, no statistical independance assumptions.
- Optimize what you care about
- Higher accuracy

2.5 Global Models

(typically EEs)

- Train to minimize labeling loss $\Theta = \text{argmin}(\theta) \sum \text{i Loss}(\text{x[i], y[i]} \mid \theta)$
- Computing the best labeling: argmin(y) Loss(x,y|Θ)
- Efficient minimization requires:
  - A "common currency" for local labeling decisions – how to decide about tradeoffs? Use probability, so we can compare different things.
  - Dynamic programming algorithm to combine local decisions (viterbi)

- principled
- can compose models
- efficient optimal decoding (usually)

2.6 Local Models

(typically machine learning)

- Train to minimize per-symbol loss in context $\Theta = \text{argmin}(\theta) \sum \text{i} \sum \text{j Loss}(\text{y[i][j]} \mid \text{x[i], y[i][k]}; \theta, \text{k} \neq \text{j})$

- Wider range of models
- more efficient training
- heuristic decoding is like pruning

Wednesday, September 12, 2001

3 Generative vs. Discriminitive

Generative: generates instance-label pairs

- process structure
- process parameters: constrain/define the nondeterminism

How do you deduce the structure? How do you estimate the parameters (from training data)?

3.1 model structure

- decomposes generation of instances to elementary steps. we don't want to generate entire documents, they all have very low probabilities. some model mapping btwn documents and smaller steps.
- define dependencies between steps
- parameterize dependencies

Generating multiple features with an HMM: the problem is that the features are not conditionally independant

3.2 Independance/intractibility

- trees are good: each node has a single immediate ancestor, so join probability can be computed in linear time.
- but that forces features to be conditionally independant, given the class.
- unrealistic that they're independant. e.g., "san" and "francisco"

3.3 Discriminitve models

- p(y|x;θ)
- binary classification: define descriminant: y = sign h(x;θ)
- train θ to maximize p(training data), or minimize p(error)

3.4 Classification Tasks

- Document classification: interested in ranking
  - ranking function is important to final outcome..
  - Tagging
  - Syntactic decisions (e.g., attatchment)

3.5 Document Models

```
t = a term.
C_t(d) = term frequency of t in d.
|d| = total words in the document
d = document
D = document collection
```

Binary Vector

Define a binary feature vector. Each term feature is either on or off for a document. (present or absent)

$f_t(d) = t \in d$

Frequency Vector

Define x_t using TF*IDF...

$x_t(d) = F_t(C_t(d)/|d|)$

F_t is a TF*IDF weighting functions:

$F_t(f) = \log(f+1)\log(1+(|D|/|d \in D : t \in d|))$

Use logs to squash F_t, because of burstiness.. first occurance is meaningful, subsequent occurances less so.

Very sparse representation. Use equiv. representations?

Langauge Model

N-gram model

$p(d|c) = p(|d||c)\Pi(i=1, |d|, p(di \mid d1, d2, \ldots, di-1, c))$
$p(di \mid d1, d2, \ldots, di-1, c) \approx p(di \mid di-n, \ldots, di-1, c)$

3.6 Term Weighting and Feature Selection

We want the most informative features. Feature selection:

- remove low, unreliable counts
- mutual information
- information gain
- etc

TF*IDF tries to solve the same problem: adjust term weight by how document-specific it is.

3.7 Kinds of Classifiers

Generative

- Binary naive Bayes
- Multinomial naive Bayes (unigram)
- General class-conditional language model (N-gram)

Discriminitive

- Binary features: exponential, boosting, winnow + embedding into real vector space
- Real vectors: rocchio, linear discriminant, SVM, ANNs

Real vector techniques are more general than binary features...

3.8 Learning Linear Classifiers

Rocchio

Average vector representations for positive and negative classes.

$a = \alpha(\sum(x \in c)x_t)/|c|$
$b = \beta(\sum(x \notin c)x_t)/(|D|-|c|)$
$w_t = \max(0, a- b)$

On average, the negative examples overwhelm the positive in b.. So resistant to using a subclass of the positive numbers..

Widrow-Hoff

Iterative approach. Estimate gradient, and use it to update our weights. η is our learning weight. Move the model in the direction of making the actual label agree with the predicted label.

$w_i+1 = w_i - 2\eta(w_i*x_i-y_i)x_i$

y = actual label w = predicted label

(Balanced) winnow

Faster approach to iterative estimate of classifier weights. Competing positive error and negative error.

Favors sparse representations: terms go to zero quickly, because it's multiplicitive rather than additive.

4 Naive Bayes (Anne)

4.1 Bayes Nets

Encodes dependence & independance relationships Sparse representation of the entire PDF, given that there aren't too many dependencies.

4.2 Using Bayes Nets for Classification

Simply compute P(C|X). But if X is highly dimensional, this is very difficult to find. E.g., if X is a binary vector of whether each word occured (in document classification).

Use Bayes rule (including independance) to calculate backwards..

```
P(c|x) = P(c)P(x|c)/P(x)
```

Naive Bayes: assume conditional independance between multiple occurances of a word (between multiple features).

Room for improvement:

- Can we use dependance information to improve effectiveness of Naive Bayes classifiers?
- Modifications of feature sets?
- Better text representations?

4.3 McCallum & Nigam

Naive Bayes: just use presence/absence Multinomial: use multiple occurances..

Compare multivariate/multinomial.. Results:

- multi-variate Bernoulli handles large vocab poorly
- multinomial event model more appropriate for classification with large, overlapping vocabs

4.4 Sahami (sp?)

We want something between naive bayes and accounting for all dependencies. Find mutual information between class and features. Add features one at a time. Connect each new node to k of the nodes that you've already added. (Pick the k with the highest mutual information). Try k values of 0..3, and use a threshold: have <k parents if they won't help (if mutual info is low)

Higher k does help; but too high can hurt (false dependencies: many features really are unrelated). Classification accuracy variance can uncover dependance properties (as you vary k).

4.5 (?)

Flat Classification

- one routine examines documents, classifies them
- large number of features (1000s)
- computationally expensive
- overfitting (& sparse data problems)

Hierarchical Classification

Multi-tier classifier

1. select features (given the data)
2. supervised learning creates the classifier for each tier

Reduces both total number of features, and the number of features used locally.

Hierarchical classification helps. :)

5 Documents vs. Vectors

- Many documents have the same binary/freq vector
- Document multiplicity must be handled correctly
- Multiplicity is not recoverable
- document probability

$$p(d|c) = p(|d||\ c) = \prod p(di|\ c)$$

Do you mean probability of a document given a class, or the probability of a count vector given a class? If so, we must add multinomial coefficients. (Add factorials)

$$p(r|c) = P(L|c)L!\ \prod p(t|c)^{r_i}|r_i!$$

- Use bayes rule for classification. When we cancel things, the multinomial coefficients cancel.

6 Maximum Entropy Modeling

(Eugen Buehler)

6.1 Entropy and Perplexity

- Entropy:

```
H(p) = -∑ p(x) lg p(x)
```

- Perplexity:

```
2^H(p)
```

6.2 ME

Given what we know, find the probability distribution that maximize entropy.

What we *do* know will reduce the entropy of the system Choose the p dist that matches what we know, without assuming anything we don't (which is done by maximizing entropy)

Assume a set of features:

```
fi: ϵ -> {0,1}
```

Constrain expectation of the feature function under the probability model p:

```
∑ p(x)fi(x) = ∑ pbar(x)fi(x) = (1/|T|) ∑ fi(x)
```

A unique solution exists, with exponential form:

```
p(x) = 1/Z exp(∑λ i fi(x))
```

Z = normalization, λ i = parameters

This is just the MLE for out training data (though it's not easy to prove – information geometry)

Conditional ME:

```
p(y|x) = 1/Z(x) exp(∑λ i fi(x,y))
```

Another take on the MLE thing.. these are equivalant:

1. assuming we're exponential, how close can we get to the constraints?
2. assumin we have a distribution with these contsraints, how close can we get to the uniform distribution?

6.3 Solving for lambdas

- generalized iterative scaling (GIS)
- improved iterative scaling (IIS)

no closed-form solution, so use hill-climbing techniques. The hill-climbing technique you use can add constraints, like binary features, features must sum to a constant, etc.

6.4 Building ME models

To build an ME model:

- phrase the problem as a prob dist
- design a set of relevant features (!)

6.5 Text Classification: Nigam et al.

- objective: Find p(c|d) (calss given document)
- use one feature type: a weighted word count

```
f_w,c'(d,c) = N(d,w)/N(d) if c=c', otherwise 0
```

- feature selection?
  - one approach: include all features, let the model work it out
  - but no feature selection is bad: it can result in over-rating very rare features. If xyz appears in 1 document in a corpus of 100, ME will say that P(xyz)=1%. (this is basically a case of over-fitting)
  - so if there's no feature selection, you need smoothing. Assume gaussian distribution, centered on zero (no effect). Maximize posterior probability, not P(training data). we can use held-out data to estimate variance of gaussian
  - results:
  - compared to multinomial naive bayes
  - better on 2/3 tests. (not particularly impressive)
  - no feature selection; could include more features for ME, that are not available for naive bayes

6.6 PP Attatchment: Ratnaparkhi, et al.

"I saw [a man in the park] [with a telescope]"

Reduce to {V=saw, N1=man, P=with, N2=telescope}, try to predict whether N2 should be attatched with N1 or V

Result of 0 if you attatch to N1, 1 if you attatch to V

Features have a value of 1 for noun attatchment, 0 for verb attatchment. But that's ok. P(noun) = 1-P(verb).

ME doesn't assume independance of features (but GIS, IIS converge faster the more independant they are)

Feature space = compositions of binary questions:

- about identity of tuple members
- about class of tuple members

Feature selection

- select best feature based on an estimate increase in log-likelihood
- train new model
- add a special set of candidate features, related to the new feature
- repeat

Binary outcome conditional ME is equivalant to logistic regression. So they should have compared to stepwise logistic regression (this tests feature selection). stepwise logistic regression is basically logistic regression with feature selection

6.7 Berger et al. and translation

- build ME as supplements to a french-english MT system
- try to find p(y|c), between languages, for a given word... e.g., should we translate "in" as "dans" or "en" etc..
- feature sets: test for a given word (this is basically the a priori probabilities.. e.g., in→ en 25% of the time). Also, check for immediate following word, immediately preceeding word, word x is in the 3 preceeding words, word x is in the 3 following words.

Feature Selection

- A set of candidate features F
- An empirical distribution p
- A set of active features S (initially empty)
- The current model, p[s] (initially uniform, since S is empty)
- For all candidate features, find the parameters using IIS, then compute gain in likelihood of training data.
- select that feature
- when new feature does not improve performance on held-out data, we're done

Problem: IIS is slow, so this training method is slooowww.. :)

Estimating likelihood gain

Instead of calculating exact likelihood gain, estimate it:

- during IIS, keep all parameters equal to original model, solve only for the new parameters (i.e., assume independance)
- this makes it computationally feasable

7 Maximum Entropy Review

7.1 Conditional Maxent Model

- multi-class
- can use diff features for different classes

8 Duality

- Maximize conditional log likelihood, given model form
- Maximize conditional entropy, subject to the constraints

9 Relationship to (binary) logistic discrimination

If we reduce this to the binary case, then we have a logistic regression problem. So maxent is a generalization of logistic discrimination

10 Relationship to Linear Discrimination

- Decision rule

```
sign(log(p(+1|x)/p(+1|x))) =
    sign ∑[k] λ[k]g[k](x)
```

- Bias term: parameter for "always on" feature – allows the discrimination to not go through the origin.
- Question: relationship to other trainers for linear discriminant functions.

11 Solution Techniques

11.1 Generalized Iterative Scaling

- parameters updates
- additive updates
- initial values? use zero. The more dependant the features, the longer it takes to converge. If we start at zero, we will converge eventually; if we start somewhere random, we might go in circles if features are linearly dependant.
- requires that features add up to a constant independant of instance or label – use a "slack feature"

11.2 Improved Iterative Scaling

- multiplicitive updates
- for binary features reduces to solving a polynomial with positive coefficients.
- Reduces to GIS if feature sum is constant

11.3 Another approach:

- use standard convex optimization techniques
  - conjugate gradient, etc.
  - converges faster?

12 Gaussian Prior

- If we have a gaussian prior, we can tweak IIS to update according to variances.. (?)

13 Representation

- fixed-size vs variable-size instances.
- multivalued features

Monday, September 24, 2001

14 AdaBoost and Variants

Andrew

- Boosting: take several "weak" predictors and combine them to make one "strong" predictor.
- "Weak" means only slightly better than random. We can use stronger "weak" predictors, but we don't need to..

14.1 Weak learner

Consider a weak learner h:

```
h : x → {0,1}
```

14.2 Motivation

When we train a classifier, some training samples are "harder" than others. One approach: take hard ones, duplicate them, and train (places emphasis of learner on the hard observations).

Boosting is like this:

```
1. Train classifier on h
2. Cake copies of hard observations
3. Go to 1
```

At the end, combine all of these somehow.

14.3 Initial Observation weights

Initially, use uniform distribution:

```
i = iteration
m = number of observations

Distribution D[1](i) = 1/m
```

Boosting loop

```
For T iterations:
   Generate clasifier h[i]
   Choose reweight term a[t]
   Calculate
   Update
```

14.4 Error bound on test data

- VC-dimension is a measure of the complexity of a hypothesis space.
- We can put an upper bound on the probability of misclassification.

Boosting seems to be resistant to overfitting. :)

14.5 Multiclass/Multi-Label

multiclass: ternary decision, etc. multilabel: each observation can have a variable number of classes. E.g., a document might have multiple document categories.

For multiclass, we can have one binary classifier for each class, and put them back together afterwards.

Two views:

- we are concentrating on the decision boundry. This is a good thing, cuz we get better classification.
- we are concentrating on outliers, and mangling our model to accomodate them.

For labeling: if you get too much label noise, then the algorithms start overfitting horribly.

15 Review of Boosting

Training instances:

```
x[i] is training instance
y[i] is label: {-1,1}
(x[1],y[1]), ..., (x[m],y[m])
```

Start with uniform distribution:

```
D1[i] = 1/m
```

For t = 1, ..., T:

- train weak learner using Dt
- get weak hypothesis h[t]: maps instances→ labels
- choose α[t] (real)
- update the distribution:

```
D[t+1][i] = D[t][t] e^(-α[t]y[i]Ht(x[i]) / Zt)
```

Where y[i] ∈ {-1,1} and Ht(x[i]) ∈ {-1,1}

```
H(x) = sign(∑[t] α[t]h[t](x))
```

```
α[t] = 0.5 ln( (1-ϵ[t])/ϵ[t] )
```

We can bound our error by:

```
∏[t] Z[t]
```

```
ϵ < P[margin(x,y) ≤ θ] + Θ(sqrt(d/(mθ^2)))
```

(Θ is order)

16 SVM

[josh]

Look for a linear separating hyperplanes. There are infinite such planes. Which one should we use? We can write each hyperplane as a linear combination of vectors (plus a const).

```
f(x, α) = (w[α] · x) + b
```

If we just pay attention to the minimum margin, we don't really care about the margin of the points we classify well anyway.

support vector = one of the vectors that we're using to define our hyperplane.. the distance from all of the support vectors to the hyperplan is 1..

We can expand svm into additional dimensions, using a mapping function. if we pick our mapping function carefully, then we can avoid a lot of computation. For example, project (x1,x2) into two dimensions:

```
Φ(<x1,x2>) = <x1^2, sqrt(2)x1x2, x2^2>
```

Then

```
Φ(u) · Φ(v) = (u· v)^2
```

"Kernel" combines projecting & combining. So it behaves like inner product, but it's acting via a higher dimension

Monday, October 1, 2001

17 SVM (continued)

Features are only refered to indirectly, via the support vectors. This makes the machine less dependant on the number of features.

SVMs tend to yield high accuracy.

VC Dimension → the fewer the support vectors, the smaller the VC dimension. Prediction: accuracy will be higher if VC dimension is smaller.

In SVM, the kernel allows the mechanism to access features that may not be available elsewhere..

Monday, October 1, 2001

18 Solving Large-Margin Problems

18.1 Linear Classification

- Linear discriminant function

```
h(x) = w · x + b = ∑ w[k]x[k] + b
```

18.2 Margin

- Instance margin: γ[i] = y[i](w*x[i] + b) Either positive or negative
- Normalized (geometric) margin (positive/negative)
- Training set margin γ = min(geometric margins)
- Assume functional margin is fixed to one.

18.3 Why maximize the margin?

- $\exists$ c s.t. for any data distribution D with support in a ball of radius R and any training sample S of size N drawn from D..

18.4 Convex Optimization

- Constrained optimization problem
-

18.5 URLs

- www.kernel-machines.org
- www.support-vectors.net

Wednesday, October 10, 2001

19 Learning Theory...

19.1 Statistical Learning Theory

Form: If problem is in a given complexity class, then with high probilibility, we can bound our error by some function of the number of training examples.

But that doesn't tell us about how hard it is to do computationally: finding the class with very low error may be intractable.

Statistical Learning Theory tells you what's possible, not what's computationally feasable.

19.2 Definition of PAC

PAC = Probability Approximately Correct

Incorperates what's possible with what's computationally feasable.

`C: class of concepts`
`concept` $\equiv$ `X` $\rightarrow$ {0,1}

World chooses a concept for us, and a distribution over the data:

- c $\in$ C
- D $\subset$ X $\times$ {0,1}

We could also define it such that there is a noise distribution that corrupts labels.

`h` $\in$ `C is a hypothesis`

Then

`P(error(h)` $\leq \epsilon$`)` $\geq$ `1-`δ

where we pick ϵ and δ

$\exists$ algorithm to find h, that is polynomial in $(1/\epsilon)(1/\delta)$

Most results from PAC are negative: we cannot do it.

book.. Kearnes & Vazarani: An intro to Computational Language Theory

20 Using Unlabeled Data

- Labeling is expensive: manual
- Unlabelled instances are easy to find: web pages etc
- Unlabeled data is useful
  - Joint pdfs of unlabeled data
  - merge 2 views of 1 example

20.1 Basic approaches

- co-training
  - exploit 2 views
  - combination of EM and NB classifier
  - exploint joint PDF of unabeled data (joint btwn features)

20.2 Co-Training

- task: find faculty member pages
- two training sets for labeled pages:
  - text pointing to the document
  - text inside the document
  - labeled examples are expensive
  - unlabeled pages are easy to get
  - reduce necessary labeled data by using feedback btwn 2 views

Bootstrapping:

- train weak predictors A and B from training data
- use weak predictor A to find new training data for B; predictors for B to find new training data for A
- repeat

Compatibility assumption:

- All labels on examples with nonzero probability under distribution D are consistant with some target function $f_i \in C_i$, i=1,2,...
- For any example x=(x_1, x_2) observed with label L: $f_1(x_1) = f_2(x_2) = L = f(x)$
- D assigns probability zero if $f_1(x_1)$!= $f_2(x_2)$
- In this case, (f_1,f_2) is compatible with D
- (C_1,C_2) is of high complexity, while compatible target concepts might be much smaller.

Compatible concept = a concept with no cross-edges.

Bitartite graph

We have 2 types of lines connecting the sides of the graph: labelled instances and unlabelled instances. Propagate from labled instances to unlabeled ones. 2 issues:

- what if you can propagate from + to -? (contradiction)
- what if you can't propagate to an edge? (no label)

Application: WSD

within a document, it is very likely that all instances of a given word have the same sense. (well, kinda. verb/noun meanings of the same word? etc.)

PAC Analysis: Rote Learning

- Assume $|X_1|=|X_2|=N$, $C_1=C_2=2^N$, all partitions consistant with D are possible.
- Output "I don't know" when you can't derive a label from training/consistancy.
- O((log N)/a) unlabeled examples are sufficient

A more robust approach: minimize an objective function that includes the errors of each learner on its own training data, plus the disagreement between the learners on unlabeled training data.

20.3 Text Classification using EM & unlabeled data

- Unlabeled data provides information about the joint PDF over words
  - "homework" tends to belong to the positive class L
  - Use this fact to estimate the classification of unlabeled documents, and get a new positive class L'
  - L' gives us "lecture" (cascading effect)

Technique:

1. Train classifier with labeled documents
2. Use classifier to assign probablisticly-weighted class labels to each unlabeled document by finding expectation of the missing labels. (E)
3. Train a new classifier using the documents (M)
4. Repeat 2/3 (E/M) until convergence

20.4 Generative Model

2 assmptions:

- document is produced by mixture model
- one-to-one correspondance between mixture components and classes.

EM Loop: Expectation: use current classifier to estimate component membership of each unlabeled document Maximation: re-estimate the clastifier, given the component membership of each document. Use a maximum a posteriori probability estimation to find argmax$[\![\theta]\!]$ P(D|θ)P(θ)

Helps more if we don't have enough labeled docs

20.5 Augmented EM

- Mixture components are not in correspondance with calss labels.
  - Give different weight to the unlabeled data
  - Multiple mixture compoenents per class

Monday, October 15, 2001

21 Expecation Maximation

web: Convexity, Maximum Likelihood and All That (Adam Berger)

21.1 Motivation

- Hidden (latent) variable models

```
z = unobserved variables
    Creates a larger class of models to fit our data
p(y,x,z | Λ)
```

Work with marginal distribution:

```
p(y,x | Λ) = ∑ p(y,x,z|Λ)
```

topics as a hidden variable:

```
class -generates→ topic -generates→ words
```

Examples:

- Mixture models
- Class-based models
- HMMs

generalize: E/M

21.2 Maximizing Likelihood

- Data log-likelihood

- D = $\{(x_1,y_1), \ldots, (x_n,y_n)\}$
- $L(D|\Lambda) = \sum_i \log p(x_i,y_i|\Lambda)$

Find parameters that maximize (log-)likelihood

Use a regularizing term to keep Λ closer to a prior distribution?

21.3 Convenient Lower Bounds

- Convex function
- Jensen's inequality

```
f(∑ p(x)x) ≤ ∑ p(x)f(x)
i.e., f(E(x)) ≤ E(f(x))
```

(where f is convex, p is a pdf)

- Find a lower bound function that touches the function we want (at

p). Maximize function to p_{max}, and then repeat with p=p_{max}

- Better than gradient asecent, since we don't need to worry about

step size: sinced it's tangent, we're going up. since its a lower bound, we're stil below. since it's pmax, the corresponding point is higher than p. (no danger of having too large of a step size, like you have with gradient ascent)

21.4 Auxilliary Function

- Find a convenient non-negative function that lower-bounds likelihood increase.
- L(D|Λ') - L(D|Λ) ≥ Q(Λ',Λ) ≥ 0
- maximize lower bound
  - $\Lambda_{i+1} = \text{argmax}_{\Lambda'} Q(\Lambda',\Lambda)$

```
  1        p(z|y)
------- = -------
 p(y)      p(y,z)
```

```
p(z|y) = p(y,z)/p(y)
```

so:

```
p(z|y,x,Λ) = p(y,z,x|Λ)/p(y,x|Λ)
```

Start with $\log(\sum_i(\ldots\Lambda_i\ldots))$ Convert to $\sum(\log(\ldots\Lambda_i\ldots))$ now we can maximize for each Λ_i (since the $\sum$ compoenents are independant - derivitive of a sum is the sum of the derivitives). So maximize each $\log(\ldots\Lambda_i\ldots)$ independantly.

21.5 Algorithm

$\Lambda_0 \leftarrow$ carefully chosen starting point repeat to log-likelihood conergence:

- E step: compute $Q(\Lambda'|\Lambda_i)$
- M step: $\Lambda_{i+1} \leftarrow \text{argmax}_{\Lambda'} Q(\Lambda'|\Lambda_i)$

21.6 Comments

- Likelihood keeps increasing but:
  - can get stuck in local maximum (or saddle point) – doesn't usually occur in practice.
  - can oscillate between different local maxima with the same log-likelihood
  - If maximizing the aux function is too hard: find any Λ that increases likelihood: generalized EM (GEM)
- Sum over hidden variable values can be exponential if we're not careful.

21.7 Mixture Model

- base distributions: $p_i(y)$
- mixture coeff: λ s
  - p(c,y|Lambda) = $\lambda_c p_c(y)$
  - auxilliary function
  - $\sum_y p(y) \sum_c p(c|y,\Lambda)\log$ (p(y,c|Λ'),p(y,c|Λ))

to do soon:

- prepare cis630 lecture
- write problems for cis530 exam: tagging + mylecs
- fix line tokenizer, repl with '{\}n{\}n' tokenizer (re)
- fix tutorial, pset
- pick f-score cutoff

conventions:

- repr = standard repr; str = verbose repr (can be multiline) pp = pretty print (usu. multiline – takes right/left args)
- exception use?
- type checking
- equality/ordering comparisons
- immutable ↔ hashable

22 Projects

22.1 Java Implementation

Do a Java implementaiton of some of these techniques that is:

- extensible
- easily modifiable
- etc.
- since we're using a higher level language, we can make things simpler.
- more emphasis on speed than my project
- nearest neighbors, svm, winnow. NB

22.2 Text Classificaiton & Nigam

Jean

- Given a set of classified documents, build a statistical model p(c|d), the probability given a document that it belongs to a class c
- Nigam et all use one feature type: frequency of a word. These add up to 1, which is very convenient.
  - Results in as many as 57k features (no feature selection)
  - No feature selection tends to create overfitting
  - Address this in hindsight by saying that paramters should have a gaussian distribution.
  - Maximize the posterior P rather than the P of training data
  - Wider set of features: phrase counts
  - $N(p,d)/N_p(d)$
  - features sum to one.
  - define phrases in complementary fasion? "computer science"

  doesn't count as "computer" or "science" alone.

22.3 A Comparison of Text Classification Algorithms

Survey. Not much implementation.

Corpus: hungarian newswire articles

- 9k news articles, 9 channels
- keywords: 13.8k keywords, 33k occurances
- task: assign a channel or keyword to new articles

22.4 Template Relations Task

- Task for MUC-7
- TRs express domain-independant relationships between entities
- TR uses LOCATION_OF, EMPLOYEE_OF, PRODUCT_OF.
  - *Nance*, who is a paid consultant of *ABC News* ...
  - Answer key contains entities for all organizations, persons, artifacts that enter into these relations
  - Training data: 500kb, 1k entities, 1k relations
  - Most relations are local (e.g., appositive)
  - Best results: 74% precision & recall

  - Project
  - Incorperate syntactic features (shallow parsing, or XTAG supertags)
  - Use discriminitive classifier

22.5 Using Machine Learning in Anaphora Resolution

Na-Re & Cassandra

- NLP system must provide "interpretation" for NP.
- Pronouns
- Use classifier: they are or are not correferential
  - If you get 0 for both or 1 for both, fail.
  - But we really want ranking: competition between antecedants.
  - Try to re-cast maxent as a ranking method.
  - Rank using likelihoods
  - Experimental results: less than spectacular
  - Use Collins descriminitive reranking ("desciminitive reraking for NLP" (2000))

22.6 Modelling author communities

Papers with text & ciations. We know what year each paper is from. General problem: see how different intellectual communitites evolve over time.

There's a bunch of hyperlink analysis etc to cluster points that you can call communities..

People in the same community use similar language..

22.7 Benchmark Comparison of the Aspect Model

22.8 and Mixtures of Naive Bayes

Andrew Schein

- with em, use totally unlabeled data, see what the model gives..
- goal: model the probability distribution of a person reading a document:

```
P(p,d)
```

Find the probability that the person has read the document.

- Use these probabilities to recommend documents to read
- 2 paraatermizations. basically like using 2 distributions:
  - mixture of naive bayes
  - aspect model
  - aspect model ("latent variable model")
  - observation = (person, document)
  - person associates with multiple classes
  - assume that each observation is generated by a single class, but one person has multiple classes (mixture model)
  - mixture of naive bayes
  - person belongs to a single class
  - c.f. autoclass
  - dataset:
  - movielens: what people watched what movies
  - ˜1k people, each recommending ˜20 people
  - ˜2k movies

Thursday, October 25, 2001

23 Sequence Modeling

- Assign a labeling to a sequence
  - story segmentation
  - POS tagging
  - shallow parsing
  - named entities
  - global models
  - train to minimize overall labeling loss
  - local models
  - train to minimize per-symbol loss in context
  - for each symbol, find best label given a hypothesized context.
  - generative vs discriminitve

24 Information Extraction with HMMs and Shrinkage

- IE: automatic extraction of subsequences of text (e.g., extract location or time of a meeting)
- apply shrinkage to HMMs
- Task:
  - given a model & parameters, figure out sequence of states
  - use viterbi
  - use HMMs with topology set by hand. there are target states

  (generate text we want to extract) and background states. (only one target state, the rest are background states)
- shrinkage combines estimates with a weighted average and learns the estimaties with EM.
- shrinkage hierarchy configurations:
  - none
  - uniform: all distributions are shrunk towards uniform
  - global: all target states & non-target states are shrunk toward a common parent
  - hierarchical: some states are shrunk towards different states
  - local estimates calculated from ratios of counts
  - find improved estimate for $P(w|s_j)$..
  - estimating weights (use EM)
  - initialize uniformly
  - find degree to which each node predicts words
  - derive improved weights

25 Named Entity Restriction with HMMs

- Task: identify names, locations, etc.
- Labels: entities, times, numerics
- start with hand-built network, model both names & locations
- find the most likely sequence of classes..viterbi
- 2 level model.. high level HMM model, with states that have bigram models inside them
- words are ordered pairs <w,f> f = features: twoDigitNum, fourDigitNum, otherNum, allCaps, capPeriod, firstWord, etc. These allow us to deal with unseen data
- Results..

Monday, October 29, 2001

26 Maximum Entropy Markov Models

27 and Conditional Random Fields

- Task: Extract question/answer pairs from a FAQ
- Task: Mining the web for research papers
- Information extraction with HMMs.
  - P(s|s')
  - P(o|s)
  - Problems with HMMs:
  - Want richer feature representation
  - But Can't have multiple overlapping features
  - Naive bayes doesn't work well
    - would prefer conditional, not generative, model

Transform:

```
Transitional HMM → Maximum Entropy Markov Model
    P(s|s')      →         P(s|o,s')
    P(o|s)
```

Think of it as haing:

$P_{s'}(s|o) = P(s|o, s')$

Each state contains a "next-state classifier" black box, that, given the next observation, will produce a PDF over next states.

This is a conditional PDF. We can't find P(o|s).. We *must* start with an output, and only then can we predict probabilities. Conditional model doesn't know the absolute distribution of outputs.

State transition probabilities based on overlapping features

Feature depends on obseration and state: $F_{o,s}(o_t,s_t) = 1$ if b(o) is true and $s = s_t$

Exponential form:

$P(s|o,s') = 1/Z(o,s) \exp(\sum \lambda_{b,s} f_{b,s}(o,s))$

Note: we have a separate PDF for each s'. Thus, the notation:

$P_{s'}(s|o)$

Do maxent training on each of these PDFs.

Models tested:

- ME-stateless: classify each line independantly with maxent
- TokenHMM: standard HMM generating tokens
- FeatureHMM: convert lines to sequence of features, then generate them independantly. I.e., naive bayes HMM with overlapping line features
- MEMM: maximum entorpy markov model

Smoothing? (e.g., for zero probability transitions)

27.1 Variation 2:

- Observations in states instead of transitions
- n^2 contexts (for n states): increased sparseness
- Do P(s|s',o) = P(s|s') * maxent..

27.2 Summary

- New probablistic sequence model based on maxent
  - arbitrary overlapping features
  - conditional model
  - positive results

27.3 Label Bias Problem in Conditional Sequence Models

Example:

```
    ---> 1 ---> 2 --0 --           --> 5
      ---> 3 ---> 4 --
```

```
0→ 1 r
1→ 2 i
2→ 3 b: rib
0→ 3 r
3→ 4 o
4→ 5 b: rob
```

P(path|observations)

```
P(1,2|ro) = P(1|r)P(2|o,1)
          = P(1|r) 1
          = P(1|r) P(2|i,1)
          = P(1,2|ri)
```

```
            P(2,o,1)
P(2|o,1) = -------- = 0/0
             P(o,1)
```

Because 1→ 2 is a forced choice.. So P(2|*,1)=1 for any *, since 2 is a forced choise from state 1.

- Biases towards states with fewer outgoing transitions (esp deterministic states)
- Per-state normalization does not allow the required property:

```
  socre(1,2|ro) << score(1,2|ri)
```

Determinization:

- not always possible
- state-space explosion

Fully-connected models:

- lacks prior structural knowledge

Their solution: conditional random fields

Suppose there is a graphical structure for Y.

```
G = (V,E)
Y = (Y_1, Y_2, ..., Y_|V|)
```

Define:

```
p(Y|X)
X = input observations
```

Probability of a node is dependant on the entire input and the element that points to it.

- With an HMM, we can only encode history of the input with expanded states.
- With CFRs, a feature can depend on the entire input, so it can encode something about the input history much more easily

- Try using conjugate gradient instead

28 Combining Models ot Improve Tagging Performance

Andy

28.1 Boosting Applied to Tagging and PP Attatchement

Abney, Schapire, Singer

Boosting

- Train a series of weak learners $h_t(x_i)$
- At each iteration t, re-weight training examples to emphasize the hard examples.
- After training all T learners, build a finalclassifier: $H(x) = \text{sign}(\sum \alpha_t h_t(x))$
- h_t are given weight according to their performance (α_t)
- n.b. 2 weightings: one over h_t, the other over training examples
- Updating weights of observations: $D_{t+1}(i) = D_t \exp(-y_i h_t(x_i))/Z$

Continuous-Valued learners

- predict probability, not just presence/absence

Weak Learners

- Predicates attribute = value (a=v)
  - PreviousWord = the
  - Boosting sselects those predicates that produce better classification accuracy.

Predicate → Classifier

- Define a predicate ϕ on instance x: $x \rightarrow \{0,1\}$
- p_b is the prediction that $\phi(x) = b$
- $h(x) = p_{\phi(x)}$
- $p_b = 1/2 \ln (w^b_{+1})/(w^b_{-1})$

Multi-label boosting

- Sometimes, we want more than 1 tag as output!
- Use adaboost.MH
- Find p_b for each class independantly.

Features:

- Lexical attributes
- Contextual attributes
- Morphological attributes

PP attatchment

```
I warned [the president   of pecedilis]
I warned [the president] [of pecedilis]
```

28.2 Improing Accuracy in Word Class Tagging through the

28.3 Combination of Machine Learning Systems

van Halteeron, Daelemans, Zavrel

- gang method - average, voting, etc.
- arbiter method - use a learner to learn which arbiter to use

Why does combining help?

- models may have similar accuracy, but they maybe different errors.

Ensemble = the combined method

Arcing methods:

- bagging: sample with replacement to build N classifiers, then combine them.
- boosting

Monday, November 5, 2001

29 Project Schedule

- Proposed topic by Oct 10
- Five-minute proposal Oct 17
- 5-min Project revies Nov 7th (this wed)
- 15 min project presentations Nov 26th and 28th
- final deliverables dec 14th

30 Class Schedule

- Information bottleneck: monday before thanksgiving
- *check this* 19th?

31 A General Finite-State Formalism

Generalize regexps to weighted rational transductions:

- reversible, composable input-output patterns
- weighted alternatives
- target for learning algorithms

Sequence models: HMMs, sequence maxent, etc.

- Structure
- Parameter setting
- Learning structure?

31.1 Weights

Weight semiring: generalize the notion of multiplicity (as in multisets).

Multiplicity: how many different ways can we recognize a string? Might include P, might not. Weighting is not necessarily a probability.

- Sum: compute the weight of an object from the weights of its possible derivations. Associative, commutative.
- Product: compute the weight of a derivation from the weights of its steps. Associative, distributes over sum.
- 0: 0+x=x; 0*x=0
- 1: 1*x=x

31.2 Regular Transductions vs. Regular Expressions

	Regexp	Rational Transduction
meaning	set of strings	functions from pairs of strings to weights
element	{a}	$[\![a:b/w]\!](u,v)$ (a to b cost w)
sequence	$[\![ST]\!]=[\![S]\!][\![T]\!]$	$[\![ST]\!](t,w) = \sum_{rs=t\ uv=w} [\![S]\!](r,u)*[\![T]\!](s,v)$
alternation	$[\![S\|T]\!] = \cup$	$[\![S+T]\!]$
Closure	$[\![S*]\!]$	$[\![S*]\!] = \sum [\![S]\!]^k$
composition	(none)	$[\![S\circ T]\!](u,w) = \sum_{v} [\![S]\!](u,v)*[\![T]\!](v,w)$

31.3 Composition of Weighted Transducers

- Composition rule:

$$\frac{s \xrightarrow{a:b/u} s' \qquad t \xrightarrow{b:c/v} t'}{(s,t) \xrightarrow{a:c/(u*v)} (s',t')}$$

- Lazy algorithm with optional memoization

31.4 Learning

- Compile n-gram stats, hmms, etc. into this form
- Compile decision trees into transducers
- Compile transformation-based taggers into transducers
- Direct automata learning by state merging

Trainable edit distance

Make weighted transducers to model edit errors.. Train an edit distance learner..

Determinization

- it's not always possible to determinize a weighted transducer
- Instead of having sets of states, have sets of state/output pairs.

DAWG = directed acyclic word graph = minimized form of a trie (retrieval tree).

Start with DAWG, and then merge states.

31.5 K-Reversibility

- A k-reversible automaton = deterministic, and reversed version of the automaton is deterministic with lookahead k.
- Means that, if you look back k steps, then you know where you must have come from.

32 Comments for my presentation

- Should feature values be more general?
- Should feature objects have IDs, and FeatureList return IDs?
- Better use of numpy? (behind-the-scenes stuff)
- Abstract the notion of a feature value list (instead of just a list?)
- Extraction whee

Instance -> FeatureList -> FeatureValueList

- what is a "FeatureValueList"? Sequence? Map? We want to be able to iterate over it..
- Should factory separate train/get_classifier (with train applying to a single text)?

32.1 Basic Classes/Interfaces

- Feature: apply() id() [**]
- FeatureList: detect(), +, len()
- FeatureValueList:
  - iterate over (id,val)
  - request val for an id?
  - LabeledType
  - ClassifierI
  - ClassifierFactoryI
  - FeatureSelectorI
  - LabeledFeatureValue

- pdf1: P(LabeledFeatureValue|Label)
  - samples = ?? Maybe LabeledFeatureValueList ??
  - pdf2: P(Label)

- LabeledFeatureValueProbDist
  - samples = LabeledFeatureValueList
  - event1 = LabelEvent
  - event2 = FeatureValueEvent
  - Uses NBProbDist?

- NBProbDist:
  - events
  - P(inst) = $\prod$ P(event)
  - Have a different PDF for each event?
    - Apply smoothing on each PDF..?

- Does smoothing apply to prob dists or freq dists??

- Notion of a random variable?

Random thoughts..

- Terminology:
  - Feature vs FeatureValue
  - FeatureExtractor vs Feature
  - FeatureExtractor vs FeatureValue
  - FeatureExtractorList (??)

Features

Feaures have the following aspects:

- Feature Extractor
- Feature Value
- Feature ID

How do they relate? Well, FeatureExtractors produce FeatureValues. Also, each feature has a unique integer identifier. Integer because that makes it much easier to do things with arrays.

FeatureExtractorList: LabeledText → FeatureValueList

FeatureExtractorList[FeatureID] → FeatureExtractor FeatureValueList[FeatureID] → FeatureValue

FeatureExtractorList.apply(LabeledText) → FeatureValueList

Classes

- FeatureExtractor (=class?)
- FeatureValue (=any?)
- FeatureExtractorListI (sparse)
  - SimpleFeatureExtractorList
  - FeatureValueListI (sparse)
  - SimpleFeatureValueList
  - ArrayFeatureValueList

What does a FeatureValue contain, other than just the value? Is there a reason to use a real class/interface, rather than just a value?

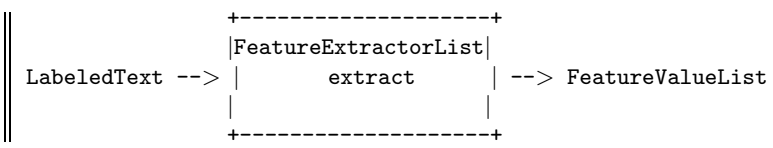
You need to be able to iterate through feature value lists.. Have an items() member or some such? Or assigmnents()? I could even define a new class:

- FeatureAssignment = ⟨ FeatureID, FeatureValue⟩

And have something like: for fa in feature_value_list.assignments():

The alternative is: for (id,val) in feature_values.assignments():

The default is *always* zero.



33 Probablistic Latent Semantic Indexing

Thomas Hofmann

(PLSI)

Domain: documents d with words w Problem: model P(d, w) Simple solution: MLE Want: semantically similar words to be similar Solution: dimensionality reduction

Observation: (w, d) Associate latent class var z with each (w,d)

Generative:

- select d with P(d)
- select z with P(z|d)
- select w with P(w|z)

33.1 Aspect Model

Independance assumptions:

- p(d,w) are independant (bag of words)
- Conditional independance: P(w|z,d) = P(w|z)
- P(w|d) is a convex combnation of factors/aspects P(w|z)

Since |Z| << |D|, the z layer acts as a "bottleneck" reducing the space..

Each document has a single mixture of z's.

Training

Maximize log likelihood: P(model|data) Use EM

34 Latent Dirichlet Allocation

David Bilie, Andrew Ng, Michael Jordan

Monday, November 19, 2001

35 Information Bottleneck Method

- From "information" to "relevant information"
- what is the information content vs. what is the relevant information content.
- what is the relevant information content?
  - ill-posed question: depends on what we want to know.
    - exact text: trditional information theory
    - what happened?
    - style
    - author
    - political biases
    - etc.
    - We want information *about* something
    - goals:
    - quantify "information about"
    - lossy compression of informaiton sources, preserving the information that we care about

35.1 Formalization

- observed variable X
- variable of interest Y
- how much information does X have about Y?
  - I(X;Y)

Goal:

- summarize X into X˜, preserving information about Y.
- Probablistic summarization rule P(X|X˜)

35.2 Assumptions

- Summary does not carry info about Y that's not already in X
  - Therefore, we have a markov chain, so the following is valid:
    - $P(\tilde{x}|y) = \sum_x p(\tilde{x}|x)P(x|y)$
    - Fix a given compression rate
    - Maximize I(X˜;Y)

35.3 Variational Principle

- Use a lagrange multiplier L[p(x˜|x), T] = I(X˜;Y) - TI(X˜;X)
- Summaries are exhaustive: $\sum_x p(\tilde{x}|x)=1$
- T=0: no compression
- T=∞: sketchy summary
- T = δ I(X˜;Y)/δ I(X˜;X)
- T = $1/\beta$

36 Schedule

36.1 classes

W 21st: ? (I may be gone) MW 26th and 28th: ? first week of december: fernando gone

content

Unknown. :) Maybe more latent variables, or something..

36.2 project

talking project details with fernando: this week or next. Final report due: 5pm on Thursday 13th. Friday 14th and Monday 17th = presentations

37 EM-Based Clustering for NLP

```
1.   Donna read the book
2. # Donna read the truck
3. # The book read Donna
```

all syntactic, (2) and (3) are semantically anomolous

Find verb-argument clusters

- hand-coded lexicon: features (+readable)
- some hidden set of classes
- use EM to find classes

$$P(v,n) = \sum_{c \in C} p(c,v,n) = \sum_{c \in C} p(v|c)p(n|c)p(c)$$

Equivalant to a probablistic grammar, with rules:

$$S \rightarrow N_i V_i$$
$$N_i \rightarrow n_j$$
$$V_i \rightarrow v_k$$

Use the inside-outside algorithm. 2-word "sentences", so we can do this in reasonable time.

Since we're doing separate P(v|c) and P(n|c), we can generalize to new noun-verb combinations.

38 A Winnow-Based Approach to

39 Context-Sensitive Spelling Correction

39.1 Intro

- high dimensional feature space
- target concept only depends on a few features

39.2 Context Sensitive Spelling Correction

- Problem: spelling errors that result in a real but unintended word (homophone, typographic, grammatical, cross-word boundries)
- Approach: WSD
- Confusion set: set of words that might replace each other
  - e.g., {hear, here}

Features:

- Context words (e.g., "cloudy" within ±10 words)
  - captures semantics, topic, etc
  - Collocations (pattern of contiguous words and/or POS tags)
  - e.g., "__ to VERB" ({weather, whether})
  - captures local syntax

39.3 Bayesian Approach

Baseline for comparison. Naive bayes except:

- no independance assumption: detect strong dependencies, try to remove redundant ones. This tries to produce a (relatively) independant model.
- Use smoothing (not just MLE)

39.4 Winnow Approach

$\cong 10^{15}$ items:

- low-level predicates: encode aspects of the current state of the world (i.e., features)
- high-level concepts: learned as functions of the lower-level predicates by a "cloud" or ensemble of classifiers (i.e., confusion sets)

Each confusion set learns its own classifier

Each classifier decides whether a particular word W_i in the confusion set belongs in the target sentence. I.e., decide whether a given word "works" in a given context.

Training (1)

Create connections between clouds and features

We have:

- set of active features
- correct confusion set

$\rightarrow$ positive example for W_c $\rightarrow$ negative example for W_i $i \neq c$

Training algorithm:

- Add connection with weight of 0.1 for each new active feature (for positive example only, not negative feature)
- For each old feature:
  - if negative feature, demote weight (multiply by $.5<\beta<.9$).
  - if positive feature, promote weight (multiply by α=1.5).

Problem: not symmetric; if we see a new feature near the end of training data, it doesn't get affected by demotions for negative occurances..

Weighted Majority

Several parallel classifier clouds decide whether W_i from the confusion set belongs in the sentence. Each classifier is given a weight γ based on its prediction accuracy.

C_j is a classifier (β = 0.5 ... 0.9)
m_j = number of mistakes made by C_j
γ = 1.0 and decreases with # of examples seen
$\sum_j \gamma^{m_j} C_j$ / $\sum_j \gamma^{m_j}$

Use highest activation level to select an outcome

39.5 Results

39.6 Conclusions

40 Verb Clustering & Ambiguity Resolution

Alexandrin A Popesoul

40.1 Clustering Verbs Semantically According to Alternations

Sabine Shulte im Walde

Cluster verbs into semantic classes based on syntactic info and semantic info for the nouns associated with the verbs

"[Verbs can be semantically classified according to their syntactic alternation behavior concerning subcat frames and selectional preferences for args within frames]"

Yay for Levin-esque alternations!

Alternation Behavior

- Syntactic subcat frames
- Semantic WordNet classes

subcat frames: the way that verbs combine with args to form VPs. (focus on objects?)

Refine subcat frames with noun semantic classes: what semantic classes of nouns can they take?

Use WN synsets & hypernyms to group noun phrases

- selectional preferences

corpus: british national corpus (5.5mil sentences)

- frames that appear at leas 2k times (88 frames)
- restrict potential WN classes to 23 nodes

Task:

- cluster 153 manually chosen verbs
  - 226 senses, 30 hand-tagged classes
  - use levin's classification for evaluation

Clustering

- agglomerative
- latent class analysis

Input:

- joint freqs of verbs & subcat frames
- frame slot values for nouns

```
t = subcat frame
v = verb
C = noun class
P(t|v)
P(t,C|v)
```

(doesn't use a coherent probablistic model)

Use agglomerative clustering of P(t|v)

- start with singleton clusters
- join clusters using something like KL-divergence
- restrict cluster size to 4 or less
- re-cluster large clusters
- when do we stop?
- expensive

40.2 Using Probablistic Class-Based Lexicon for Lexical

40.3 Ambiguity Resolution

Datlef Prescher et al

41 ...

41.1 Problem Description: IPS

Inference Problem:

- combine outcomes of several different classifiers in a way that provides a coherent inference that satisfies some constraints.

IPS

IPS = identifying phrase structure

- Instance of inference problem

Problem:

- input string O = o_1, ..., o_n
- phrase = a substring of consecutive symbols
- goal = identify the phrase in a stream

Learn classifiers that can recognize the local signals which are indicitive to the existance of a phrase:

- IO model: a symbol is "inside" or "outside" a phrase (variant = IOB, B = begin a new phrase)
- OC model: a symbol "opens" or "closes" a phrase

We're trying to merge independant classifiers – which makes OC work better. with IO, there's no state, so classifiers can interfere in annoying way. OC allows us to capture some notion of state.

Combine output of the classifiers. Respect constraints:

- phrases can't overlap
- probablistic constraints on order of phrases, lengths, etc.

41.2 General Approaches

Approach 1: Markov Modeling

- probablistic framework that extends HMMs in two ways:
  - simple HMM
  - projection-based HMM

Train HMM with supervised learning

Incorperating constraings:

- constrain the state transition probability (e.g., set transition probabilities to 0 when they are disallowed)

Local signal classifiers:

- NB
- SNoW
- Simple HMM

Incorperate local signal classifiers intto a single HMM framework.

Approach 2: Constratin Satisfaction with Classifiers

CSCL for IPS

- optimal problem
- encode phrases as variables s.t.

```
V = E = {e_i | e_i is a possible phrase}
```

- $f = \bigwedge_{ei \text{ overlaps } ej}(\neg e_i \vee \neg e_j)$ where e_i=1 (0) iff e_i is (not) a phrase
- cost: c: E$\rightarrow\mathbb{R}$

Approach:

- use graphical model, find the shortest path.

Two issues:

- find τ
  - polynomial time (graphical method)
  - use weights, and find shortest path
  - determine cost function c?
  - natural definition: c(e) = 1 - P(o)P(c)
  - use this instead: -P(o)P(c)

41.3 Results

Corpus = WSJ in Penn Treebank Compare CSCL, HMM, PHMM. Each uses all 3 classifiers

CSCL outperforms PHMM outperforms HMM.

42 Document Modeling with Latent Class Models

Alexandrin A Popescul

Data set:

- documents from citeseer
- "text" and "learning", plus citations
- remove stop words
- porter stemmer
- keep 3k most frequent word ($\geq$ 15 tokens)

Authomatically cluster documents..

Model: $\sum_z$ P(z)P(d|z)P(w|z)

5 latent classes

Hard clusters.

- Assign each document to z_d=argmax$_z$P(z|d)
- Clusters vary in size..