

# Naïve Bayes, Maximum Entropy and Text Classification

COSI 134



# Conditional Parameterization

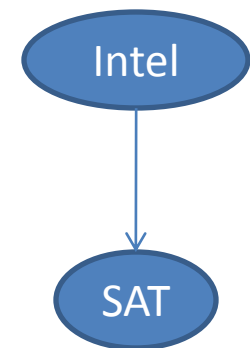
- Two RVs: Intelligence(I) and SAT(S)
- $\text{Val}(I) = \{\text{High}, \text{Low}\}$ ,  $\text{Val}(S) = \{\text{High}, \text{Low}\}$
- A possible joint distribution
- Can describe using chain rule as

$$P(I,S) = P(I)P(S|I)$$

I	S	P(I,S)
Low	Low	0.665
Low	High	0.035
High	Low	0.06
High	High	0.24

P(I=Low)	P(I=High)
0.7	0.3

P(S I)	S=Low	S=High
I=Low	0.95	0.05
I=High	0.2	0.8



# Conditional Independence

- Assume another RV, Grade(G)

- Grade in some course

- $\text{Val}(G) = \{\text{High, Medium, Low}\}$

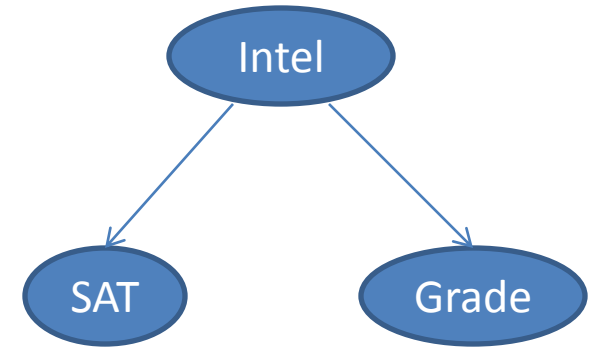
- Might assume that G is conditionally independent of S given I

$$P(G | I, S) = P(G | I)$$

- Then:  $P(I, S, G) = P(S, G | I)P(I)$

By cond.indep.  $P(S, G | I) = P(S | I)P(G | I)$

So,  $P(I, S, G) = P(S | I)P(G | I)P(I)$



- Another CPT for  $P(G | I)$

- More compact than full joint

- Possible to update joint with new information

$P(G   I)$	G=High	G=Med	G=Low
I=Low	0.2	0.34	0.46
I=High	0.74	0.17	0.09

# Statistical Modeling

- Four Questions

- 1) What is the form of the **model**?
  - What random variables? How are probabilities computed? What distributions? What parameters?
- 2) Given a set of data (items from the sample space), how is the **likelihood** of that data computed, for the given model structure and parameter values?
- 3) Given a likelihood function, how are the “optimal” parameters **estimated** given a set of data?
- 4) Given a model form and a set of induced parameter values, how is **inference** performed in the model to make predictions/ask queries

# Random Variable Distributions

- Bernoulli Distribution

- Outcome is success (1) or failure (0)
- Success with probability  $p$
- Probability mass function  $P(X=1) = 1 - P(X=0) = p$

- Categorical Distribution

- Outcome is one of a finite number of categories
- Probability mass function  $P(X = x_i) = p_i$   $\sum_{i=1}^n p_i = 1$

- Binomial Distribution is a series of Bernoulli trials

- Multinomial Distribution is a series of Categorical trials

# Naïve Bayes

- Very simple, but effective probabilistic classifier

$$p(y | x_1, \dots, x_n) = \frac{p(y, x_1, \dots, x_n)}{p(x_1, \dots, x_n)} = \frac{p(x_1, \dots, x_n | y)p(y)}{p(x_1, \dots, x_n)}$$

- But – how do we calculate  $p(x_1, \dots, x_n | y)$

- Naïve Bayes Assumption:  $p(x_1, \dots, x_n | y) = \prod_{i=1}^n p(x_i | y)$

- Each observed variable is assumed to be independent of each other given the class

# Naïve Bayes Inference

- First, note that to use the model in most settings, we do not need to explicitly compute

$$\frac{p(x_1, \dots, x_n | y)p(y)}{p(x_1, \dots, x_n)}$$

- Denominator can be ignored since the data are given and the same across all  $y$
- We are interested in

$$\begin{aligned} \arg \max_y (p(y | x_1, \dots, x_n)) &= \arg \max_y \frac{p(x_1, \dots, x_n | y)p(y)}{p(x_1, \dots, x_n)} \\ &= \arg \max_y p(x_1, \dots, x_n | y)p(y) \end{aligned}$$

# Example: Document Classification

DOCUMENTS:

To finance extra spending on Labour's policies, such as education, Mr. Brown announced that the Treasury would collect 30 billion pounds by selling national assets like the Tote as well as government shares in British Energy and the .....

FINANCE

England have won the third Test at Mumbai by 212 runs and secured a share of the series in which few observers, if any, gave them hope of avoiding defeat. Set 313 to win, India folded to 100 all out an hour and a half into the afternoon session, with their ...

SPORTS

Classify documents based on their vocabulary.

$$p(\text{class} = C \mid w_{\text{Brown}} = 1, w_{\text{finance}} = 1, w_{\text{spending}} = 1, w_{\text{Treasury}} = 1, \dots)$$

# Observed Variables in NB

- The X variables in  $p(x_1, \dots, x_n | y)$
- Bernoulli model introduces a set of Bernoulli RVs, one for each item in our vocabulary, such that  $X_w = 1$  iff  $w$  appears in the document
- The multinomial model introduces an RV for each position in a document. The RV is multinomial, ranging over the vocabulary
  - E.g.  $X_1 = \textit{England}, X_2 = \textit{have}, X_3 = \textit{won}$
  - But, we'd like positional independence

$$p(X_i = \textit{England} | C) = p(X_j = \textit{England} | C)$$

# Generative Story

- Bernoulli Case

- 1) Generate a document class from  $p(y)$
- 2) Generate an indicator variable  $X_i$  for each vocabulary item
- 3) Generate words according to which  $X_i = 1$

- Multinomial Case

- 1) Generate a document class from  $p(y)$
- 2) For each position  $k$ , generate a word from  $p(X_k = w | C)$
- 3) Do this for all positions in document
  - Note that true generative model would require modeling document length

# Estimation

- Maximum likelihood estimation  $p(D | \theta) = \prod_{i=1}^n p(x^{(i)}, y^{(i)})$

$$\log p(D_{1:n} | \theta) = \log \prod_{k=1}^n p(x^{(k)}, y^{(k)}) = \sum_{k=1}^n \log p(x^{(k)}, y^{(k)}) = \sum_{k=1}^n \log p(x^{(k)} | y^{(k)}) + \log p(y^{(k)})$$

- We need to find estimates for  $p(y)$
- And for class conditional posteriors  $p(x_i | y)$
- That MAXIMIZE the likelihood

# Estimation Cont.

$c(x, y)$  = # of docs of class  $y$  that  $x$  occurs in

$c'(x, y)$  = # of times  $x$  occurs across documents of class  $y$

- Bernoulli ML estimate  $p(x_i | y) = \frac{c(x_i, y)}{c(y)}$

- Multinomial ML estimate  $p(x_i | y) = \frac{c'(x_i, y)}{\sum_j c'(x_j, y)}$

- Class prior ML estimate  $p(y) = \frac{c(y)}{\sum_{y'} c(y')}$

# Smoothing

- Estimates can be problematic with small amounts of data
- Other estimates can be more reliable
- Laplace smoothing 
$$p(x_i | y) = \frac{c(x_i, y) + 1}{c(y) + 2}$$

- Generalized Laplace smoothing 
$$p(x_i = v_j | y) = \frac{c(x_i, v_j, y) + 1}{c(y) + s_i}$$

- Where  $s_i = | \text{Val}(x_i) |$

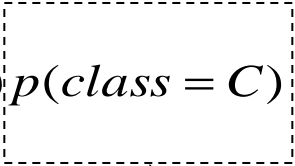
# Document Classification with NB

$$p(\text{class} = C \mid w_{\text{Brown}} = 1, w_{\text{finance}} = 1, w_{\text{spending}} = 1, w_{\text{Treasury}} = 1, \dots)$$

Is proportional to:

$$p(w_{\text{Brown}} = 1, w_{\text{finance}} = 1, w_{\text{spending}} = 1, w_{\text{Treasury}} = 1 \mid \text{class} = C) p(\text{class} = C)$$

$$p(w_{\text{Brown}} = 1, w_{\text{finance}} = 1, w_{\text{spending}} = 1, w_{\text{Treasury}} = 1, \dots \mid \text{class} = C) = \\ p(w_{\text{Brown}} = 1 \mid \text{class} = C) p(w_{\text{finance}} = 1 \mid \text{class} = C) p(w_{\text{spending}} = 1 \mid \text{class} = C) \dots$$



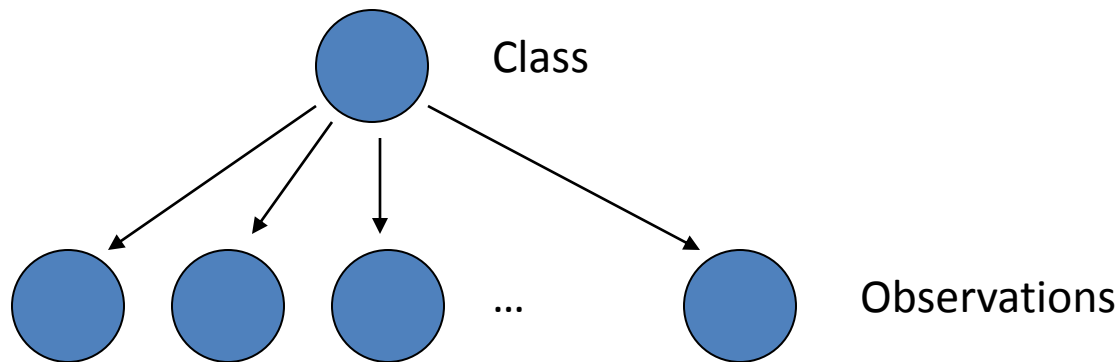
Class prior probability is just the frequency of the class in the training data.

Note that the model assumes each word in a document is independent, *given the class* of the document.

Clearly, this assumption is wrong. However, the classifier still performs well in practice.

# Preview of Graphical Models

- Naïve Bayes is a simple model
- Strong conditional independence assumptions



- Graphical models allow us to determine/specify conditional independence assumptions
- Facilitate development of **algorithms** for learning and inference

# Motivation for Conditional Model

- Strong independence assumptions in NB
- Results in poorly calibrated posterior probabilities

- Also, NB is **generative**

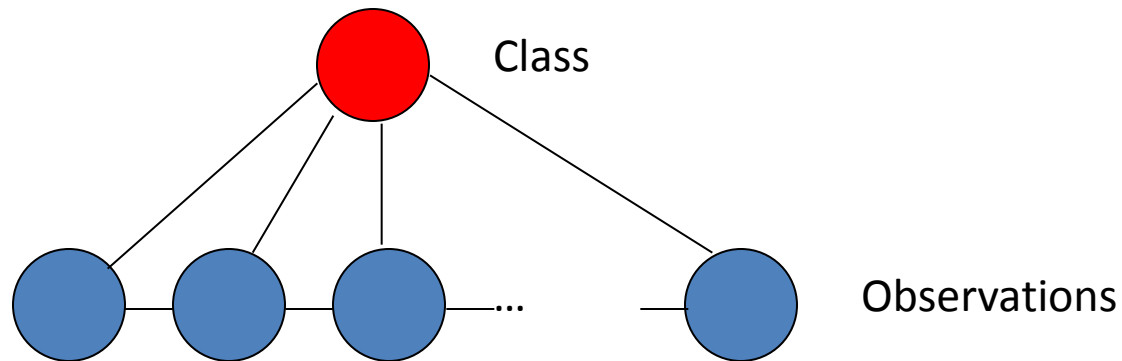
- It models the joint distribution  $p(y | x_1, \dots, x_n) = \frac{p(y, x_1, \dots, x_n)}{p(x_1, \dots, x_n)}$
- It can generate the observed data (e.g. given a class)
- AND make predictions about the class given the data
- We usually only care about making predictions
- Modeling “power” is used to properly generate the data

# A Conditional Model

- Instead of modeling joint distribution
- Model only conditional directly

$$p(y | x_1, \dots, x_n) = \frac{1}{Z} F(x_1, \dots, x_n, y)$$

- This means we can't generate the data
- Model is weaker
- BUT – training it means we need not worry about independence or lack thereof among the observed variables



# Why Maximum Entropy?

- Strong mathematical foundations
- Provides *probabilities* over outcomes
- Is a conditional, discriminative model and allows for mutually dependent variables
- Scales extremely well
  - Training with millions of features and data points
  - Decoding/prediction very fast
- Lots of state-of-the-art results for NLP problems
  - Tagging, parsing, co-reference, parse re-ranking, semantic role labeling, sentiment analysis, etc.
- Forms the core of more complicated, *structured* classification models
  - CRFs, MEMMs, etc.

# Entropy

- X: discrete RV,  $p(X)$
- Entropy (or self-information)

$$H(p) = H(X) = - \sum_{x \in X} p(x) \log_2 p(x)$$

- Entropy measures the amount of information in a RV; it's the average length of the message needed to transmit an outcome of that variable using the optimal code

# Entropy (cont)

$$\begin{aligned} H(X) &= -\sum_{x \in X} p(x) \log_2 p(x) \\ &= \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)} \\ &= E \left( \log_2 \frac{1}{p(x)} \right) \end{aligned}$$

$$H(X) \geq 0$$

$$H(X) = 0 \Leftrightarrow p(X) = 1$$

i.e when the value of  $X$  is determinate, hence providing no new information

# Joint Entropy

- The joint entropy of 2 RV  $X, Y$  is the amount of the information needed on average to specify both their values

$$H(X, Y) = - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(x, y)$$

# Conditional Entropy

- The conditional entropy of a RV  $Y$  given another  $X$ , expresses how much extra information one still needs to supply on average to communicate  $Y$  given that the other party knows  $X$

$$\begin{aligned} H(Y | X) &= \sum_{x \in X} p(x) H(Y | X = x) \\ &= - \sum_{x \in X} p(x) \sum_{y \in Y} p(y | x) \log_2 p(y | x) \\ &= - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y | x) = - E \left[ \log_2 p(Y | X) \right] \end{aligned}$$

# Chain Rule

$$H(X, Y) = H(X) + H(Y | X)$$

$$H(X_1, \dots, X_n) = H(X_1) + H(X_2 | X_1) + \dots + H(X_n | X_1, \dots, X_{n-1})$$

# Mutual Information

$$H(X, Y) = H(X) + H(Y | X) = H(Y) + H(X | Y)$$

$$H(X) - H(X | Y) = H(Y) - H(Y | X) = I(X, Y)$$

- $I(X, Y)$  is the mutual information between  $X$  and  $Y$ . It is the reduction of uncertainty of one RV due to knowing about the other, or the amount of information one RV contains about the other

# Mutual Information (cont)

$$I(X, Y) = H(X) - H(X | Y) = H(Y) - H(Y | X)$$

- I is 0 only when X, Y are independent:  $H(X|Y)=H(X)$
- $H(X)=H(X)-H(X|X)=I(X,X)$  Entropy is the self-information

# Entropy and Linguistics

- Entropy is a measure of uncertainty. The more we know about something, the lower the entropy.
- If a language model captures more of the structure of the language, then the entropy should be lower.
- We can use entropy as a measure of the quality of our models

# Entropy and Linguistics

$$H(p) = H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

- H: entropy of language; we don't know  $p(X)$ ; so..?
- Suppose our model of the language is  $q(X)$
- How good estimate of  $p(X)$  is  $q(X)$ ?

# Entropy and Linguistics

## Kullback-Leibler Divergence

- Relative entropy or KL (Kullback-Leibler) divergence

$$D(p \parallel q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$
$$= E_p \left( \log \frac{p(X)}{q(X)} \right)$$

# Entropy and Linguistics

- Measure of how different two probability distributions are
- Average number of bits that are wasted by encoding events from a distribution  $p$  with a code based on a not-quite right distribution  $q$
- Goal: minimize relative entropy  $D(p || q)$  to have a probabilistic model as accurate as possible

# Maximum Entropy: Intuition

- First, consider the *joint* distribution:
  - {likesCourse x background} x {doesWell}
  - $P(\text{likesCourse}, \text{background}, \text{doesWell})$
- Given no information about this distribution what should we assume?

likesCourse	Background	doesWell	
Y	Y	Y	0.125
Y	Y	N	0.125
Y	N	Y	0.125
Y	N	N	0.125
N	Y	Y	0.125
N	Y	N	0.125
N	N	Y	0.125
N	N	N	0.125

# Maximum Entropy: Intuition

- What if we examine data and see that Jane does well and likes the course 70% of the time?

likesCourse	Background	doesWell	
<b>Y</b>	<b>Y</b>	<b>Y</b>	<b>0.35</b>
Y	Y	N	0.05
<b>Y</b>	<b>N</b>	<b>Y</b>	<b>0.35</b>
Y	N	N	0.05
N	Y	Y	0.05
N	Y	N	0.05
N	N	Y	0.05
N	N	N	0.05

# What is Entropy?

- Measures uncertainty in a distribution

$$H(X, Y) = - \sum_{x, y} p(x, y) \log p(x, y)$$

- For a fixed value of  $x$ , we have:

$$H(Y | X = x) = - \sum_y p(y | x) \log p(y | x)$$

- Conditional entropy:  $H(Y | X) = - \sum_{x, y} \tilde{p}(x) p(y | x) \log p(y | x)$

- Goal: select a distribution  $p$  from a set of allowed distributions that maximizes  $H(Y | X)$

$$p^* = \arg \max_{p \in \mathcal{P}} H(Y | X)$$

# Maximum Entropy Model

- Such a model can be shown to have the following form:

$$p(y | x) = \frac{\exp(\sum_k \lambda_k f_k(x, y))}{\sum_z \exp(\sum_k \lambda_k f_k(x, z))}$$

Where the  $\lambda_k$  are the model parameters and the  $f_k$  are the features of the model.

# Constraints: Empirical Expectations

- We want the most uniform distribution subject to some constraints
  - Constraints we see in some example data
- Constraints operate over features
- Defined as:  $\tilde{E}[f_k] = \sum_{x,y} \tilde{p}(x,y) f_k(x,y)$        $f_{likesCourse,doesWell}(x,y) = \{1,0\}$
- E.g. if Jane has taken 100 courses in the past, and she did well in 50 of them, and of those 50 in 35 she liked the material. In the 50 she didn't do well, she liked the material in 5 courses

$$E[f_{likesCourse,doesWell}(x,y)] = .35$$

$$E[f_{likesCourse,doesNOTdoWell}(x,y)] = .05$$

# Model Expectations

- Feature *expectations*, according to a model are defined:

$$E[f_k] = \sum_{x,y} \tilde{p}(x) p(y|x) f(x,y)$$

- Goal:

$$p^* = \arg \max_{p \in \mathcal{P}} H(Y | X)$$

such that  $E[f_k] = \tilde{E}[f_k]$  for all  $k$

i.e.  $\sum_{x,y} \tilde{p}(x) p(y|x) f_k(x,y) = \sum_{x,y} p(x,y) f_k(x,y)$

# Lagrange Multipliers (\* Optional slide)

- General method for finding function optima given equality constraints

$$\Lambda(x, \lambda) = f(x) + \sum_k \lambda_k g_k(x) \quad g_k(x) = 0$$

- For our problem:

$$\begin{aligned} \Lambda(p, \lambda) = & -\sum_{x,y} \tilde{p}(x) p(y|x) \log p(y|x) + \lambda_0 \left( \sum_y p(y|x) - 1 \right) \\ & + \sum_k \lambda_k \left( \sum \tilde{p}(x, y) f_k(x, y) - \tilde{p}(x) p(y|x) f_k(x, y) \right) \end{aligned}$$

# Derivation of Max Entropy (\* Optional Slide)

$$\frac{\partial \Lambda(x, \lambda)}{\partial p(y | x)} = \tilde{p}(x)(1 + \log p(y | x)) + \lambda_0 - \left( \sum_k \lambda_k \tilde{p}(x) f_k(x, y) \right)$$

Set this to zero and solve:

$$\tilde{p}(x) + \tilde{p}(x) \log p(y | x) + \lambda_0 - \left( \sum_k \lambda_k \tilde{p}(x) f_k(x, y) \right) = 0$$

$$\log p(y | x) = \sum_k \lambda_k f_k(x, y) - \frac{\lambda_0}{\tilde{p}(x)} - 1$$

$$p(y | x) = \exp\left( \sum_k \lambda_k f_k(x, y) \right) \exp\left( -\frac{\lambda_0}{\tilde{p}(x)} - 1 \right)$$

We know that  $\lambda_0$  is the multiplier over the constraint that requires the distribution sum to 1, therefore it corresponds to a normalizing constant:

$$p(y | x) = \frac{\exp\left( \sum_k \lambda_k f_k(x, y) \right)}{\sum_z \exp\left( \sum_k \lambda_k f_k(x, z) \right)}$$

# Maximum Likelihood Training

- Given a set of training data, we would like to find a set of model parameters that best explain the data – a set of parameters that make the data most likely:
- Example:
  - You observe an (unfair) coin flipped 100 times. It turns up heads 60 times. The possible ‘parameters’ for the coin are:  $p(\text{HEADS}) = 1/3$ ,  $p(\text{HEADS}) = 1/2$ ,  $p(\text{HEADS}) = 2/3$
  - Which coin was most likely used?
- For prediction tasks using a conditional probability model (not just MaxEnt), this is formulated as:

$$\arg \max_{\Lambda} L_D(p_{\Lambda}) = \log \prod_{i=1}^{|D|} p_{\Lambda}(y^{(i)} | x^{(i)}) = \sum_{i=1}^{|D|} \log p_{\Lambda}(y^{(i)} | x^{(i)})$$

# Maximum Likelihood

$$\arg \max_{\Lambda} L_D(p_{\Lambda}) = \log \prod_{i=1}^{|D|} p_{\Lambda}(y^{(i)} | x^{(i)}) = \sum_{i=1}^{|D|} \log p_{\Lambda}(y^{(i)} | x^{(i)})$$

$$= \sum_{i=1}^{|D|} \log \frac{\exp \sum_k \lambda_k f_k(x^{(i)}, y^{(i)})}{\sum_z \exp \sum_k \lambda_k f_k(x^{(i)}, z^{(i)})}$$

$$= \sum_{i=1}^{|D|} \sum_k \lambda_k f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{|D|} \log \sum_z \exp \sum_k \lambda_k f_k(x^{(i)}, z^{(i)})$$

This function turns out to be convex with a single global maximum.  
How do we maximize such a function?

# Gradient of the Log-Likelihood

We take the partial derivative with respect to each parameter,  $\lambda_k$

$$\begin{aligned}\frac{\partial L_D(p_\Lambda)}{\partial \lambda_k} &= \sum_{i=1}^{|D|} f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{|D|} \sum_z \frac{f_k(x, z) \exp \sum_k \lambda_k f_k(x, z)}{\sum_{z'} \exp \sum_k \lambda_k f_k(x, z')} \\ &= \sum_{i=1}^{|D|} f_k(x, y) - \sum_{i=1}^{|D|} \sum_z p_\Lambda(z | x) f_k(x, z)\end{aligned}$$

And set to 0

$$\tilde{E}[f_k] - E_\Lambda[f_k] = 0$$

Gradient is just the difference in feature expectations. But, expectation for a particular feature is dependent on ALL the other parameters. No closed form!

# MaxEnt Estimation

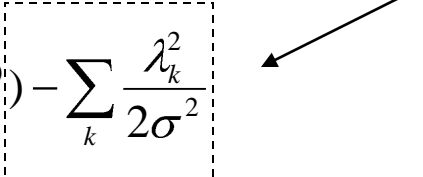
- Contrast with Naïve Bayes
  - No closed form
  - Computationally Expensive
- The expectation for each feature requires knowing the expectations of all the other features
  - We must determine the best parameter values “jointly” over all features
  - This is what allows MaxEnt to gracefully handle features that are not independent and “do the right thing”
  - If two features are completely dependent, they will have the same learned parameter values

# Parameter Estimation

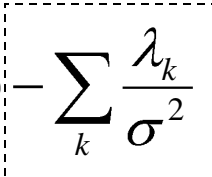
- Use iterative scaling methods
  - Adjust one parameter with all others fixed
- Apply any non-linear numerical optimization method
- Methods divided into:
  - First order methods:
    - Move in direction of steepest ascent
    - Direction a function of steepest direction + last direction
    - Conjugate gradient, Newton's method
  - Second order methods:
    - Consider the curvature of the function – it's second derivative – Hessian matrix
    - Smarter about picking good directions
    - Hessian is too big, methods use an approximate version

# MAP Inference

- Many probabilistic models benefit from *smoothing*, or *regularization*.
  - Biases introduced to prevent the model from fitting the data too closely and to improve generalization
- With Maximum Entropy, smoothing often achieved by introducing a Gaussian prior over the parameters

$$= \sum_{i=1}^{|D|} \sum_k \lambda_k f_k(x^{(i)}, y^{(i)}) - \sum_{i=1}^{|D|} \log \sum_z \exp \sum_k \lambda_k f_k(x^{(i)}, z^{(i)}) - \sum_k \frac{\lambda_k^2}{2\sigma^2}$$


The gradient is also modified accordingly:

$$= \sum_{i=1}^{|D|} f_k(x, y) - \sum_{i=1}^{|D|} \sum_z p_{\Lambda}(z | x) f_k(x, z) - \sum_k \frac{\lambda_k}{\sigma^2}$$


# Other Ways to Estimate Parameters

- Averaged Perceptron

- Repeatedly classify examples in training data
- When mistakes are made with current parameters
  - Update parameter values
- Repeat until convergence

- Stochastic Gradient Descent

- Take a small sample of the training data
- Compute the log-likelihood gradient for just that sample
- Update parameters based on the gradient
- Repeat until convergence

# Averaged Perceptron

- Input: Training examples  $D = \{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\}$
- Initialization:  $\Lambda = [0 \dots 0]$
- For  $t = 1, \dots, T, i = 1, \dots, n$ 
  - Calculate  $y'^{(i)} = \arg \max_y \sum_k \lambda_k f_k(x^{(i)}, y)$
  - If  $y'^{(i)} \neq y^{(i)}$  then  $\lambda_k = \lambda_k + f_k(x^{(i)}, y^{(i)}) - f_k(x^{(i)}, y'^{(i)})$
- Output  $\Lambda$

- Predict using:

$$\arg \max_y \sum_k f_k(x^{(i)}, y) \bullet \text{avg}(\lambda_k^1, \dots, \lambda_k^{Tn})$$

# Doc. Classification using Maximum Entropy

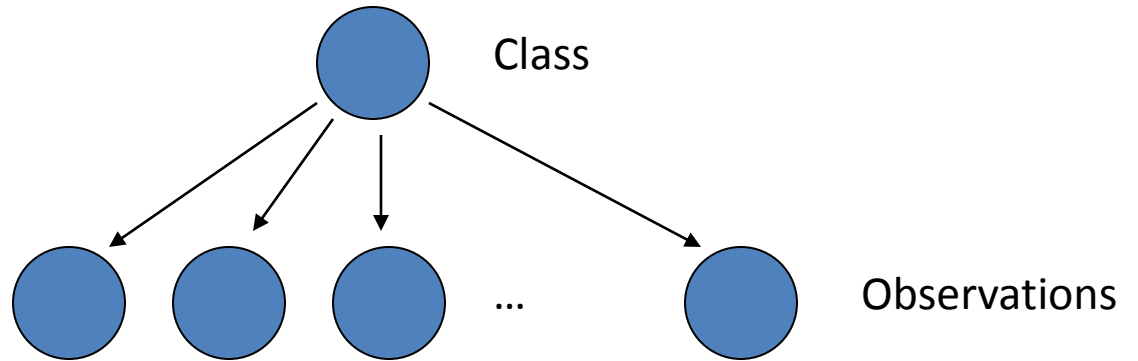
- View given data as the whole document itself (not a vector of words). Each feature queries whether a word is present.

$$p(\text{class} = c \mid \text{document} = d) = \frac{\exp(\lambda_{\text{Brown}}^c f_{\text{Brown}}(d, c) + \lambda_{\text{finance}}^c f_{\text{finance}}(d, c) + \lambda_{\text{spending}}^c f_{\text{spending}}(d, c) + \dots)}{\sum_{c'} \exp(\lambda_{\text{Brown}}^{c'} f_{\text{Brown}}(d, c') + \lambda_{\text{finance}}^{c'} f_{\text{finance}}(d, c') + \lambda_{\text{spending}}^{c'} f_{\text{spending}}(d, c') + \dots)}$$

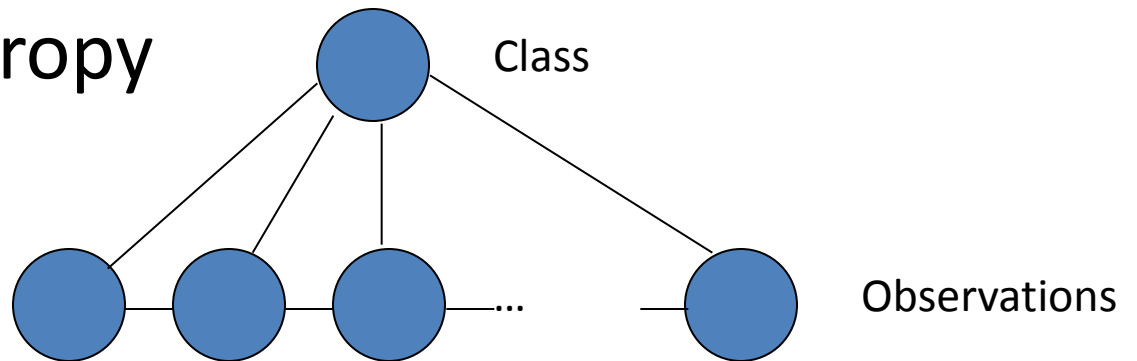
- Feature values can be indicators (0 or 1) or frequencies
- Model handles feature dependencies very well
  - E.g. San Francisco

# Graphical Models

- Naïve Bayes



- Maximum Entropy



# Summary

- Maximum Entropy classifier:
  - Directly estimates the conditional distribution,  $p(y|x)$
  - Learn by maximizing conditional likelihood
  - Allows for interacting, non-independent features
  - Training relatively complex: numerical optimization
- Naïve Bayes
  - Estimates the joint distribution  $p(x,y)$
  - Learn by maximizing joint likelihood
  - Makes strong independence assumptions about features
  - Very easy to train – just count