Maxent Models and Discriminative Estimation

Generative vs. Discriminative models

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
  - They allow automatic building of language independent, retargetable NLP modules

Christopher Manning

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

$P(c|d)$

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

Logistic Regression

Generative

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

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

- 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)

Maxent Models and Discriminative Estimation

Generative vs. Discriminative models

Christopher Manning

Discriminative Model Features

Making features from text for discriminative NLP models

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

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

LOCATION	LOCATION	DRUG	PERSON
in Arcadia	*in Québec*	*taking Zantac*	*saw Sue*

- 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

$$f_i(c, d) \equiv [\Phi(d) \wedge c = c_j]$$

[Value is 0 or 1]

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

Christopher Manning

Feature-Based Models

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

Text Categorization	Word-Sense Disambiguation	POS Tagging
Data BUSINESS: Stocks hit a yearly low …	Data … to restructure bank:MONEY debt.	Data DT JJ NN … The previous fall …
Label: BUSINESS Features {…, stocks, hit, a, yearly, low, …}	Label: MONEY Features {…, w_{-1}=restructure, w_{+1}=debt, L=12, …}	Label: NN Features {w=fall, t_{-1}=JJ w_{-1}=previous}

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.)

Discriminative Model Features

Making features from text for discriminative NLP models

Christopher Manning

Feature-based Linear Classifiers

How to put features into a classifier

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:
    - vote(c) = $\Sigma\lambda_i f_i(c,d)$

PERSON
in Québec

LOCATION
in Québec

DRUG
in Québec

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

Feature-Based Linear Classifiers

There are many ways to chose 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 $\Sigma\lambda_i f_i(c,d)$

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

Makes votes positive (numerator)

Normalizes votes (denominator)

- P(LOCATION|*in Québec*) = $e^{1.8}e^{-0.6}/(e^{1.8}e^{-0.6} + e^{0.3} + e^{0}) = 0.586$
- P(DRUG|*in Québec*) = $e^{0.3}/(e^{1.8}e^{-0.6} + e^{0.3} + e^{0}) = 0.238$
- P(PERSON|*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 $\{\lambda_i\}$ 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 – when might this be useful?

Quiz Question

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

- 1.8 $f_1(c, d) \equiv [c = \text{LOCATION} \wedge w_{-1} = \text{“in”} \wedge \text{isCapitalized}(w)]$
- -0.6 $f_2(c, d) \equiv [c = \text{LOCATION} \wedge \text{hasAccentedLatinChar}(w)]$
- 0.3 $f_3(c, d) \equiv [c = \text{DRUG} \wedge \text{ends}(w, \text{“c”})]$

PERSON *by Goéric*

LOCATION *by Goéric*

DRUG *by Goéric*

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

Feature-based Linear Classifiers

How to put features into a classifier

Building a Maxent Model

The nuts and bolts

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

Christopher Manning

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

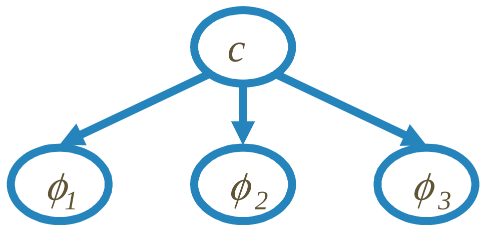
Naive Bayes vs. Maxent models

Generative vs. Discriminative models: Two examples of overcounting evidence

Christopher Manning

Comparison to Naïve-Bayes

- Naïve-Bayes is another tool for classification:
  - We have a bunch of random variables (data features) which we would like to use to predict another variable (the class):
  - The Naïve-Bayes likelihood over classes is:



$$P(c \mid d, \lambda) = \frac{P(c)\prod_i P(\phi_i \mid c)}{\sum_{c'} P(c')\prod_i P(\phi_i \mid c')} \Rightarrow \frac{\exp\left[\log P(c) + \sum_i \log P(\phi_i \mid c)\right]}{\sum_{c'} \exp\left[\log P(c') + \sum_i \log P(\phi_i \mid c')\right]}$$

$$\Rightarrow \frac{\exp\left[\sum_i \lambda_{ic} f_{ic}(d, c)\right]}{\sum_{c'} \exp\left[\sum_i \lambda_{ic'} f_{ic'}(d, c')\right]}$$

Naïve-Bayes is just an exponential model.

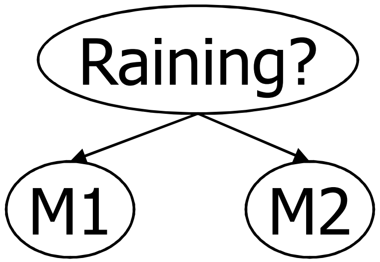
Christopher Manning

Example: Sensors

Reality: sun and rain equiprobable

P(+,+,r) = 3/8 P(–,–,r) = 1/8 P(+,+,s) = 1/8 P(–,–,s) = 3/8

NB Model



NB FACTORS:

- P(s) =
- P(+|s) =
- P(+|r) =

PREDICTIONS:

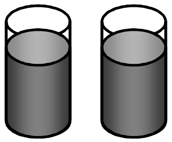
- P(r,+,+) =
- P(s,+,+) =
- P(r|+,+) =
- P(s|+,+) =

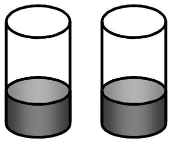
Christopher Manning

Example: Sensors

Reality

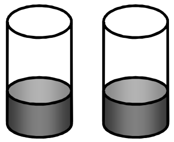
Raining

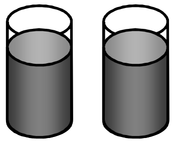




$P(+,+,r) = 3/8$ $P(–,–,r) = 1/8$

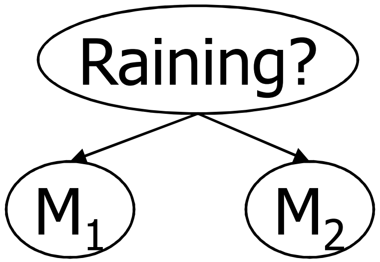
Sunny





$P(+,+,s) = 1/8$ $P(–,–,s) = 3/8$

NB Model



NB FACTORS:

- $P(s) = 1/2$
- $P(+|s) = 1/4$
- $P(+|r) = 3/4$

PREDICTIONS:

- $P(r,+,+) = (½)(¾)(¾)$
- $P(s,+,+) = (½)(¼)(¼)$
- $P(r|+,+) = 9/10$
- $P(s|+,+) = 1/10$

Example: Sensors

- Problem: NB multi-counts the evidence

$$\frac{P(r \mid M_1=+,\ldots,M_n=+)}{P(s \mid M_1=+,\ldots,M_n=+)} = \frac{P(r)}{P(s)}\frac{P(M_1=+ \mid r)}{P(M_1=+ \mid s)}\cdots\frac{P(M_n=+ \mid r)}{P(M_n=+ \mid s)}$$

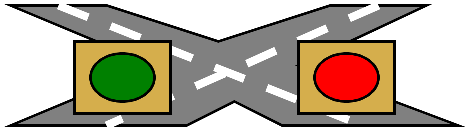
Example: Sensors

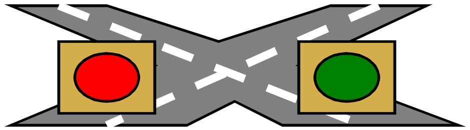
- Maxent behavior:
  - Take a model over $(M_1, \ldots M_n, R)$ with features:
    - $\boldsymbol{f_{ri}}$: $\mathbf{M_i=+, R=r}$ weight: λ_{ri}
    - $\boldsymbol{f_{si}}$: $\mathbf{M_i=+, R=s}$ weight: λ_{si}
  - $\exp(\lambda_{ri} - \lambda_{si})$ is the factor analogous to $P(+|r)/P(+|s)$
  - … but instead of being 3, it will be $3^{1/n}$
  - … because if it were 3, $E[f_{ri}]$ would be far higher than the target of 3/8!

Example: Stoplights

Reality

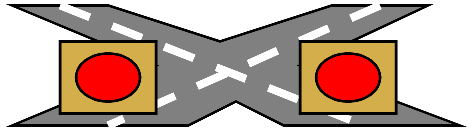
Lights Working





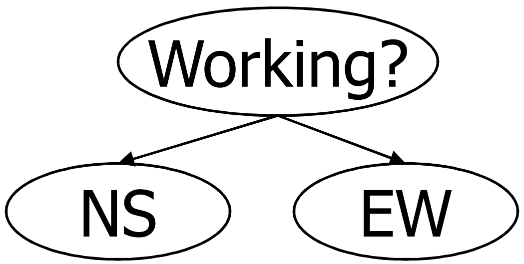
$P(g,r,w) = 3/7$ $P(r,g,w) = 3/7$

Lights Broken



$P(r,r,b) = 1/7$

NB Model



NB FACTORS:

- $P(w) =$
- $P(r|w) =$
- $P(g|w) =$

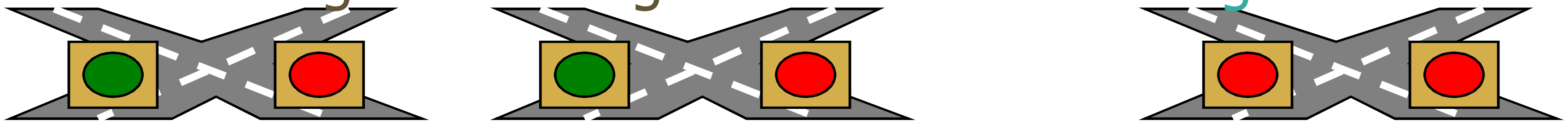
- $P(b) =$
- $P(r|b) =$
- $P(g|b) =$

Example: Stoplights

Reality

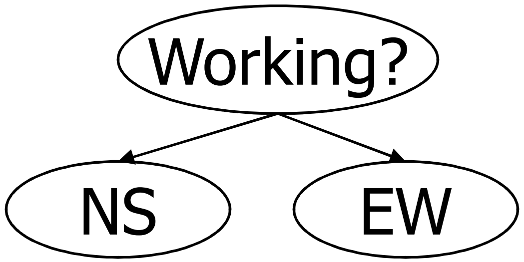
Lights Working

Lights Broken



$P(g,r,w) = 3/7$ $P(r,g,w) = 3/7$ $P(r,r,b) = 1/7$

NB Model



NB FACTORS:

- $P(w) = 6/7$
- $P(r|w) = 1/2$
- $P(g|w) = 1/2$

- $P(b) = 1/7$
- $P(r|b) = 1$
- $P(g|b) = 0$

Example: Stoplights

- What does the model say when both lights are red?
  - P(b,r,r) =
  - P(w,r,r) =
  - P(w|r,r) =

- We'll guess that (r,r) indicates the lights are working!

Example: Stoplights

- What does the model say when both lights are red?
  - P(b,r,r) = (1/7)(1)(1) = 1/7 = 4/28
  - P(w,r,r) = (6/7)(1/2)(1/2) = 6/28 = 6/28
  - P(w|r,r) = 6/10 !!

- We'll guess that (r,r) indicates the lights are working!

Example: Stoplights

- Now imagine if P(b) were boosted higher, to ½:
  - P(b,r,r) =
  - P(w,r,r) =
  - P(w|r,r) =

- Changing the parameters bought conditional accuracy at the expense of data likelihood!
  - The classifier now makes the right decisions

Example: Stoplights

- Now imagine if P(b) were boosted higher, to ½:
  - P(b,r,r) = (1/2)(1)(1) = 1/2 = 4/8
  - P(w,r,r) = (1/2)(1/2)(1/2) = 1/8 = 1/8
  - P(w|r,r) = 1/5!
- Changing the parameters bought conditional accuracy at the expense of data likelihood!
  - The classifier now makes the right decisions

Naive Bayes vs. Maxent models

Generative vs. Discriminative models: Two examples of overcounting evidence

Christopher Manning

Maxent Models and Discriminative Estimation

Maximizing the likelihood

Exponential Model Likelihood

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

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

The Likelihood Value

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

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

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

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

The Likelihood Value

- We can separate this into two components:

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

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

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

The Derivative I: Numerator

$$\frac{\partial N(\lambda)}{\partial \lambda_i} = \frac{\partial \sum_{(c,d)\in(C,D)} \log \exp \sum_i \lambda_{ci} f_i(c,d)}{\partial \lambda_i} = \frac{\partial \sum_{(c,d)\in(C,D)} \sum_i \lambda_i f_i(c,d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \frac{\partial \sum_i \lambda_i f_i(c,d)}{\partial \lambda_i}$$

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

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

The Derivative II: Denominator

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

$$= \sum_{(c,d)\in(C,D)} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'',d)} \frac{\partial \sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'',d)} \sum_{c'} \frac{\exp \sum_i \lambda_i f_i(c',d)}{1} \frac{\partial \sum_i \lambda_i f_i(c',d)}{\partial \lambda_i}$$

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

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

= predicted count(f_i, λ)

The Derivative III

$$\frac{\partial \log P(C \mid D, \lambda)}{\partial \lambda_i} = \text{actual count}(f_i, C) - \text{predicted count}(f_i, \lambda)$$

- 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).
- These models are also called maximum entropy models because we find the model having maximum entropy and satisfying the constraints: $E_p(f_j) = E_{\tilde{p}}(f_j), \forall j$

Fitting the Model

- To find the parameters $\lambda_1, \lambda_2, \lambda_3$

write out the conditional log-likelihood of the training data and maximize it

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

- The log-likelihood is concave and has a single maximum; use your favorite numerical optimization package….

Fitting the Model
Generalized Iterative Scaling

- A simple optimization algorithm which works when the features are non-negative
- We need to define a slack feature to make the features sum to a constant over all considered pairs from $D \times C$
- Define $$M = \max_{i,c} \sum_{j=1}^{m} f_j(d_i, c)$$
- Add new feature $$f_{m+1}(d,c) = M - \sum_{j=1}^{m} f_j(d,c)$$

Generalized Iterative Scaling

- Compute empirical expectation for all features

$$E_{\tilde{p}}(f_j) = \frac{1}{N}\sum_{i=1}^{n} f_j(d_i, c_i)$$

- Initialize $\lambda_j = 0,\ j = 1 \ldots m+1$

Generalized Iterative Scaling

- Repeat
  - Compute feature expectations according to current model

$$E_{p^t}(f_j) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} P(c_k \mid d_i) f_j(d_i, c_k)$$

  - Update parameters

$$\lambda_j^{(t+1)} = \lambda_j^{(t)} + \frac{1}{M} \log\left(\frac{E_{\tilde{p}}(f_j)}{E_{p^t}(f_j)} \right)$$

- Until converged

Fitting the Model

- In practice, people have found that good general purpose numeric optimization packages/methods work better
- Conjugate gradient or limited memory quasi-Newton methods (especially, L-BFGS) is what is generally used these days
- Stochastic gradient descent can be better for huge problems

Maxent Models and Discriminative Estimation

Maximizing the likelihood