



Maxent Models and Discriminative Estimation

Generative vs. Discriminative models

borrowing from Christopher Manning

Introduction

- So far we've looked at "generative models"
 - Language models, Naive Bayes
- But there is now much use of conditional or discriminative probabilistic models in NLP, Speech, IR (and ML generally)
- Because:
 - They give high accuracy performance
 - They make it easy to incorporate lots of linguistically important features

Joint vs. Conditional Models

- We have some data $\{(d, c)\}$ of paired observations d and hidden classes c .
- **Joint (generative) models** place probabilities over both observed data and the hidden stuff (generate the observed data from hidden stuff):
 - All the classic StatNLP models:
 - n -gram models, Naive Bayes classifiers, hidden Markov models, probabilistic context-free grammars, IBM machine translation alignment models

$$P(c, d)$$

Joint vs. Conditional Models

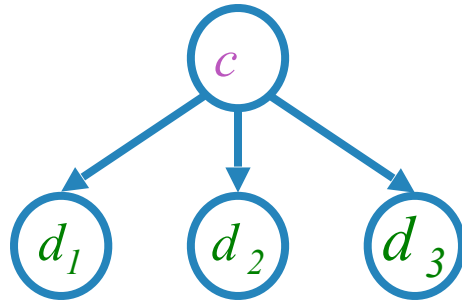
- **Discriminative (conditional) models** take the data as given, and put a probability over hidden structure given the data:

$$P(c|d)$$

- Logistic regression, conditional loglinear or maximum entropy models, conditional random fields
- Also, SVMs, (averaged) perceptron, etc. are discriminative classifiers (but not directly probabilistic)

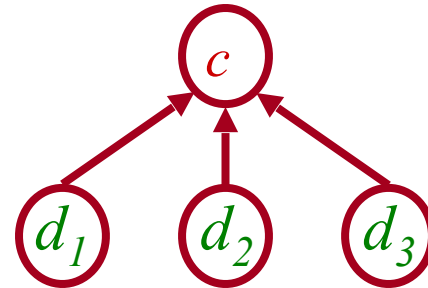
Bayes Net/Graphical Models

- Bayes net diagrams draw circles for random variables, and lines for direct dependencies
- Some variables are observed; some are hidden
- Each node is a little classifier (conditional probability table) based on incoming arcs



Naive Bayes

Generative



Logistic Regression

Discriminative

Conditional vs. Joint Likelihood

- A *joint* model gives probabilities $P(d,c)$ and tries to maximize this joint likelihood.
 - It turns out to be trivial to choose weights: just relative frequencies.
- A *conditional* model gives probabilities $P(c|d)$. It takes the data as given and models only the conditional probability of the class.
 - We seek to maximize conditional likelihood.
 - Harder to do (as we'll see...)
 - More closely related to classification error.

Conditional models work well: Word Sense Disambiguation

Training Set	
Objective	Accuracy
Joint Like.	86.8
Cond. Like.	98.5

Test Set	
Objective	Accuracy
Joint Like.	73.6
Cond. Like.	76.1

- Even with exactly the same features, changing from joint to conditional estimation increases performance
- That is, we use the same smoothing, and the same word-class features, we just change the numbers (parameters)

(Klein and Manning 2002, using Senseval-1 Data)



Discriminative Model Features

Making features from text for discriminative NLP models

borrowing from Christopher Manning

Features

- In these slides and most maxent work: *features* f are elementary pieces of evidence that link aspects of what we observe d with a category c that we want to predict
- A feature is a function with a bounded real value

Example features

- $f_1(c, d) \equiv [c = \text{LOCATION} \wedge w_{-1} = \text{“in”} \wedge \text{isCapitalized}(w)]$
- $f_2(c, d) \equiv [c = \text{LOCATION} \wedge \text{hasAccentedLatinChar}(w)]$
- $f_3(c, d) \equiv [c = \text{DRUG} \wedge \text{ends}(w, \text{“c”})]$



- Models will assign to each feature a *weight*:
 - A positive weight votes that this configuration is likely correct
 - A negative weight votes that this configuration is likely incorrect

Feature Expectations

- We will crucially make use of two *expectations*
 - actual or predicted counts of a feature firing:

- Empirical count (expectation) of a feature:

$$\text{empirical } E(f_i) = \sum_{(c,d) \in \text{observed}(C,D)} f_i(c, d)$$

- Model expectation of a feature:

$$E(f_i) = \sum_{(c,d) \in (C,D)} P(c, d) f_i(c, d)$$

Features

- In NLP uses, usually a feature specifies
 1. an indicator function – a yes/no boolean matching function – of properties of the input and
 2. a particular class

$$f_i(c, d) \equiv [\Phi(d) \wedge c = c_j] \quad \text{[Value is 0 or 1]}$$

- Each feature picks out a data subset and suggests a label for it

Feature-Based Models

- The decision about a data point is based only on the **features** active at that point.

Data BUSINESS: Stocks hit a yearly low ...
Label: BUSINESS Features {..., stocks, hit, a, yearly, low, ...}

Text
Categorization

Data ... to restructure bank:MONEY debt.
Label: MONEY Features {..., w_{-1} =restructure, w_{+1} =debt, L=12, ...}

Word-Sense
Disambiguation

Data DT JJ NN ... The previous fall ...
Label: NN Features { w =fall, t_{-1} =JJ w_{-1} =previous}

POS Tagging

Example: Text Categorization

(Zhang and Oles 2001)

- Features are presence of each **word** in a document and the document **class** (they do feature selection to use reliable indicator words)
- Tests on classic Reuters data set (and others)
 - Naïve Bayes: 77.0% F_1
 - Linear regression: 86.0%
 - **Logistic regression: 86.4%**
 - Support vector machine: 86.5%
- Paper emphasizes the importance of *regularization* (smoothing) for successful use of discriminative methods (not used in much early NLP/IR work)

Other Maxent Classifier Examples

- You can use a maxent classifier whenever you want to assign data points to one of a number of classes:
 - Sentence boundary detection (Mikheev 2000)
 - Is a period end of sentence or abbreviation?
 - Sentiment analysis (Pang and Lee 2002)
 - Word unigrams, bigrams, POS counts, ...
 - PP attachment (Ratnaparkhi 1998)
 - Attach to verb or noun? Features of head noun, preposition, etc.
 - Parsing decisions in general (Ratnaparkhi 1997; Johnson et al. 1999, etc.)



Feature-based Linear Classifiers

How to put features into a classifier

borrowing from Christopher Manning

Feature-Based Linear Classifiers

- Linear classifiers at classification time:
 - Linear function from feature sets $\{f_i\}$ to classes $\{c\}$.
 - Assign a weight λ_i to each feature f_i .
 - We consider each class for an observed datum d
 - For a pair (c,d) , features vote with their weights:
 - $\text{vote}(c) = \sum \lambda_i f_i(c,d)$

PERSON
in Québec

LOCATION
in Québec

DRUG
in Québec

- Choose the class c which maximizes $\sum \lambda_i f_i(c,d)$

Feature-Based Linear Classifiers

- Linear classifiers at classification time:
 - Linear function from feature sets $\{f_i\}$ to classes $\{c\}$.
 - Assign a weight λ_i to each feature f_i .
 - We consider each class for an observed datum d
 - For a pair (c,d) , features vote with their weights:
 - $\text{vote}(c) = \sum \lambda_i f_i(c,d)$

PERSON
in Québec

1.8 LOCATION
in Québec -0.6

0.3 DRUG
in Québec

- Choose the class c which maximizes $\sum \lambda_i f_i(c,d) = \text{LOCATION}$

Feature-Based Linear Classifiers

There are many ways to choose weights for features

- Perceptron: find a currently misclassified example, and nudge weights in the direction of its correct classification
- Margin-based methods (Support Vector Machines)

Feature-Based Linear Classifiers

- Exponential (log-linear, maxent, logistic, Gibbs) models:
 - Make a probabilistic model from the linear combination $\sum \lambda_i f_i(c, d)$

$$P(c | d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}$$

Makes votes positive

Normalizes votes

- $P(\text{LOCATION} | \text{in Québec}) = e^{1.8} e^{-0.6} / (e^{1.8} e^{-0.6} + e^{0.3} + e^0) = 0.586$
- $P(\text{DRUG} | \text{in Québec}) = e^{0.3} / (e^{1.8} e^{-0.6} + e^{0.3} + e^0) = 0.238$
- $P(\text{PERSON} | \text{in Québec}) = e^0 / (e^{1.8} e^{-0.6} + e^{0.3} + e^0) = 0.176$
- The **weights** are the **parameters** of the probability model, combined via a “soft max” function

Feature-Based Linear Classifiers

- Exponential (log-linear, maxent, logistic, Gibbs) models:
 - Given this model form, we will choose parameters $\{\lambda_j\}$ that *maximize the conditional likelihood* of the data according to this model.
 - We construct not only classifications, but probability distributions over classifications.
 - There are other (good!) ways of discriminating classes – SVMs, boosting, even perceptrons – but these methods are not as trivial to interpret as distributions over classes.

Aside: logistic regression

- Maxent models in NLP are essentially the same as multiclass logistic regression models in statistics (or machine learning)
 - If you haven't seen these before, don't worry, this presentation is self-contained!
 - If you have seen these before you might think about:
 - The parameterization is slightly different in a way that is advantageous for NLP-style models with tons of sparse features (but statistically inelegant)
 - The key role of feature functions in NLP and in this presentation
 - The features are more general, with f also being a function of the class

Quiz Question

- Assuming exactly the same set up (3 class decision: LOCATION, PERSON, or DRUG; 3 features as before, maxent), what are:

- $P(\text{PERSON} \mid \textit{by Goéric}) =$

- $P(\text{LOCATION} \mid \textit{by Goéric}) =$

- $P(\text{DRUG} \mid \textit{by Goéric}) =$

- $1.8 \quad f_1(c, d) \equiv [c = \text{LOCATION} \wedge w_{-1} = \textit{in} \wedge \text{isCapitalized}(w)]$

- $-0.6 \quad f_2(c, d) \equiv [c = \text{LOCATION} \wedge \text{hasAccentedLatinChar}(w)]$

- $0.3 \quad f_3(c, d) \equiv [c = \text{DRUG} \wedge \text{ends}(w, \textit{c})]$

PERSON
by Goéric

LOCATION
by Goéric

DRUG
by Goéric

$$P(c \mid d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}$$



Building a Maxent Model

The nuts and bolts

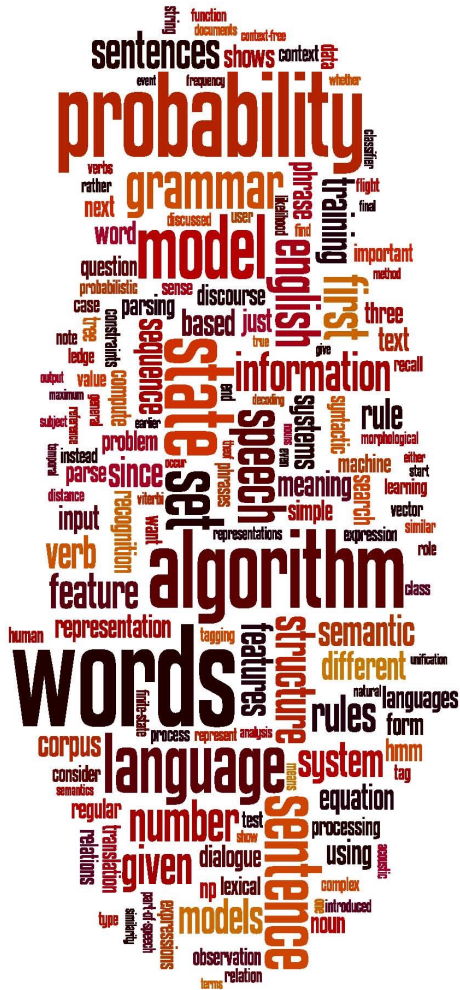
borrowing from Christopher Manning

Building a Maxent Model

- We define features (indicator functions) over data points
 - Features represent sets of data points which are distinctive enough to deserve model parameters.
 - Words, but also “word contains number”, “word ends with *ing*”, etc.
- We will simply encode each Φ feature as a unique String
 - A datum will give rise to a set of Strings: the active Φ features
 - Each feature $f_i(c, d) \equiv [\Phi(d) \wedge c = c_j]$ gets a real number weight
- We concentrate on Φ features but the math uses i indices of f_i

Building a Maxent Model

- Features are often added during model development to target errors
 - Often, the easiest thing to think of are features that mark bad combinations
- Then, for any given feature weights, we want to be able to calculate:
 - Data conditional likelihood
 - Derivative of the likelihood wrt each feature weight
 - Uses expectations of each feature according to the model
- We can then find the optimum feature weights (discussed later).



Building a Maxent Model

The nuts and bolts

borrowing from Christopher Manning

Text classification: Asia or Europe

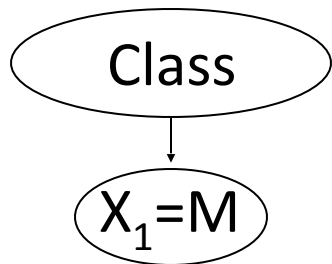
Europe

Training Data

Asia



NB Model



NB FACTORS:

- $P(A) = P(E) =$
- $P(M|A) =$
- $P(M|E) =$

PREDICTIONS:

- $P(A, M) =$
- $P(E, M) =$
- $P(A|M) =$
- $P(E|M) =$

Text classification: Asia or Europe

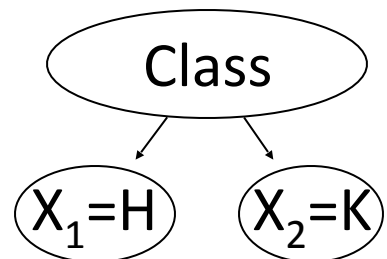
Europe

Training Data

Asia



NB Model



NB FACTORS:

- $P(A) = P(E) =$
- $P(H|A) = P(K|A) =$
- $P(H|E) = P(K|E) =$

PREDICTIONS:

- $P(A,H,K) =$
- $P(E,H,K) =$
- $P(A|H,K) =$
- $P(E|H,K) =$

Text classification: Asia or Europe

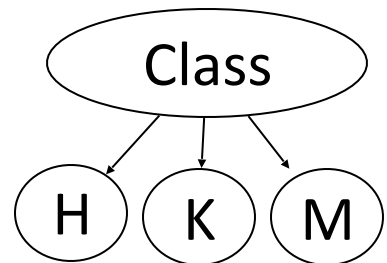
Europe

Training Data

Asia



NB Model



NB FACTORS:

- $P(A) = P(E) =$
- $P(M|A) =$
- $P(M|E) =$
- $P(H|A) = P(K|A) =$
- $P(H|E) = P(K|E) =$

PREDICTIONS:

- $P(A, H, K, M) =$
- $P(E, H, K, M) =$
- $P(A | H, K, M) =$
- $P(E | H, K, M) =$

Naive Bayes vs. Maxent Models

- Naive Bayes models multi-count correlated evidence
 - Each feature is multiplied in, even when you have multiple features telling you the same thing
- Maximum Entropy models (pretty much) solve this problem
 - As we will see, this is done by weighting features so that model expectations match the observed (empirical) expectations



Naive Bayes vs. Maxent models

Generative vs. Discriminative models: The problem of overcounting evidence

borrowing from Christopher Manning



Maxent Models and Discriminative Estimation

Maximizing the likelihood

borrowing from Christopher Manning

Exponential Model Likelihood

- Maximum (Conditional) Likelihood Models :
 - Given a model form, choose values of parameters to maximize the (conditional) likelihood of the data.

$$\log P(C | D, \lambda) = \sum_{(c,d) \in (C,D)} \log P(c | d, \lambda) = \sum_{(c,d) \in (C,D)} \log \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}$$

The Likelihood Value

- The (log) conditional likelihood of iid data (C, D) according to maxent model is a function of the data and the parameters λ :

$$\log P(C | D, \lambda) = \log \prod_{(c, d) \in (C, D)} P(c | d, \lambda) = \sum_{(c, d) \in (C, D)} \log P(c | d, \lambda)$$

- If there aren't many values of c , it's easy to calculate:

$$\log P(C | D, \lambda) = \sum_{(c, d) \in (C, D)} \log \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}$$

The Likelihood Value

- We can separate this into two components:

$$\log P(C | D, \lambda) = \sum_{(c,d) \in (C,D)} \log \exp \sum_I \lambda_i f_i(c, d) - \sum_{(c,d) \in (C,D)} \log \sum_{\mathbf{c}} \exp \sum_I \lambda_i f_i(\mathbf{c}, d)$$

$$\log P(C | D, \lambda) = N(\lambda) - M(\lambda)$$

- The derivative is the difference between the derivatives of each component

The Derivative I: Numerator

$$\begin{aligned}\frac{\partial N(\lambda)}{\partial \lambda_i} &= \frac{\partial \sum_{(c,d) \in (C,D)} \log \exp \sum_I \lambda_{c_i} f_i(c, d)}{\partial \lambda_i} = \frac{\partial \sum_{(c,d) \in (C,D)} \sum_I \lambda_i f_i(c, d)}{\partial \lambda_i} \\ &= \sum_{(c,d) \in (C,D)} \frac{\partial \sum_I \lambda_i f_i(c, d)}{\partial \lambda_i} \\ &= \sum_{(c,d) \in (C,D)} f_i(c, d)\end{aligned}$$

Derivative of the numerator is: the empirical count(f_i, c)

The Derivative II: Denominator

$$\begin{aligned}
 \frac{\partial M(\lambda)}{\partial \lambda_i} &= \frac{\partial \sum_{(c,d) \in (C,D)} \log \sum_{c'} \exp \sum_I \lambda_i f_i(c', d)}{\partial \lambda_i} \\
 &= \sum_{(c,d) \in (C,D)} \frac{1}{\sum_{c''} \exp \sum_I \lambda_i f_i(c'', d)} \frac{\partial \sum_{c'} \exp \sum_I \lambda_i f_i(c', d)}{\partial \lambda_i} \\
 &= \sum_{(c,d) \in (C,D)} \frac{1}{\sum_{c''} \exp \sum_I \lambda_i f_i(c'', d)} \sum_{c'} \frac{\exp \sum_I \lambda_i f_i(c', d) \partial \sum_I \lambda_i f_i(c', d)}{1 \partial \lambda_i} \\
 &= \sum_{(c,d) \in (C,D)} \sum_{c'} \frac{\exp \sum_I \lambda_i f_i(c', d)}{\sum_{c''} \exp \sum_I \lambda_i f_i(c'', d)} \frac{\partial \sum_I \lambda_i f_i(c', d)}{\partial \lambda_i} \\
 &= \sum_{(c,d) \in (C,D)} \sum_{c'} P(c' | d, \lambda) f_i(c', d) = \text{predicted count}(f_i, \lambda)
 \end{aligned}$$

The Derivative III

$$\frac{\partial \log P(\mathbf{C} | D, \lambda)}{\partial \lambda_j} = \text{actual count}(f_j, \mathbf{C}) - \text{predicted count}(f_j, \lambda)$$

- The optimum parameters are the ones for which each feature's **predicted expectation** equals its **empirical expectation**. The optimum distribution is:
 - Always unique (but parameters may not be unique)
 - Always exists (if feature counts are from actual data).

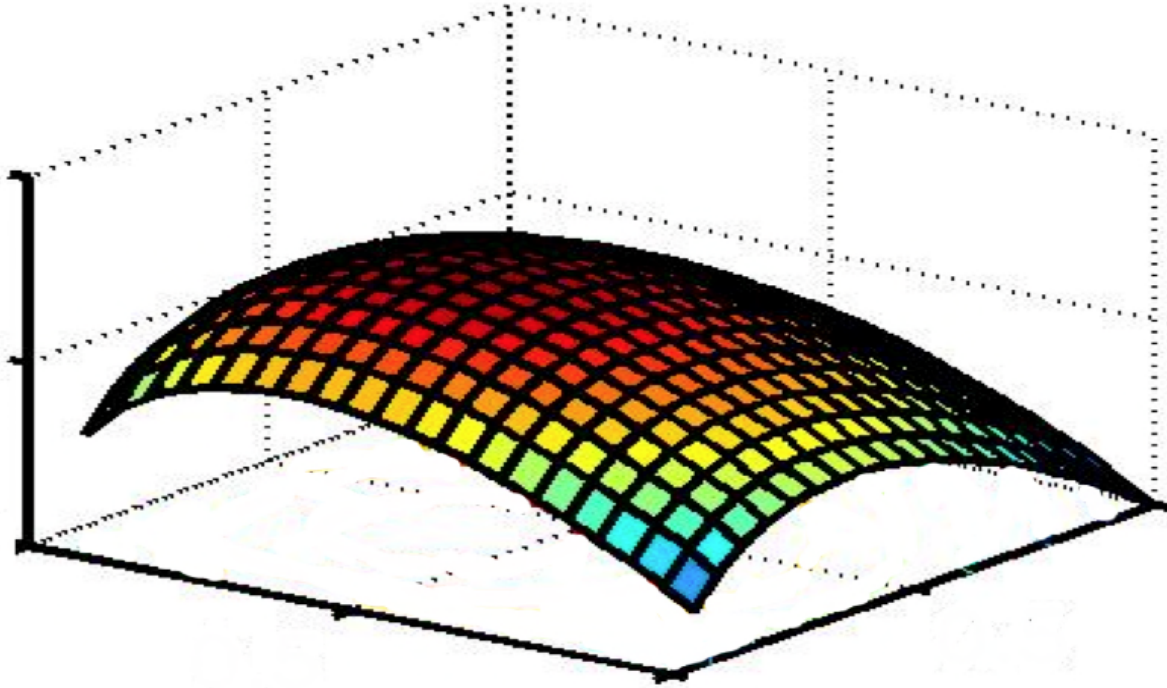
Finding the optimal parameters

- We want to choose parameters $\lambda_1, \lambda_2, \lambda_3, \dots$ that maximize the conditional log-likelihood of the training data

$$CLogLik(D) = \sum_{i=1}^n \log P(c_i | d_i)$$

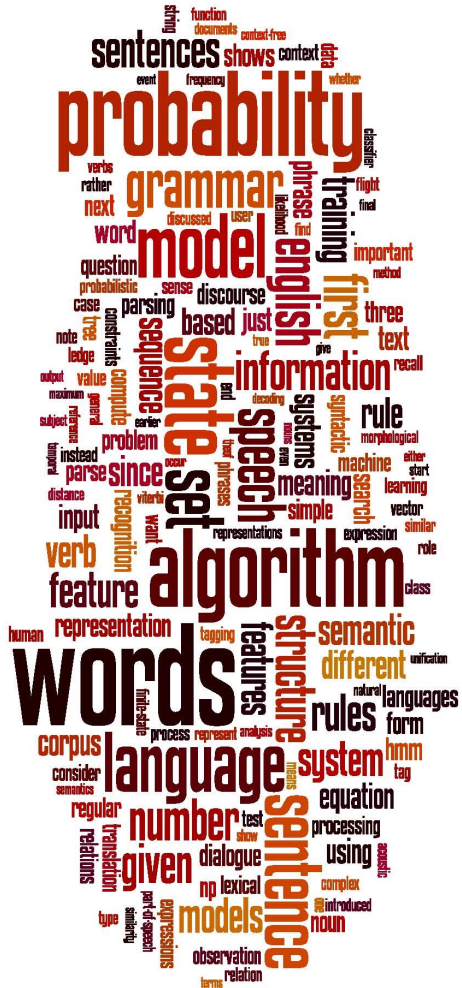
- To be able to do that, we've worked out how to calculate the function value and its partial derivatives (its gradient)

A likelihood surface



Finding the optimal parameters

- Use your favorite numerical optimization package....
 - Commonly, you **minimize** the negative of $C\text{LogLik}$
 1. Gradient descent (GD); Stochastic gradient descent (SGD)
 2. Iterative proportional fitting methods: Generalized Iterative Scaling (GIS) and Improved Iterative Scaling (IIS)
 3. Conjugate gradient (CG), perhaps with preconditioning
 4. Quasi-Newton methods – limited memory variable metric (LMVM) methods, in particular, L-BFGS



Maxent Models and Discriminative Estimation

Maximizing the likelihood

borrowing from Christopher Manning