Columbia University

Statistical NLP for the Web

Maximum Entropy Models

Sameer Maskey

Week 11, Nov 14, 2012

Topics for Today

- Logistic Regression/Maximum Entropy Models

Final Project

- Intermediate Report I Grades sent out
- Intermediate Report II Oral
- Final Project Presentation Day
  - December 12th, 9:30 AM to 2pm
  - Wednesday
  - Each team 12 min talk
  - 3 min for Q&A
  - CS Conference room

Final Project Grading

- Final Project Remaining Grade – 85%
- Intermediate Report II
  - 20% of 55 = 11 points
- Final Project : Report+Presentation+Demo
  - 65% of 55 = 35.75 points
  - Final Project Report (30%)
  - Demo (15%)
  - Final Presentation (20%)

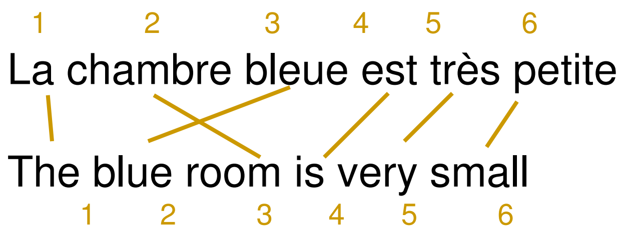
Intermediate Report II

- Grading based on
  - Demo
    - Is it working?
  - Approach
    - Is your approach well thought through
  - Results
    - Is the model accuracy real low?
  - Discussion
    - Have you thought about ways to improve the model
  - Q&A
    - Theory behind algorithms you have used

HW3

- HW3 is out
- Due Nov 30^{th} (11:59pm)
- Q1 : implementing simple example from the class is ok as well
- Q2 : Look at previous slides, Python code is already in one of the slides
  - Modify to do n-gram counts
  - Compute bigram probabilities

Alignments to Phrases



Phrase Table

la || the
chambre bleue || blue room
bleue || blue
est tres petite || is very small
est tres || very small
chambre bleue est tres petite || blue room is very small
la chambre bleue est tres petite || the blue room is very small

- Once we get the alignments we can extract phrase pairs
- Phrase pairs are then used to compute relative frequencies that gives us P(e|f) and P(f|e)

Translation Features

the small house || klein haus
small house || klein haus
very small house || klein haus

How do you compute phrase translation features?

Translation Features

- P(e|f) and P(f|e) can be estimated using Maximum Likelihood Estimate

- $P(e|f) = count(e,f)/count(e)$

- $P(f|e) = count(f,e)/count(e)$

Translation Features

the small house || klein haus
small house || klein haus
very small house || klein haus

0.33

$$P(\text{small house} \mid\mid \text{klein haus}) = \frac{\text{count(klein haus, small house)}}{\text{count(klein haus)}}$$

$= 1/3 = 0.33$

Translation Features

- Besides P(e|f) and P(f|e) we can add many different features in similar framework

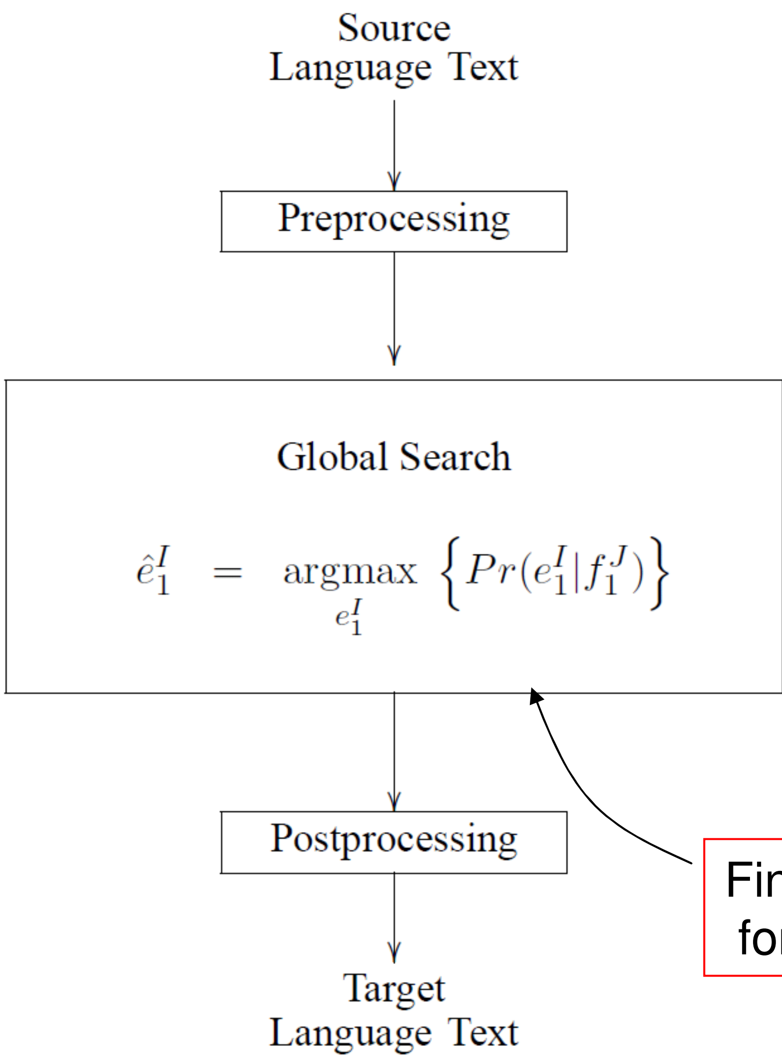
the small house || klein haus
small house || klein haus
very small house || klein haus
the small house || haus

0.33 1 3/2 1 1

$$f_1, f_2, f_3, f_4, f_5$$

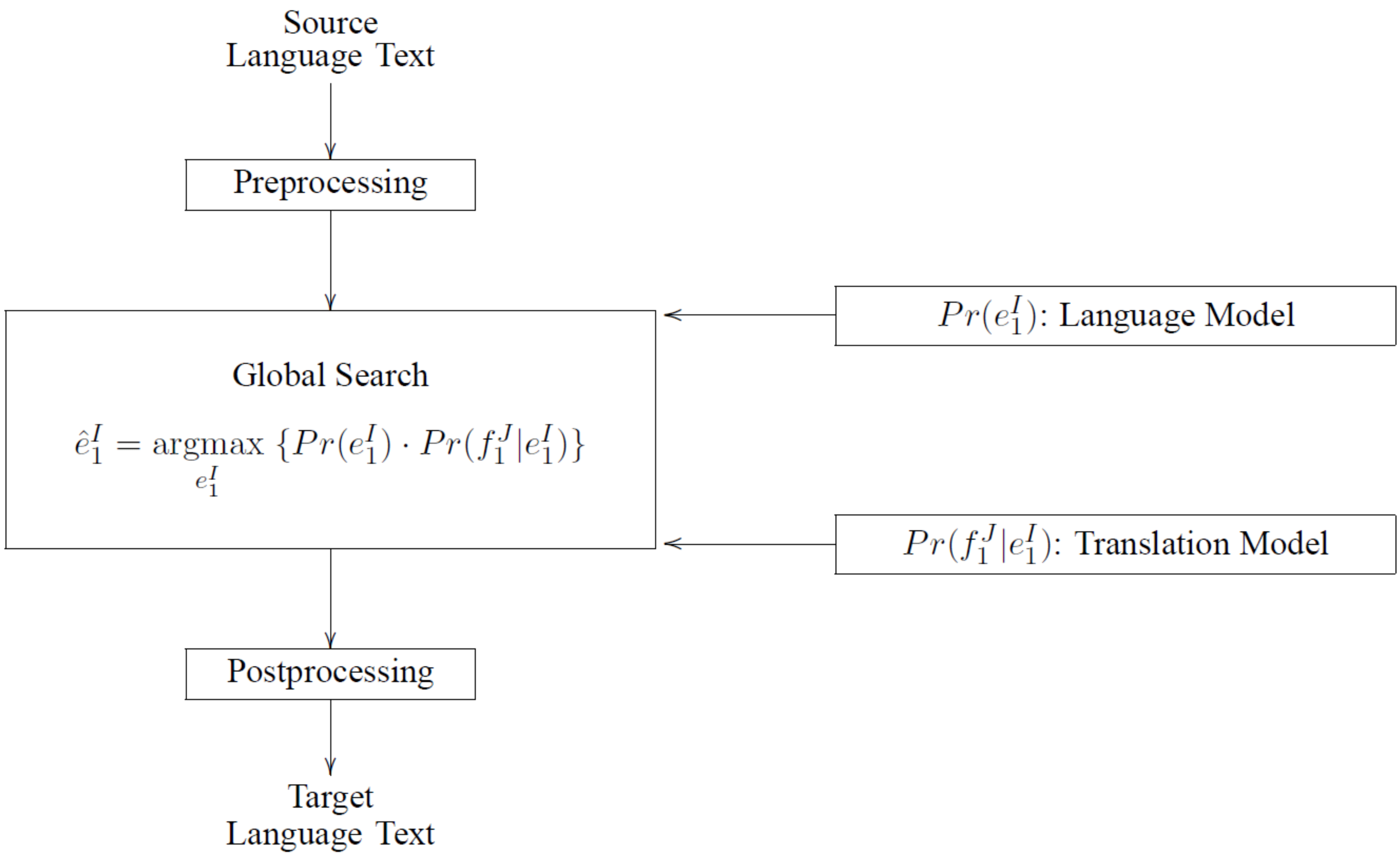
f1 = p(e|f)
f2 = p(f|e)
f3 = len(e)/len(f)
..
f5 = #(noun in e)/ #(noun in f)

Features and Machine Translation

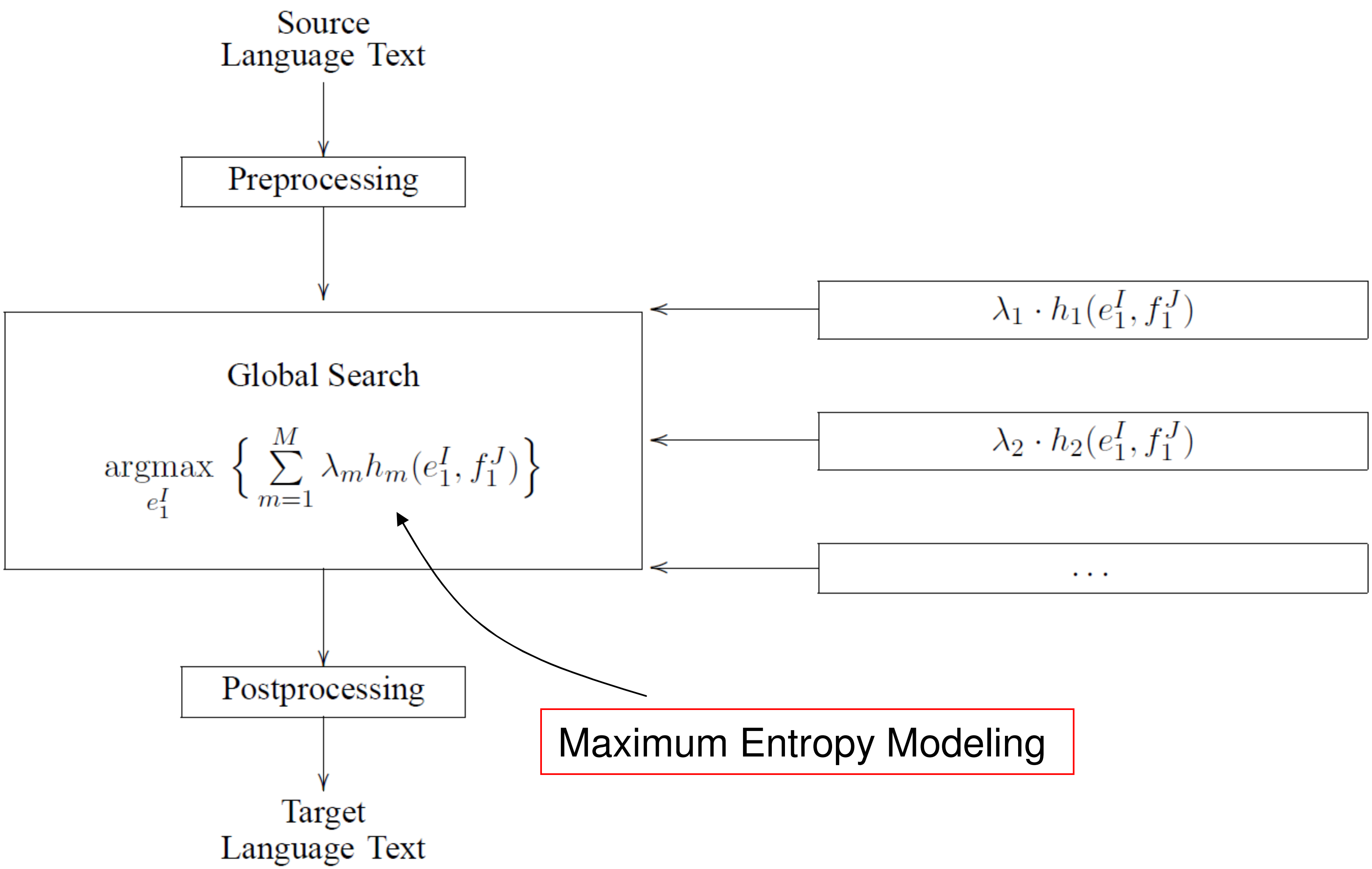


Find me best English translation e(1..I) for foreign sentence f(1..J)

Source Channel Approach



Maximum Entropy Based Machine



Maximum Entropy Method for Translation

Translation probability for foreign word sequence f1 to fj is given by above equation

$$
\begin{aligned}
Pr(e_1^I|f_1^J) &= p_{\lambda_1^M}(e_1^I|f_1^J) \\
&= \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J)]}{\sum_{e'^I_1} \exp[\sum_{m=1}^{M} \lambda_m h_m(e'^I_1, f_1^J)]}
\end{aligned}
$$

We are still find argmax of e(1..I) but not using source channel approach

Log-Linear Model

- If x is any data point and y is the label, general log-linear linear model can be described as follows

Makes Positive

Weight for
Given feature functions

$$p(y|x;w) = \frac{\exp \sum_j w_j F_j(x,y)}{\sum_{y'} \exp \sum_j w_j F_j(x,y')}$$

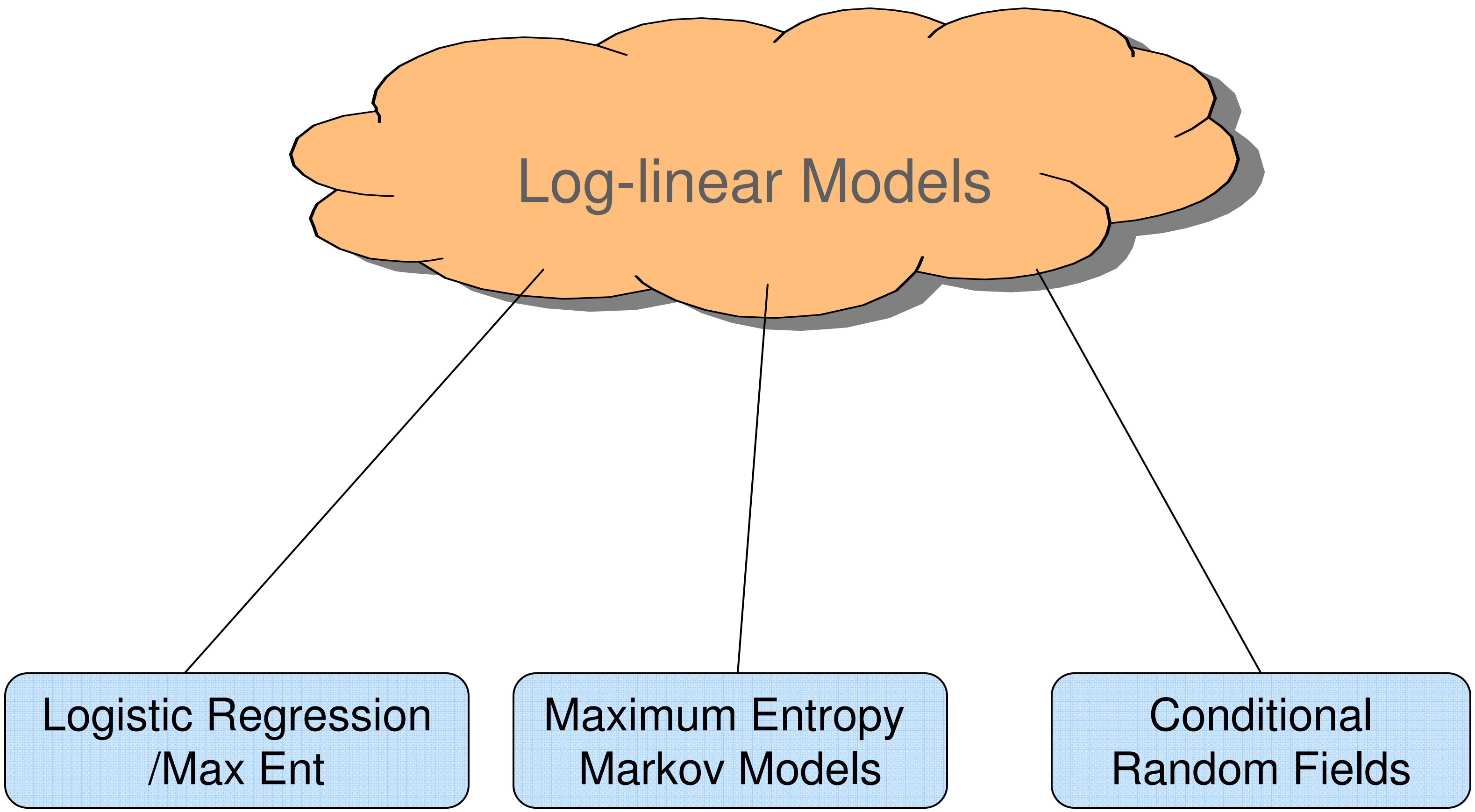
Feature
Functions

Normalization Term
(Partition Function)

Understanding the Equation Form

- Linear combination of features and weights
  - Can be any real value
- Numerator always positive (exponential of any number is +ve)
- Denominator normalizes the output making it valid probability between 0 and 1
- Ranking of output same as ranking of linear values
  - i.e. exponentials magnify the ranking difference but ranking still stay the same

Log-linear Models



All of these models are a type of log-linear models, there are more of them

Direct Translation Probability

Translation probability for foreign word sequence f1 to fj is given by above equation

$$Pr(e_1^I|f_1^J) = p_{\lambda_1^M}(e_1^I|f_1^J)$$
$$= \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J)]}{\sum_{e'^I_1} \exp[\sum_{m=1}^{M} \lambda_m h_m(e'^I_1, f_1^J)]}$$

Look similar?

The above equation is same as a general equation that defines a type of classifiers generally known as log linear models

$$p(y|x;w) = \frac{\exp \sum_j w_j F_j(x,y)}{\sum_{y'} \exp \sum_j w_j F_j(x,y')}$$

Log Linear Model

Training Weights for Features

- How do we know the values of those lambdas in previous equation
- We train them in Maximum Entropy Framework

Maximum Entropy Model

- Maximum Entropy Models are a type of log-linear models
- Maximum Entropy Model has shown to perform well in many NLP tasks
  - POS tagging [Ratnaparkhi, A., 1996]
  - Text Categorization [Nigam, K., et. al, 1999]
  - Named Entity Detection [Borthwick, A, 1999]
  - Parser [Charniak, E., 2000]
- Discriminative classifier
  - Conditional model P(c|d)
  - Maximize conditional likelihood
  - Can handle variety of features

Naïve Bayes vs. Maximum Entropy Models

Naïve Bayes Model	Maximum Entropy Model
Trained by maximizing likelihood of data and class	Trained by maximizing conditional likelihood of classes
Features are assumed independent	Dependency on features taken account by feature weights
Feature weights set independently	Feature weights are set mutually

Entropy

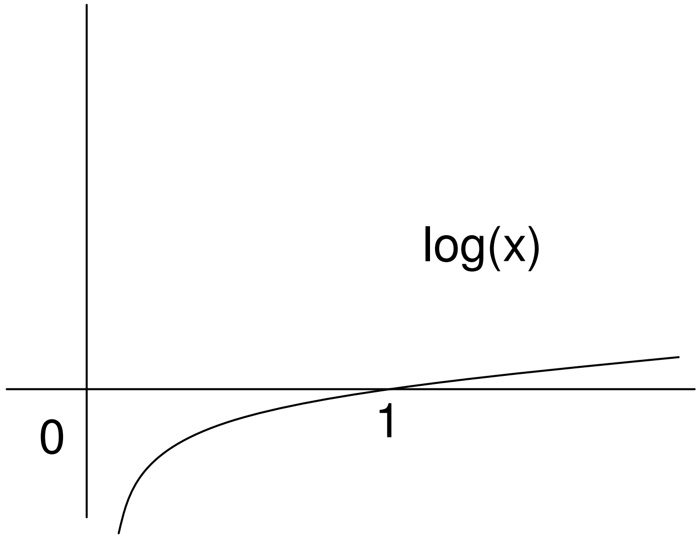
Event x

Probability	p_x
Surprise	$log(1/p_x)$

$$H(p) = -\sum_x p(x) log_2\, p(x)$$

- Measure of uncertainty of a distribution
- Higher uncertainty equals higher entropy
- Degree of surprise of an event
- If I see something that is highly unlikely (very low p(x) e.g. event that doesn't happen a lot) then that carries lot more information so lower entropy

Entropy



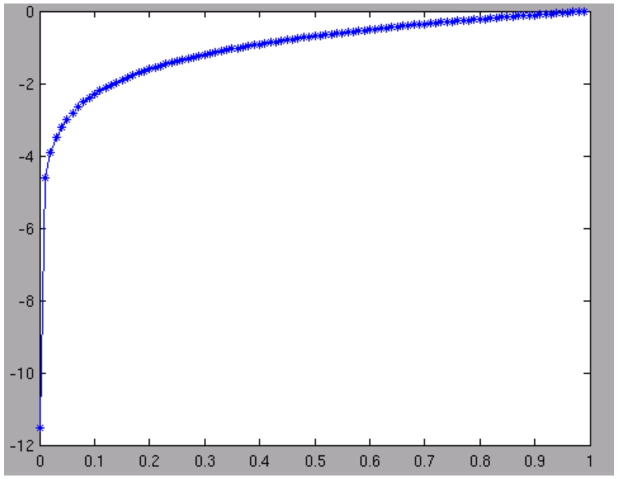
- p(x) log p(x) → 0 as p(x)→0
- p(x) log p(x) → 0 as p(x)→ 1

Exploring Entropy Formulation

- How much information received when observing a random variable ‘x’ ?
- Highly improbable event = received more information
- Highly probable event = received less information

- Need h(x) that express information content of p(x); we want
  1. Monotonic function of p(x)
  2. If p(x,y) = p(x). p(y) when x and y are unrelated, i.e. statistically independent then we want h(x,y) = h(x) + h(y) such that information gain by observing two <u>unrelated</u> events is their sum

Exploring Entropy Formulation (cont.)



Remember logarithm function

- What kind of h(x) satisfies two conditions mentioned previously

$$\boxed{h(x) = -\log_2 p(x)}$$

Entropy Example

- If X ={1, 2, 3} and p = (1/2, ¼, ¼)

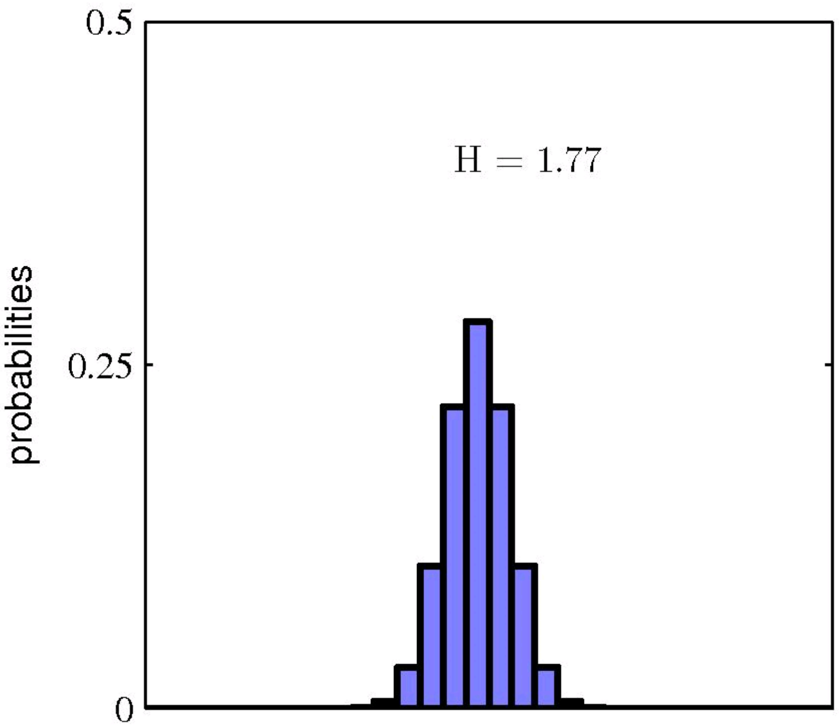
$$H(p) = -\sum_x p(x) log_2\, p(x)$$

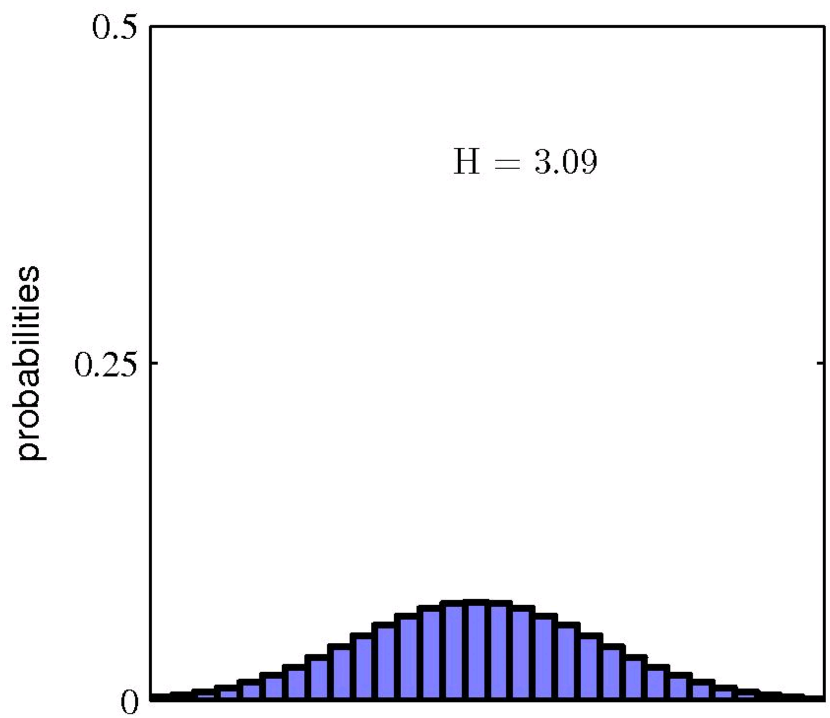
$$= -\frac{1}{2} log \frac{1}{2} - \frac{1}{4} log \frac{1}{4} - \frac{1}{4} log \frac{1}{4}$$

$$= \frac{1}{2} 1 + \frac{1}{4} 2 + \frac{1}{4} 2$$

$$= 3/2$$

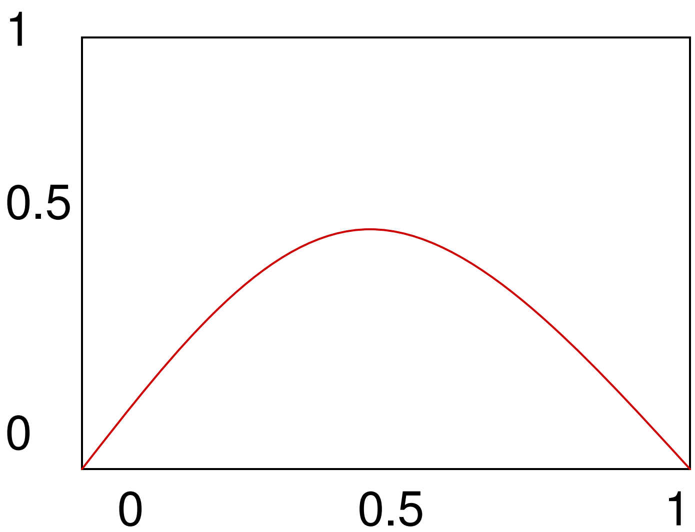
Comparing Entropy Across Distributions

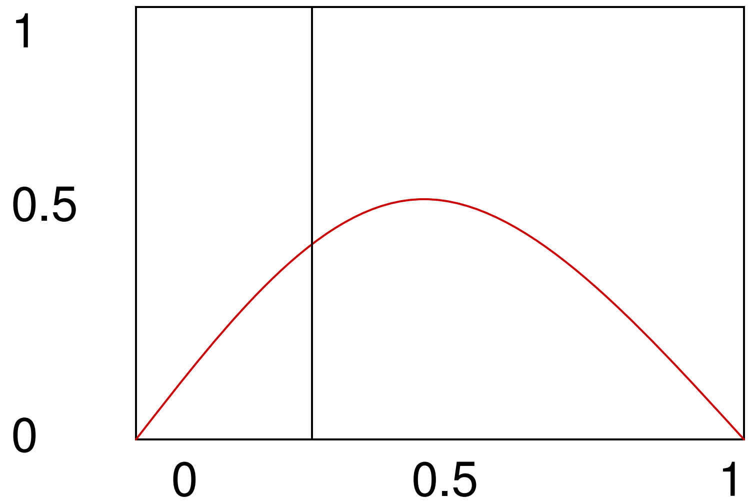




[1] Uniform distribution has higher entropy

Maximizing Entropy





Constraint P(head) = 0.3

- Maximizing Entropy subject to constraints
  - Lowers maximum entropy of the distribution
  - Raises maximum likelihood
  - Brings distribution is further away from uniform distribution
  - Brings distribution is closer to the data

Maximizing Entropy

- How can we find a distribution with maximum entropy?
- What about maximizing entropy of a distribution with a set of constraints?
- What does maximizing entropy has to do with classification task anyway?

- Let us first look at logistic regression to understand this

Remember Linear Regression

$$y_j = \theta_0 + \theta_1 x_j$$

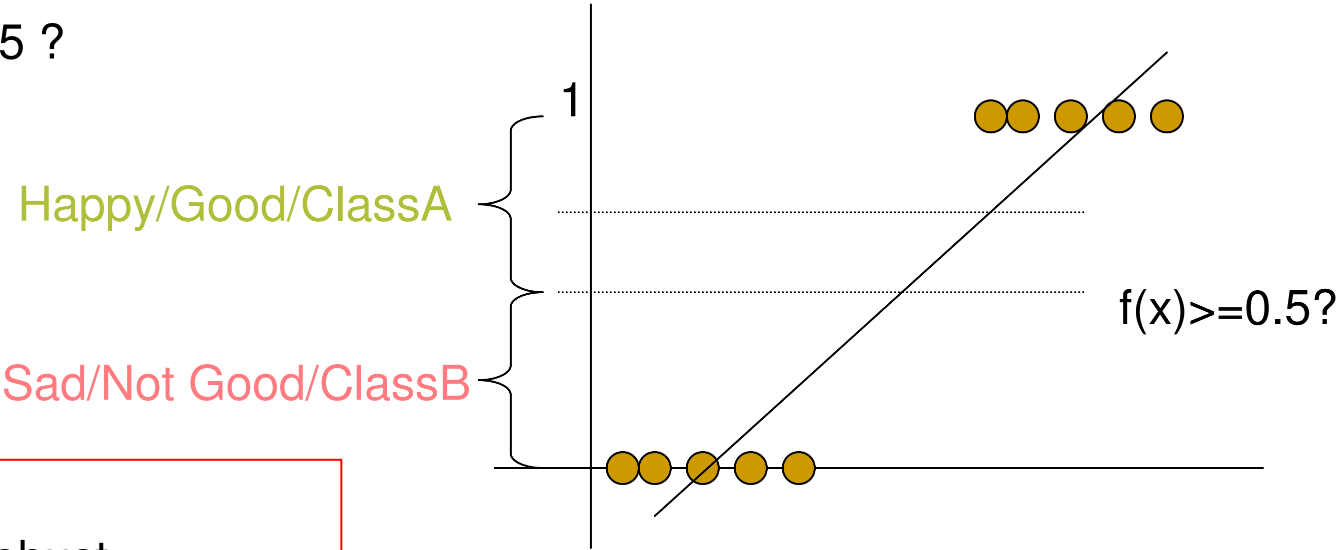
$$y_j = \sum_{i=0}^{N} \theta_i x_{ij}$$

N is the number of dimensions where each input lives in

- We estimated theta by setting square loss function's derivative to zero

Regression to Classification

- We also looked at why linear regression may not work well if ‘y’ are binary
  - Output (-infinity to +infinity) is not limited to class labels (0 and 1)
  - Assumption of noise (errors) normally distributed
- Train Regression and threshold the output
  - If f(x) >= 0.7 CLASS1
  - If f(x) < 0.7 CLASS2
  - f(x) >= 0.5 ?



Problems :
Not robust
output –inf to +ve inf

Ratio

- Instead of thresholding the output we can take the ratio of two predicted output

- Ratio is odds of predicting y=1 or y=0
  - E.g. for given 'x' if p(y=1) = 0.8 and p(y=0) = 0.2
  - Odds = 0.8/0.2 = 4

Not predicted y, predicting odds of y

- Better?
  - We can make the linear model predict odds of y=1 instead of 'y' itself

$$\frac{p(y=true|\mathbf{x})}{p(y=false|\mathbf{x})} = \sum_{i=0}^{N} \theta_i \; x_i$$

Log Ratio

$$\frac{p(y=true|\mathbf{x})}{p(y=false|\mathbf{x})} = \sum_{i=0}^{N} \theta_i \, x_i$$

- LHS Range?
  - LHS is between 0 and infinity, we want to be able to handle –infinity to +infinity which RHS can produce
- If we take log of LHS, it can also range between –infinity and +ve infinity

$$log\left(\frac{p(y=true|\mathbf{x})}{p(y=false|\mathbf{x})}\right)$$

$$log\left(\frac{p(y=true|\mathbf{x})}{(1-p(y=true|\mathbf{x})}\right)$$

$$\boxed{logit(p(x)) = log\frac{p(x)}{1-p(x)}}$$

Logistic Regression

$$log(\frac{p(y=true|\mathbf{x})}{(1-p(y=true|\mathbf{x})})=\sum_{i=0}^{N}\theta_i \times x_i$$

- Logistic Regression: A Linear Model in which we predict logit of probability instead of probability

$$log(\frac{p(y=true|\mathbf{x})}{(1-p(y=true|\mathbf{x})})=w \cdot f$$

Logistic Regression Derivation

$$log(\frac{p(y=true|\mathbf{x})}{(1-p(y=true|\mathbf{x})}) = w \cdot f$$

$$\frac{p(y=true|\mathbf{x})}{(1-p(y=true|\mathbf{x}))} = exp(w \cdot f)$$

$$p(y = true|\mathbf{x}) = (1 - p(y = true|\mathbf{x})exp(w \cdot f)$$

$$p(y = true|\mathbf{x}) = exp(w \cdot f) - p(y = true|\mathbf{x})exp(w \cdot f)$$

$$p(y = true|\mathbf{x}) + p(y = true|\mathbf{x})exp(w \cdot f) = exp(w \cdot f)$$

$$p(y = true|\mathbf{x}) = \frac{exp(w \cdot f)}{1+exp(w \cdot f)}$$

Logistic Regression

$$p(y = true|\mathbf{x}) = \frac{exp(\sum_{i=0}^{N} \theta_i x_i)}{1+exp(\sum_{i=0}^{N} \theta_i x_i)}$$

$$p(y = false|\mathbf{x}) = \frac{1}{1+exp(\sum_{i=0}^{N} \theta_i x_i)}$$

For notation convenience for later part of the lecture replace theta with lambda and x with f where f is an indicator function

$$p(y = true|\mathbf{x}) = \frac{exp(\sum_{i=0}^{N} \lambda_i f_i)}{1+exp(\sum_{i=0}^{N} \lambda_i f_i)}$$

$$p(y = false|\mathbf{x}) = \frac{1}{1+exp(\sum_{i=0}^{N} \lambda_i f_i)}$$

Logistic Regression

Dividing Numerators and denominator by $exp(-\sum_{i=0}^{N} \lambda_i f_i)$

$$p(y = true|\mathbf{x}) = \frac{1}{1+exp(-\sum_{i=0}^{N} \lambda_i f_i)}$$

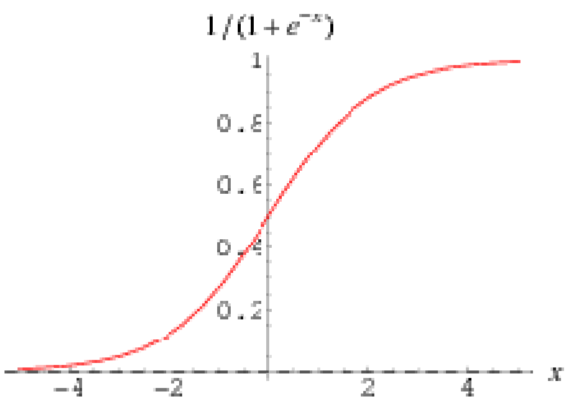
$$p(y = false|\mathbf{x}) = \frac{exp(-\sum_{i=0}^{N} \lambda_i f_i)}{1+exp(-\sum_{i=0}^{N} \lambda_i f_i)}$$

Logistic Regression

Let's call $\mathrm{h}_\lambda(x)$ represent p(y=true|x)

$$p(y = true|\mathbf{x}) = \frac{1}{1+exp(-\sum_{i=0}^{N} \lambda_i f_i)}$$

We have seen this before, remember?



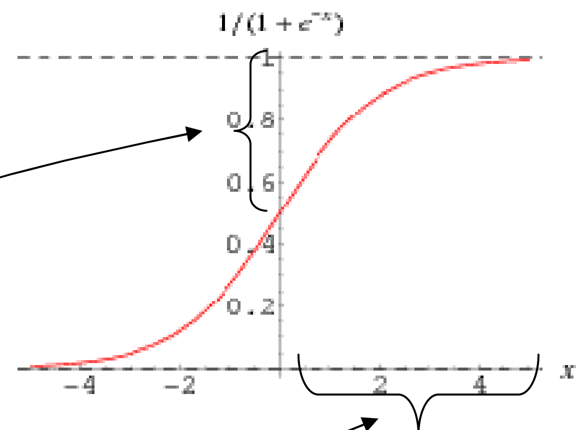
$$g(x) = \frac{1}{1+e^{-\beta x}}$$

Sigmoid functions we used for our Neural Networks!
Also known as logistic function, thus the name logistic regression

Decision Boundary of Logistic Regression

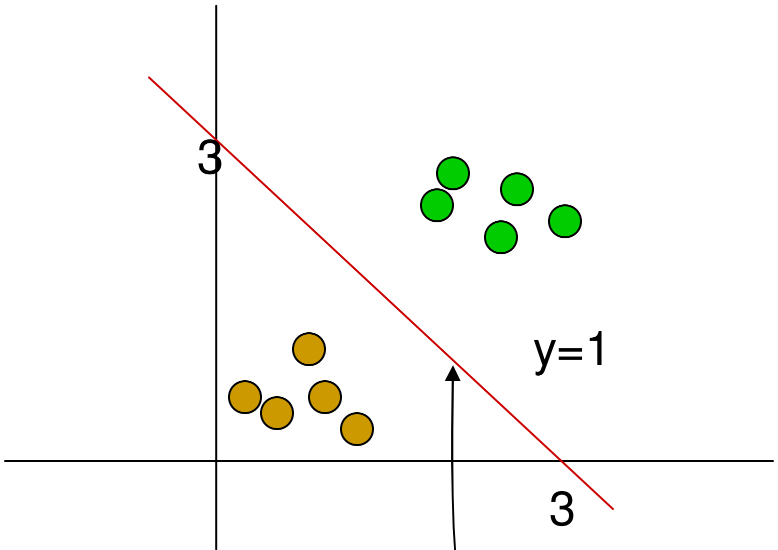
If we predict $y = 1$ when $h_\lambda(x) \geq 0.5$

i.e. when $\lambda^T f \geq 0$



This is saying that we will predict y=1 when logit function outputs more than 0.5
which happens when the linear combination of weights and features is greater than zero

Example of Decision Boundary



$$h_\lambda(x) = \lambda_0 + \lambda_1 x_1 + \lambda_2 x_2$$

Assume we get lambdas to be [-3 1 1]

$$\text{Y=1 if } -3 + x1 + x2 \geq 0$$

$$x_1 + x_2 \geq 3$$

$$h_\lambda(x) \geq 0.5$$

Logistic Regression Based Classification

- If we estimate the weights (Lambdas) we can classify between 2 classes
- How to estimate weights (Lambdas)
- We can estimate weights by maximizing (conditional) likelihood of data according to the model

So why did we talk all about logistic regression when we were trying to learn Maximum Entropy Models?

Let's find out

Logistic Regression for Multiple Classes

- We can also have logistic regression for multiple classes
- Normalization has to take account of all classes

$$p(c|\mathbf{x}) = \frac{exp(\sum_{i=0}^{N} \lambda_{ci} f_i)}{\sum_{c' \in C} exp(\sum_{i=0}^{N} \lambda_{c' i} f_i)}$$

Logistic Regression for Multiple Classes

- We can also have logistic regression for multiple classes
- Normalization has to take account of all classes

$$p(c|\mathbf{x}) = \frac{exp(\sum_{i=0}^{N} \lambda_{ci} f_i)}{\sum_{c' \in C} exp(\sum_{i=0}^{N} \lambda_{c'i} f_i)}$$

This equation looks familiar?

$$\begin{aligned} Pr(e_1^I|f_1^J) &= p_{\lambda_1^M}(e_1^I|f_1^J) \\ &= \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J)]}{\sum_{e'^I_1} \exp[\sum_{m=1}^{M} \lambda_m h_m(e'^I_1, f_1^J)]} \end{aligned}$$

Maximum Entropy and Logistic Regression

"Exponential Model for Multinomial Logistic Regression, when trained according to the maximum likelihood criterion, also finds the Maximum Entropy Distribution subject to the constraints from the feature function" [2]

Turns out logistic regression models also finds maximum entropy distribution!

Multinomial Logistic Regression is also known as Maximum Entropy Model in NLP and Speech

Why Maximum Entropy for NLP?

- Maximum Entropy Modeling technique is particularly very useful for NLP problems where we want to extract all sorts of features
- Distribution can be spiked for certain features for which we have more information
- Assume nothing for features we have not seen
- What kind of features?

Maximum Entropy Features for NLP Problems

- We have seen many different types of features
  - Count of words, length of docs, etc
- We can think of features as indicator functions that represent co-occurrence relation between input phenomenon and the class we are trying to predict

$$f_i(c_d) = \phi(d) \wedge c_d = c_i$$

Example: Features for POS Tagging

- $f_1(c,d) = \{ c{=}NN \wedge curword(d){=}book \wedge prevword(d){=}to\}$
- $f_2(c,d) = \{ c{=}VB \wedge curword(d){=}book \wedge prevword(d){=}to\}$
- $f_3(c,d) = \{ c{=}VB \wedge curword(d){=}book \wedge \wedge prevClass(d){=}ADJ\}$

Maximum Entropy Example

Given Event space

NN	JJ	NNS	VB

Maximum Entropy Distribution

1/4	1/4	1/4	1/4

Add a constraint P(NN) + P(JJ) + P(NNS) = 1

1/3	1/3	1/3	0

Add another constraint P(NN) + P(NNS) = 8/10

4/10	2/10	4/10	0

Another Example

$$f_1(c,x) = \begin{cases} 1 & \text{if } word_i = \text{“race”} \;\&\; c = \text{NN} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(c,x) = \begin{cases} 1 & \text{if } t_{i-1} = \text{TO} \;\&\; c = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

		f1	f2	f3	f4	f5	f6
VB	f	0	1	0	1	1	0
VB	w		.8		.01	.1	
NN	f	1	0	0	0	0	1
NN	w	.8					-1.3

$$P(NN|x) = \frac{e^{.8}e^{-1.3}}{e^{.8}e^{-1.3} + e^{.8}e^{.01}e^{.1}} = .20$$

$$P(VB|x) = \frac{e^{.8}e^{.01}e^{.1}}{e^{.8}e^{-1.3} + e^{.8}e^{.01}e^{.1}} = .80$$

Example from [2]

Training MaxEnt

- We saw that given the features and weights we just need to plug them in our equation and we get classification probability
- Previous example:
  - Given “to race” our model correctly predicted race is Verb with 0.8 probability
- But how do we train these weights?

Training Weights for MaxEnt Model

- Joint Generative Models
  - P(c,d) – Naïve Bayes
  - Maximize joint likelihood
  - Maximum Likelihood Estimation – Relative Frequencies
- Discriminative Models
  - P(c|d) – MaxEnt
  - Maximize conditional likelihood
  - Conditional Maximum Likelihood Estimation – not as simple as relative frequencies

Conditional Likelihood

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

$$logP(C|D,\lambda) = \sum_{(c,d)\in(C,D)} logp(c|d,\lambda)$$

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

Maximizing Conditional Log Likelihood

$$logP(C|D,\lambda) = \sum_{(c,d)\in(C,D)} logexp \sum_i \lambda_i f_i(c,d)$$

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

Write the log likelihood in 2 separate terms
Derivative of log likelihood is sum of derivative of each term

Maximizing Conditional Log Likelihood

$$logP(C|D,\lambda) = \sum_{(c,d)\in(C,D)} logexp \sum_i \lambda_i f_i(c,d)$$

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

Derivative of the above term is given by

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

Maximizing Conditional Log Likelihood

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

Empirical count (f_i, c)

Predicted count (f_i, λ)

- Optimum parameters when empirical expectation equals predicted expectation

Expectation of a Feature

- We can count the features from the labeled set of data

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

- Expectation of a feature given the trained model

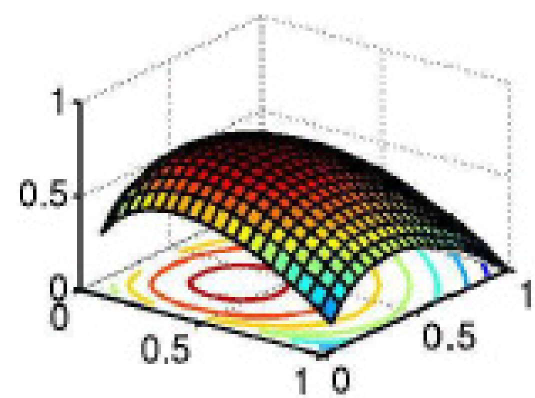
$$E(f_i) = \sum_{(c,d)\in(C,D)} p(c,d) f_i(c,d)$$

Optimal Parameters

- Chose parameters $\lambda_1, \lambda_2, ..., \lambda_n$ to maximize conditional log likelihood

$$logP(C|D,\lambda) = \sum_{(c,d)\in(C,D)} logp(c|d,\lambda)$$

- We showed how to compute partial derivatives with respect to different features
- For MaxEnt model this conditional log likelihood surface is convex
- Find maxima by doing gradient descent



Finding Optimal Parameters

- Commonly numerical optimization packages
- Gradient descent
- Stochastic gradient descent
- Quasi Newton Methods
- L-BFGS
- Generalized Iterative Scaling

Generalized Iterative Scaling

- Empirical Expectation $$E_{\tilde{p}}(f_j) = \frac{1}{N}\sum_{i=1}^{N} f_j(d_i, c_i)$$
- Initialize m+1 lambdas to 0
- Loop Until Converged

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

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

- End loop

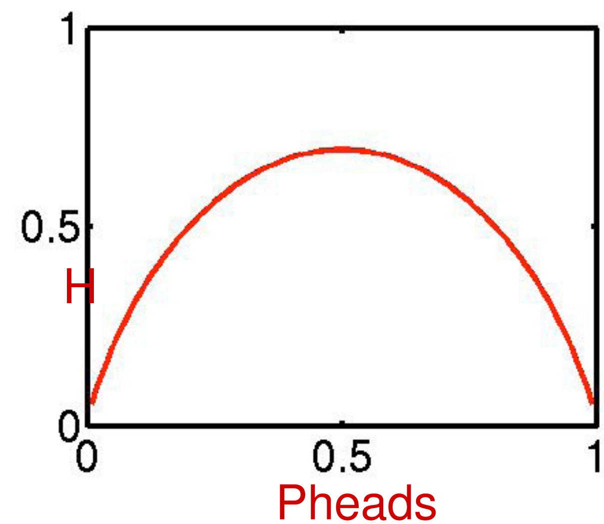
Other numerical methods are more common

Maximum Entropy

- We saw what entropy is

$$H(p) = -\sum_x p(x) log_2 p(x)$$

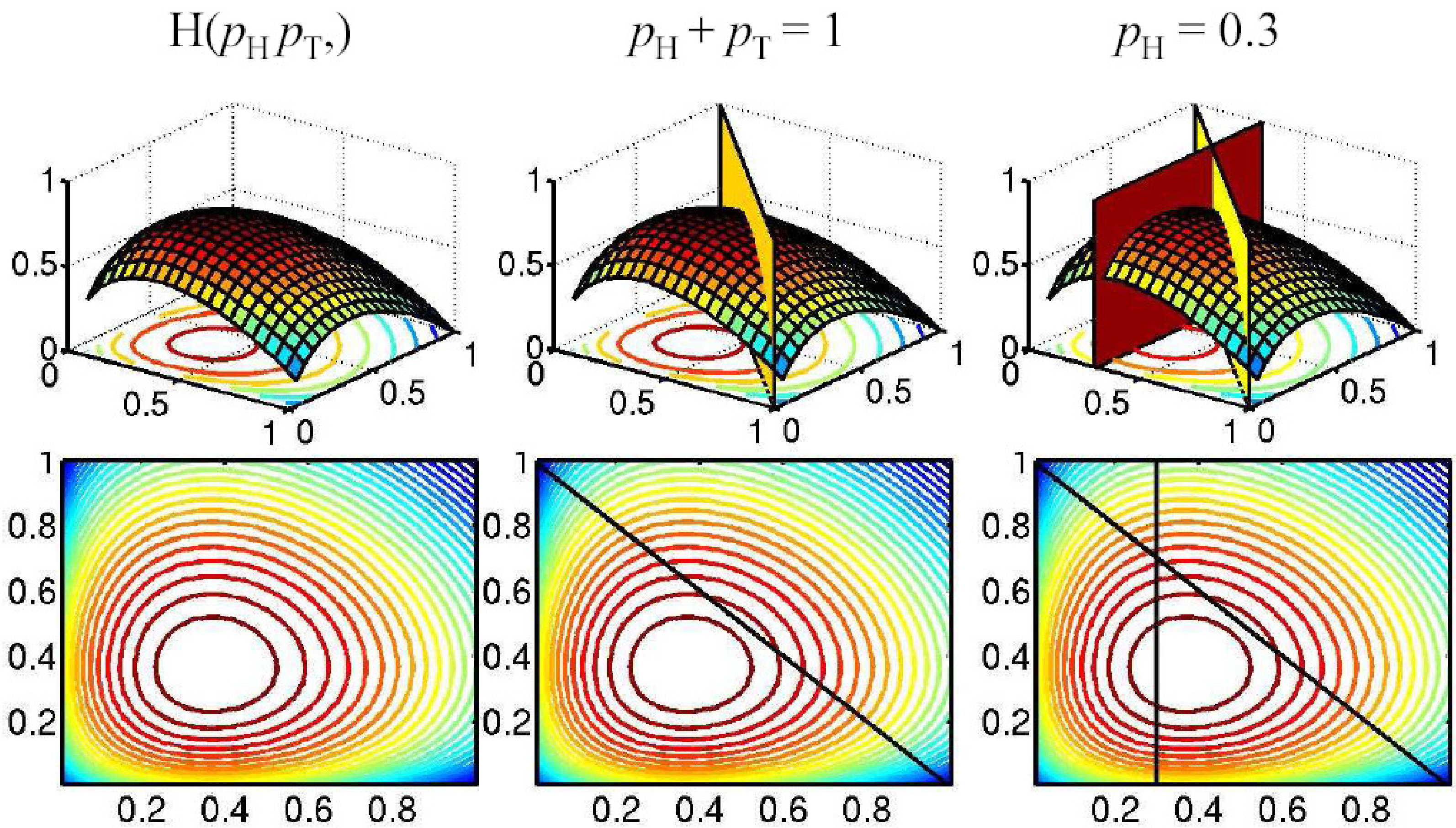
- We want to maximize entropy
  - Maximize subject to feature-based constraints
  - Feature based constraints help us bring the model distribution close to empirical distribution (data)
  - In other words it increases maximum likelihood of data given the model but makes the distribution less uniform



Fair coin has the highest entropy

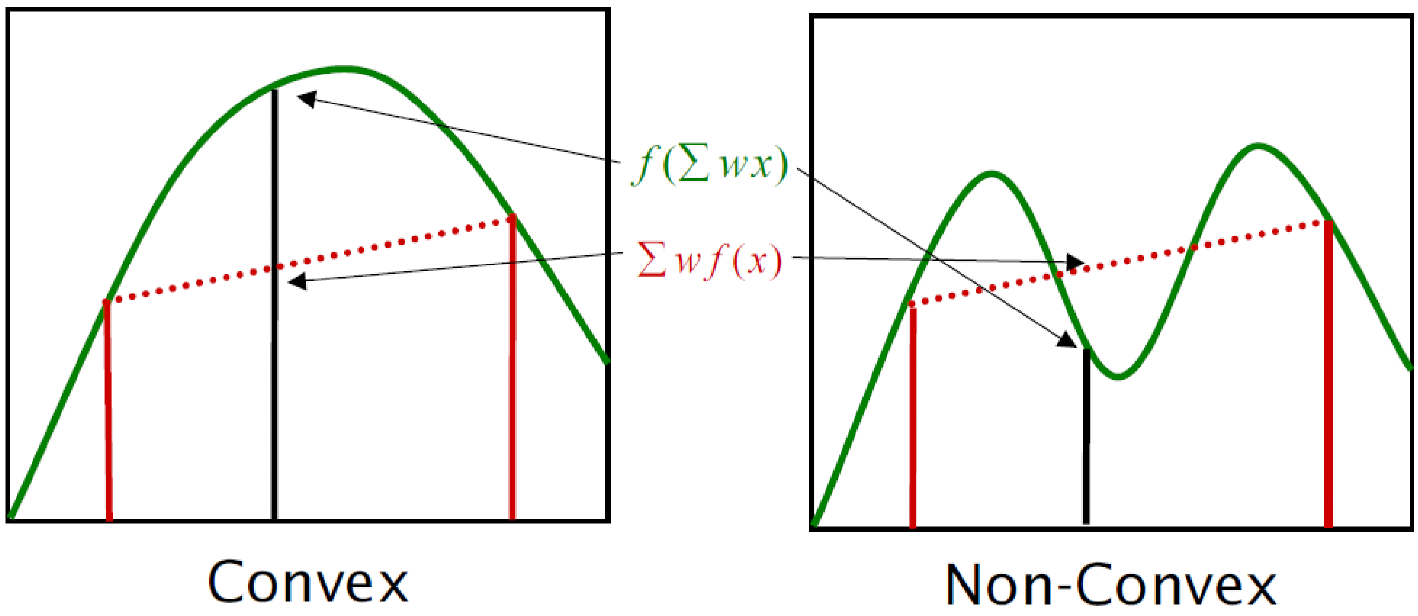
Constraints on a Entropy Function

Figure below is from Klein, D. and Manning, C., Tutorial [1]



Convexity

$$f(\sum_i w_i x_i) \geq \sum_i w_i f(x_i), \sum_i w_i = 1$$



picture[1]

Maximization with Constraints

$$max_{p(x)} H(p) = -\sum_x p(x) log p(x)$$

$$s.t. \sum_x p(x) f_i(x) = \sum_x p(\tilde{x}) f_i(x), i = 1...N$$

$$\sum_x p(x) = 1$$

Solving MaxEnt

- MaxEnt is a convex optimization problem with concave objective function and linear constraints
  - Solved with Lagrange multipliers

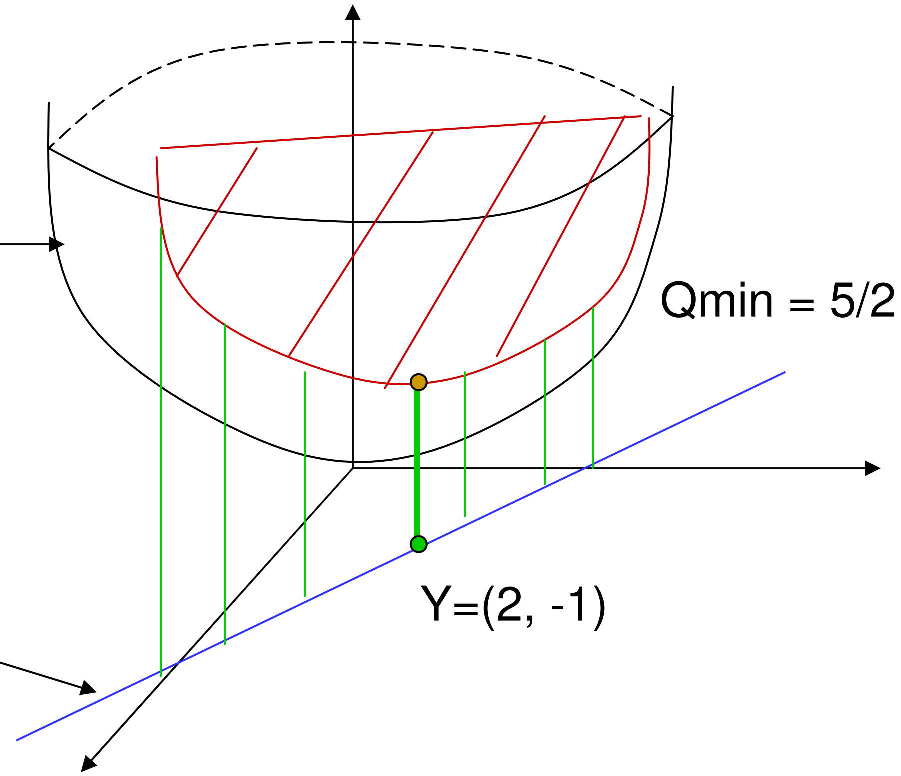
Constraints and Langrange Multiplers

Minimize

$$Q = \frac{1}{2}(x_1^2 + x_2^2)$$

With constraint of

$$2\ \mathrm{x}_1 - x_2 = 5$$



Lagrange Multipliers

- One way to handle constraints is to use Lagrange Multipliers
- Each of n constraints is multiplied by new variable q

$$L(x,q) = \frac{1}{2}(x_1^2 + x_2^2) + q_1(2x_1 - 2x_2 - 5)$$

3 unknowns are then identified by setting 3 partial derivatives to zero

Lagrange Multipliers

$$\frac{\partial L}{\partial q_1} = 2x_1 - x_2 - 5 = 0$$

$$\frac{\partial L}{\partial x_1} = x_1 + 2q_1 = 0$$

$$\frac{\partial L}{\partial x_2} = x_2 - q_1 = 0$$

Taking partial derivatives of 3 variables and setting them to zero gives us 3 simultaneous equation. We can solve these to get the values of x1 and x2

Lagrange Multipliers

Substitute $x_1 = -2q_1$ and $x_2 = q_1$ in the constraints

gives us $-4q_1 - q_1 - 5 = 0$ or $-5q_1 = 5$

hence $q_1 = -1$

Which gives us $x_1 = 2$ and $x_2 = -1$

Plugging in $x_1$and x_2 in Q

We get $Q(x) = \frac{1}{2}(2^2 + (-1)^2) = \frac{5}{2}$

Lagrange Equation for Maximum Entropy Model

$$L(p,\lambda) = -\sum_x p(x) log p(x) + \lambda_0 [\sum_x p(x) - 1] + \sum_{i=1}^{N} \lambda_i [\sum_x p(x) f_i(x) - \sum_x p(\tilde{x}) f_i(x)]$$

Lagrangian gives us unconstrained optimization as constraints are built into the equation. We can now solve it by setting derivatives to zero

Maximum Entropy and Logistic Regression

- This unconstrained optimization problem is a dual problem equivalent to estimating maximum likelihood of logistic regression model we saw before

Maximizing entropy subject to our constraints
Is equivalent to
Maximum likelihood estimation over exponential family of $p_\lambda(x)$

Language Modeling

- Isolated digits: implicit language model

$$p(\text{"}one\text{"}) = \frac{1}{11},\ p(\text{"}two\text{"}) = \frac{1}{11}, \ldots, p(\text{"}zero\text{"}) = \frac{1}{11},\ p(\text{"}oh\text{"}) = \frac{1}{11}$$

- All other word sequences have probability zero
- Language models describe what word sequences the domain allows
- The better you can model acceptable/likely word sequences, or the fewer acceptable/likely word sequences in a domain, the better a bad acoustic model will look
- e.g. isolated digit recognition, yes/no recognition

N-gram Models

- It's hard to compute
  p("and nothing but the truth")
- Decomposition using conditional probabilities can help

  p("and nothing but the truth") = p("and") x
  p("nothing"|"and") x p("but"|"and nothing") x
  p("the"|"and nothing but") x
  p("truth"|"and nothing but the")

The N-gram Approximation

- Q: What's a trigram? What's an n-gram?
  A: Sequence of 3 words. Sequence of n words.

- Assume that each word depends only on the previous two words (or n-1 words for n-grams)

p("and nothing but the truth") =	p("and") x p("nothing"|"and") x p("but"|"and nothing") x p("the"|"nothing but") x p("truth"|"but the")

- Trigram assumption is clearly false

  p(w | of the) vs. p(w | lord of the)
- Should we just make n larger?

  can run into data sparseness problem
- N-grams have been the workhorse of language modeling for ASR over the last 30 years
- Uses almost no linguistic knowledge

Bigram Model Example

training data:

JOHN READ MOBY DICK
MARY READ A DIFFERENT BOOK
SHE READ A BOOK BY CHER

testing data / what's the probability of:

JOHN READ A BOOK

$$p(JOHN \mid \triangleright) = \frac{count(\triangleright\ JOHN)}{count(\triangleright)} = \frac{1}{3}$$

$$p(READ \mid JOHN) = \frac{count(JOHN \cdot READ)}{count(JOHN)} = 1$$

$$p(A \mid READ) = \frac{count(READ \cdot A)}{count(READ)} = \frac{2}{3}$$

$$p(BOOK \mid A) = \frac{count(A \cdot BOOK)}{count(A)} = \frac{1}{2}$$

$$p(\triangleleft \mid BOOK) = \frac{1}{2}$$

$$p(w) = \frac{1}{3} \cdot 1 \cdot \frac{2}{3} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{2}{36}$$

Trigrams, cont'd

Q: How do we estimate the probabilities?

A: Get real text, and start counting…

Maximum likelihood estimate would say:

p("the"|"nothing but") =
 C("nothing but the") / C("nothing but")

where C is the count of that sequence in the data

Data Sparseness

- Let's say we estimate our language model from yesterday's court proceedings
- Then, to estimate
  p("to"|"I swear") we use
  count ("I swear to") / count ("I swear")
- What about p("to"|"I swerve") ?

  If no traffic incidents in yesterday's hearing,

  count("I swerve to") / count("I swerve")

= 0 if the denominator > 0, or else is undefined

Very bad if today's case deals with a traffic incident!

Language Model Smoothing

- How can we adjust the ML estimates to account to account for the effects of the prior distribution when data is sparse?
- Generally, we don't actually come up with explicit priors, but we use it as justification for *ad hoc* methods

Smoothing: Simple Attempts

- Add one: (V is vocabulary size)

$$p(z \mid xy) \approx \frac{C(xyz)+1}{C(xy)+V}$$

  Advantage: Simple
  Disadvantage: Works very badly
- What about delta smoothing:

$$p(z \mid xy) \approx \frac{C(xyz)+\delta}{C(xy)+V\delta}$$

  A: Still bad…..

Summary

- Logistic Regression
  - Maximize conditional log-likelihood to estimate parameters
- Maximum Entropy Model
  - Maximize entropy with feature constraints
  - Constrained maximization
- Solving for H(p) with maximum entropy is equivalent to maximizing conditional log-likelihood for our exponential model

References

- [1] Klein, D., Manning C., "Maximum Models, Conditional Estimation and Optimization" ACL 2003
- [2] Jurafsky, D. and Martin, J., J&M Book, 2^{nd} Edition
- [3] http://webdocs.cs.ualberta.ca/~swang/me.html