INF5830 – 2015 FALL
NATURAL LANGUAGE PROCESSING

Jan Tore Lønning, Lecture 12, 2.11

Today

- Feature selection 1(Oblig 2)
- Scikit-Learn from NLTK
- Linear classifiers
- Naive Bayes is log linear
- Logistic Regression
- Multinomial Logistic Regression = Maximum Entropy Classifiers

Machine Learning

Selecting
Cleaning
Tokenization
Lemmatizing?
“Munging”
…

Feature Selection
Arguably the most important step for the results

Example: Word Sense Disambiguation

An electric guitar and **bass** player stand off to one side, not really part of the scene,
just as a sort of nod to gringo expectations perhaps.

"Bag of words"-features

- Features: [*fishing, big, sound, player, fly, rod, pound, double, runs, playing, guitar, band*]
- The example: [0,0,0,1,0,0,0,0,0,0,1,0]
- Which words as features? How many?
  - NLTK, initially: The most frequent ones
  - There might be better ways to select (we return to this later)

- Many features
- Boolean values

Hard-line-serve

Number of word features	Hard		
0	0.802		
10	0.768		
20	0.764		
50	0.774		
100	0.800		
200	0.812		
500	0.830		
1000	0.842		
2000	0.846		
5000	0.844		

Hard-line-serve

Number of word features	Hard	Serve	
0	0.802	0.350	
10	0.768	0.550	
20	0.764	0.622	
50	0.774	0.692	
100	0.800	0.728	
200	0.812	0.766	
500	0.830	0.784	
1000	0.842	0.794	
2000	0.846	0.802	
5000	0.844	0.804	

Hard-line-serve

Number of word features	Hard	Serve	Line
0	0.802	0.350	0.528
10	0.768	0.550	0.528
20	0.764	0.622	0.534
50	0.774	0.692	0.576
100	0.800	0.728	0.688
200	0.812	0.766	0.706
500	0.830	0.784	0.744
1000	0.842	0.794	0.774
2000	0.846	0.802	0.802
5000	0.844	0.804	0.826

Base Line

Collocational features

An electric guitar and **bass** player stand off to one side, not really part of the scene,
just as a sort of nod to gringo expectations perhaps.

- With tags:
  - $[w_{i-2}, POS_{i-2}, w_{i-1}, POS_{i-1}, w_{i+1}, POS_{i+1}, w_{i+2}, POS_{i+2}]$
  - Example: [guitar, NN, and, CC, player, NN, stand, VB]
- Without tags:
  - $[w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}]$
  - Example: [guitar, and, player, stand]

- Few features
- Many possible values

Window size (without tags)

Words on each side	Hard	Serve	Line
0	0.802	0.350	0.528
1	0.898	0.742	0.734
2	0.886	0.818	0.772
3	0.868	0.856	0.776
4	0.864	0.856	0.782
5	0.854	0.858	0.768

Base Line

Both BoW and Colloc. features

Line.pos

BOW features	The size of the collocational window: 0	1	2	3	4	5
0	0.528	0.734	0.772	0.776	0.782	0.768
10	0.528	0.724	0.788	0.780	0.780	0.772
20	0.534	0.758	0.772	0.770	0.796	0.776
50	0.576	0.764	0.800	0.800	0.804	0.796
100	0.688	0.788	0.814	0.834	0.818	0.816
200	0.706	0.784	0.814	0.828	0.830	0.814
500	0.744	0.814	0.848	0.836	0.852	0.842
1000	0.774	0.836	0.860	0.864	0.854	0.848
2000	0.802	0.846	0.866	0.864	0.872	0.874
5000	0.826	0.872	0.886	0.886	0.894	0.890

Today

- Feature selection 1(Oblig 2)
- Scikit-Learn from NLTK
- Linear classifiers
- Naive Bayes is log linear
- Logistic Regression
- Multinomial Logistic Regression = Maximum Entropy Classifiers

Other ML algorithms in NLTK

- Included:
  - Naive Bayes (Bernoulli)
  - Decision trees
- Import from Scikit-Learn
  - Example:
    - from sklearn.linear_model import LogisticRegression
    - sk_classifier = SklearnClassifier(LogisticRegression())
    - sk_classifier.train(train_set)
  - Instead of:
    - classifier = nltk.NaiveBayesClassifier.train(train_set)
  - Then use the same set-up as in the oblig

Scikit-Learn

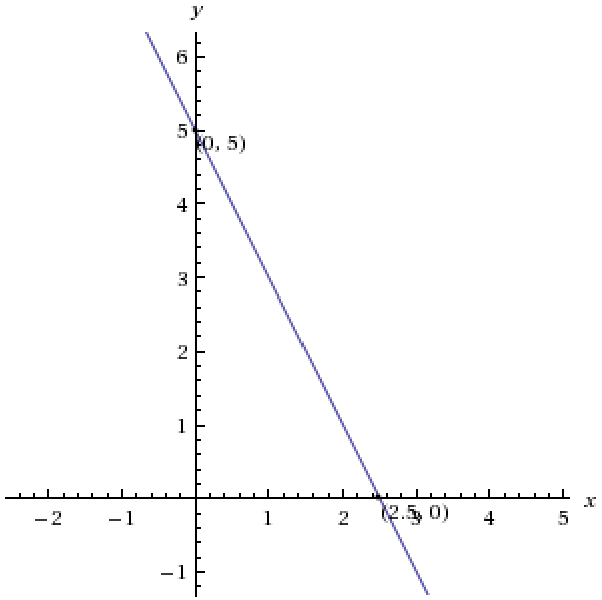
- A large set of various ML classification algorithms
  - They can be imported into NLTK
- In general faster than NLTK's algorithms
- Beware how the features are selected/formulated:
  - They may be reformulated/altered when translated into Scikit
  - Example:
    - SklearnClassifier(BernoulliNB()) performed inferior to nltk.NaiveBayesClassifier when we used the NLTK-features

Today

- Feature selection 1(Oblig 2)
- Scikit-Learn from NLTK
- Linear classifiers
- Naive Bayes is log linear
- Logistic Regression
- Multinomial Logistic Regression = Maximum Entropy Classifiers

Geometry: lines

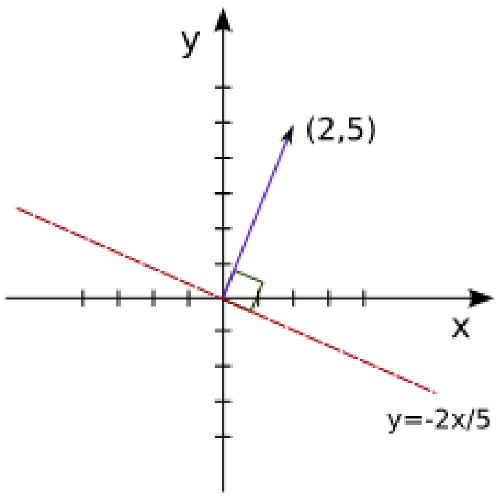
- Descartes
  - (1596-1650)
- Line:
- $ax + by + c = 0$
- If $b \neq 0$:
  - $y = mx + n$
  - $n = -c/b$ is the intercept with the y-axis
  - $m = -a/b$ is the slope
- A point = intersection of two lines



- $y = -2x + 5$
- $4x + 2y - 10 = 0$

Normal vector of a line

- $\cos(\pi/2) = 0$
- If P passes through (0,0) there is an **n** = (x_n, y_n) s.t.
- (x,y) is on P iff
  - $(x,y) \bullet (x_n, y_n) = 0$
  - $x \times x_n = - y \times y_n$
  - If $(a,b) \neq (0,0)$ is on P:
    - **n** $= s \times (b, -a)$ for some s

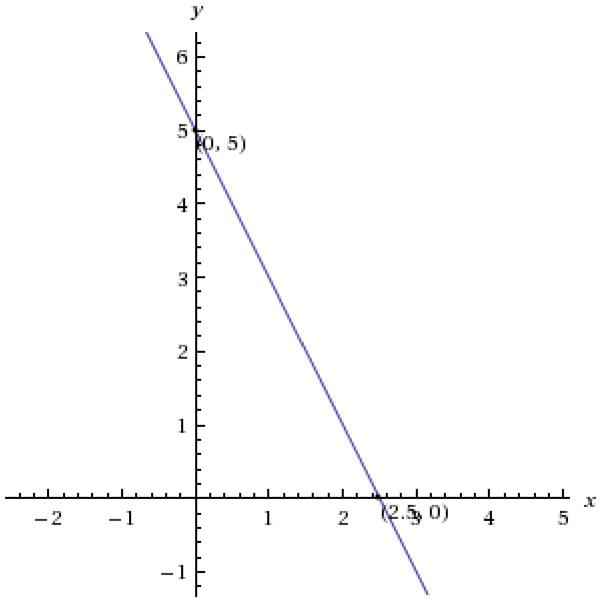


Vector (2,5) is normal to the line y=-2x/5

- Example:
  - $y = -2x/5$
  - $2x + 5y = 0$
  - $(x,y) \bullet (2,5) = 0$

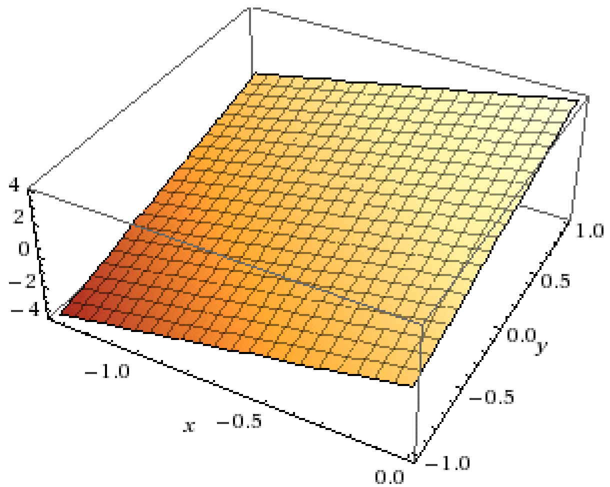
Lines not through (0,0)

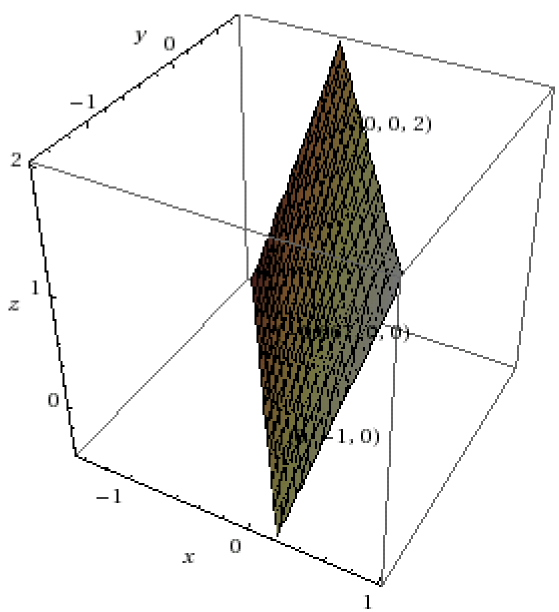
- $y = -2x + 5$
- $2x + y - 5 = 0$
- $(x,y) \bullet (2,1) = 5$



Geometry: planes

- Plane:
- $ax + by + cz + d = 0$
- If $c \neq 0$:
  - $z = mx + ny + n$
- A line is
  the intersection of two planes



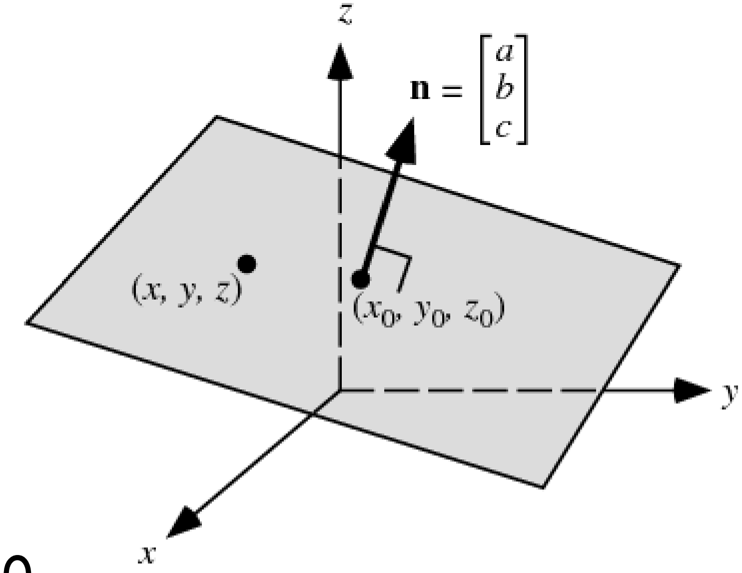


- $3x + 2y - z + 2 = 0$
- $z = 3x + 2y + 2$

http://www.univie.ac.at/future.media/moe/galerie/geom2/geom2.html#eb

Normal vector of a plane

- All points (x,y,z) where
- $((x,y,z)-(x_0,y_0,z_0))\bullet(a,b,c) = 0$
- $(x,y,z) \bullet (a,b,c) = d$
  - $(d = a\ x_0 + b\ y_0 + c\ z_0)$



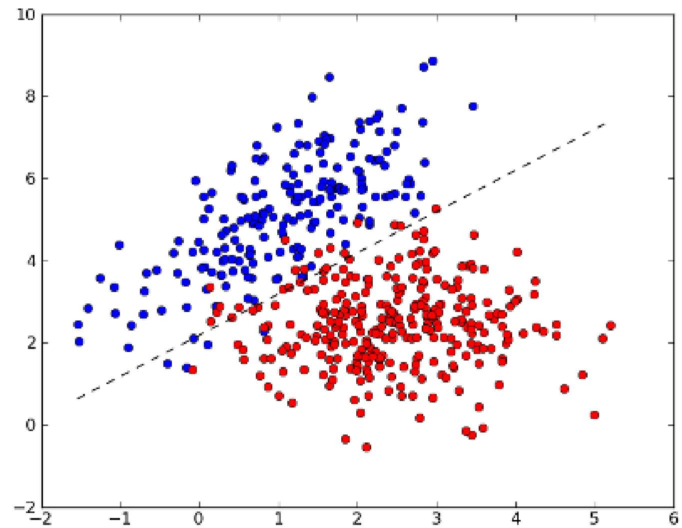
- Hyperplane
  - $w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = 0$
  - $(w_1, w_2, \ldots, w_n) \bullet (x_1, x_2, \ldots x_n) = -w_0$
- Sometimes (n+1 dimensions):
  - $(w_0, w_1, w_2, \ldots, w_n) \bullet (1, x_1, x_2, \ldots x_n) = 0$

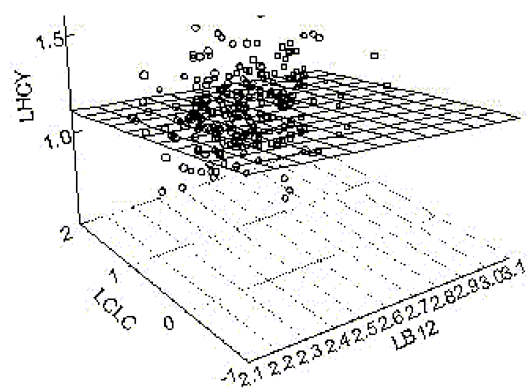
Hyperplanes

- Generalizes to higher dimensions
- In n-dimensional space $(x_1, x_2, \ldots, x_n)$:
  - Points satisfying:
- $w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n = 0$
  - for any choice of $w_0, w_1, w_2, \ldots w_n$
  - where not all of $w_1, w_2, \ldots w_n = 0$
- is called a hyper-plane
- (In machine learning) the same as the intersection of two hyper-planes in n+1 dimensional space:
  - $w_0x_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$
  - $x_0 = 1$

Linear classifiers

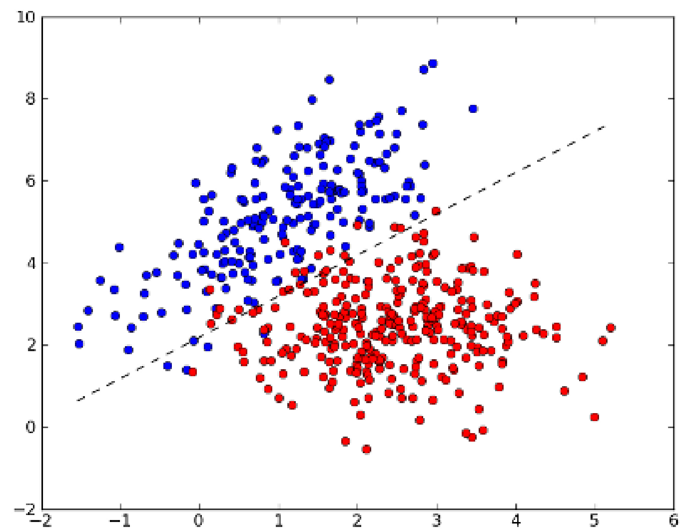
- Assume:
  - All features are numerical (including Boolean)
  - Two classes
- The two classes are <u>linearly separable</u> if they can be separated by a hyperplane
- In 2 dimensions that is a line:
  - $ax + by < c$ for red points
  - $ax + by > c$ for blue points

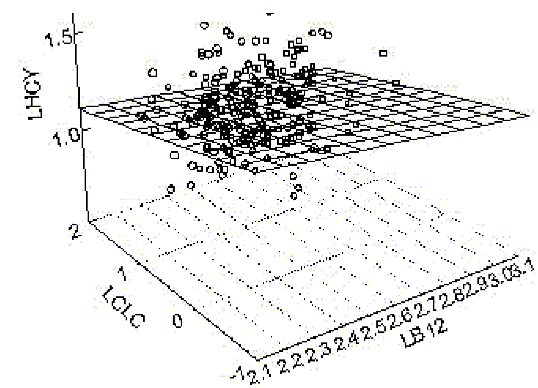




Linear classifiers

- A linear classifier introduces a hyperplane and classifies accordingly
- (If the data aren't linearly separable, the classifier will make mistakes).



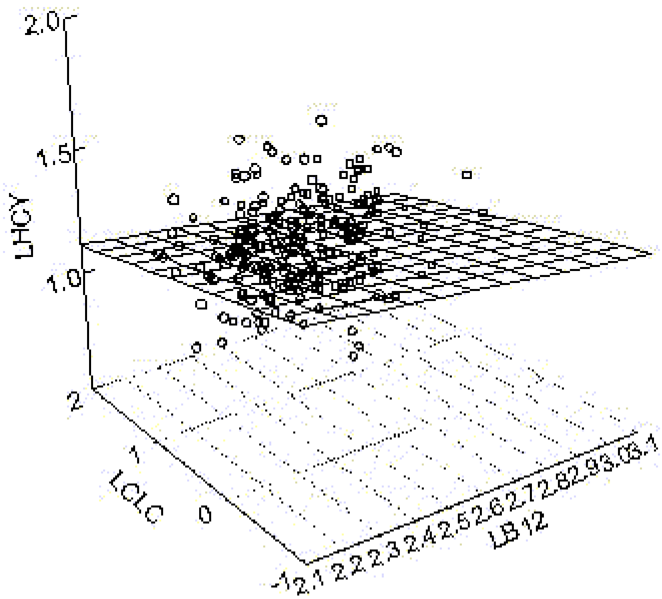


Linear classifiers – general case

- Try to separate the classes by a hyperplane

$$\sum_{i=1}^{M} w_i x_i = \theta$$

- (equivalently $\vec{w} \bullet \vec{x} = \sum_{i=0}^{M} w_i x_i = 0$
  - taking $w_0 = -\theta$ and $x_0 = 1$)
- The object represented by $(x_1, x_2, \ldots, x_n)$
  - is in C if and only if $\sum_{i=1}^{M} w_i x_i > \theta$
  - and in –C if $\sum_{i=1}^{M} w_i x_i < \theta$



Today

- Feature selection 1(Oblig 2)
- Scikit-Learn from NLTK
- Linear classifiers
- Naive Bayes is log linear
- Logistic Regression
- Multinomial Logistic Regression = Maximum Entropy Classifiers

Naive Bayes is a log linear classifier

$$\hat{c} = \arg\max_{c\in\{c_1,c_2\}} P(c)\prod_{j=1}^{n} P(f_j \mid c)$$

$$P(c_1)\prod_{j=1}^{n} P(f_j \mid c_1) > P(c_2)\prod_{j=1}^{n} P(f_j \mid c_2)$$

$$\frac{P(c_1)\prod_{j=1}^{n} P(f_j \mid c_1)}{P(c_2)\prod_{j=1}^{n} P(f_j \mid c_2)} > 1$$

$$\frac{P(c_1)}{P(c_2)}\prod_{j=1}^{n}\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)} > 1$$

$$\log\left(\frac{P(c_1)}{P(c_2)}\prod_{j=1}^{n}\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)}\right) > 0$$

$$\log\left(\frac{P(c_1)}{P(c_2)}\right) + \sum_{j=1}^{n}\log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)}\right) > 0$$

$$\sum_{i=1}^{M} w_i x_i = \theta \qquad w_j = \log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)}\right)$$

$$\theta = -w_0 = -\log\left(\frac{P(c_1)}{P(c_2)}\right)$$

A closer look: The Bernoulli model

$$\log\left(\frac{P(c_1)}{P(c_2)}\right) + \sum_{j=1}^{n} \log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)}\right) > 0 \qquad \sum_{i=1}^{M} w_i x_i = \theta \qquad w_j = \log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)}\right)$$

- A feature x_i equals 0 or1 and corresponds to the combination of
  - what we earlier registered as a feature, and
  - the value of such a feature

Example 1 (gender of names, NLTK), where one feature registers the last letter of the name

- Original view:
  - One (categorical) feature f1
  - 26 possible different values: a, b, c, …,z
- Current view:
  - 26 different features x1, x2, …, x26
  - Each takes as value 0 or 1
  - Exactly one equals 1, the rest equals 0

A closer look: The Bernoulli model

$$\log\left(\frac{P(c_1)}{P(c_2)}\right) + \sum_{j=1}^{n} \log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)}\right) > 0 \qquad \sum_{i=1}^{M} w_i x_i = \theta \qquad w_j = \log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)}\right)$$

- A feature x_i equals 0 or1 and corresponds to the combination of
  - what we earlier registered as a feature, and
  - the value of such a feature

Example 2: text categorization:

- Original view: one feature f_i for a term t_i:
  - $f_i = 1$ if t_i is present, $f_i = 0$ if t_i isn't present
- Current view
  - one term x_{2i} corresponding to t_i being present and one term x_{2i+1} corresponding to t_i being absent
  - One of these equals 1, the other equals 0

A closer look: the multinomial model

- The multinomial does not strictly fit the NB-model:

$$\log\left(\frac{P(c_1)}{P(c_2)}\right) + \sum_{j=1}^{n} \log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)}\right) > 0$$

- But it fits the linear model $\sum_{i=j}^{M} w_j x_j = \theta$
  - If
  - j is the index of a feature term (lexeme) t_j (not a particular occurrence in a document)
  - x_j is the number of occurrrences of t_j in the document
  - and w_j is

$$w_j = \log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)}\right)$$

Today

- Feature selection 1(Oblig 2)
- Scikit-Learn from NLTK
- Linear classifiers
- Naive Bayes is log linear
- Logistic Regression
- Multinomial Logistic Regression = Maximum Entropy Classifiers

NB and logistic regression

- The NB uses a linear expression to decide

$$\log\left(\frac{P(c_1 \mid \vec{f})}{P(c_2 \mid \vec{f})}\right) = \log\left(\frac{P(c_1 \mid \vec{f})}{1 - P(c_1 \mid \vec{f})}\right) = \vec{w} \bullet \vec{f} = \sum_{i=0}^{M} w_i x_i = w_0 x_0 + \sum_{i=1}^{M} w_i x_i > 0$$

- where

$$w_j = \log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)}\right)$$

- Are these the best choices for the w_is?
- Logistic regression instead faces the question directly:
- Which w_is make the best classifier of the form

$$\text{logit}\left(P(c_1 \mid \vec{f})\right) = \ln\left(\frac{P(c_1 \mid \vec{f})}{1 - P(c_1 \mid \vec{f})}\right) = \vec{w} \bullet \vec{f} = \sum_{i=0}^{M} w_i x_i = w_0 x_0 + \sum_{i=1}^{M} w_i x_i > 0$$

Logistic regression – learning

- Conditional maximum likelihood estimation: Choose the model that fits the training data best!

$$\hat{w} = \arg\max_{w} \prod_{i=1}^{m} P(c^i \mid \vec{f}^{\,i}) = \arg\max_{w} \sum_{i=1}^{m} \log P(c^i \mid \vec{f}^{\,i})$$

- where:
  - There are *m* many training data
  - c^i is the class of observation *i*, *i*.e. c_1 or c_2.
  - The feature vector for observation *i* is: $\vec{f}^{\,i} = (f_1^i, f_2^i, \ldots f_n^i)$

Furthermore

- To estimate

$$\hat{w} = \arg\max_{w} \prod_{i=1}^{m} P(c^i \mid \vec{f}^{\,i}) = \arg\max_{w} \sum_{i=1}^{m} \log P(c^i \mid \vec{f}^{\,i})$$

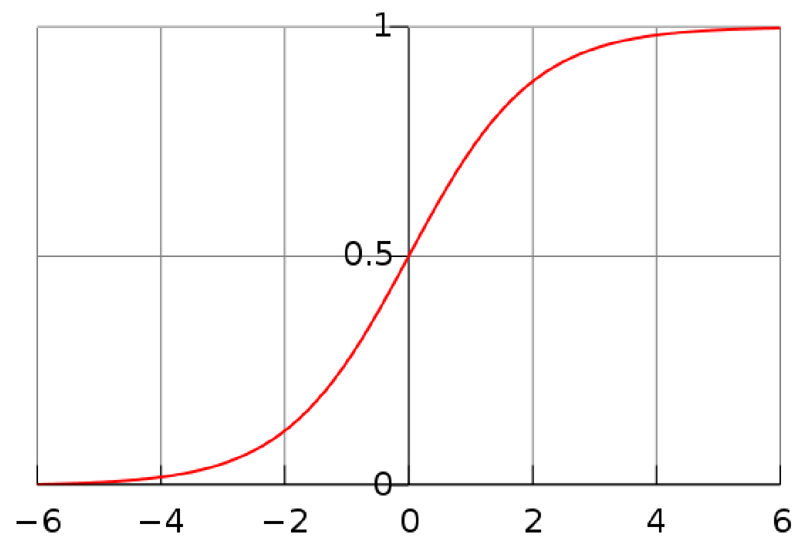
- we must find the relationship between **w** and $P(c^i \mid f^i)$

$$\ln\left(\frac{P(c_1 \mid \vec{f})}{1 - P(c_1 \mid \vec{f})}\right) = \vec{w} \bullet \vec{f}$$

$$\frac{P(c_1 \mid \vec{f})}{1 - P(c_1 \mid \vec{f})} = e^{\vec{w} \bullet \vec{f}}$$

$$P(c_1 \mid \vec{f}) = \frac{e^{\vec{w} \bullet \vec{f}}}{1 + e^{\vec{w} \bullet \vec{f}}}$$

$$P(c_1 \mid \vec{f}) = \frac{1}{1 + e^{-\vec{w} \bullet \vec{f}}}$$



Learning algorithms

- There is no analytic solution to

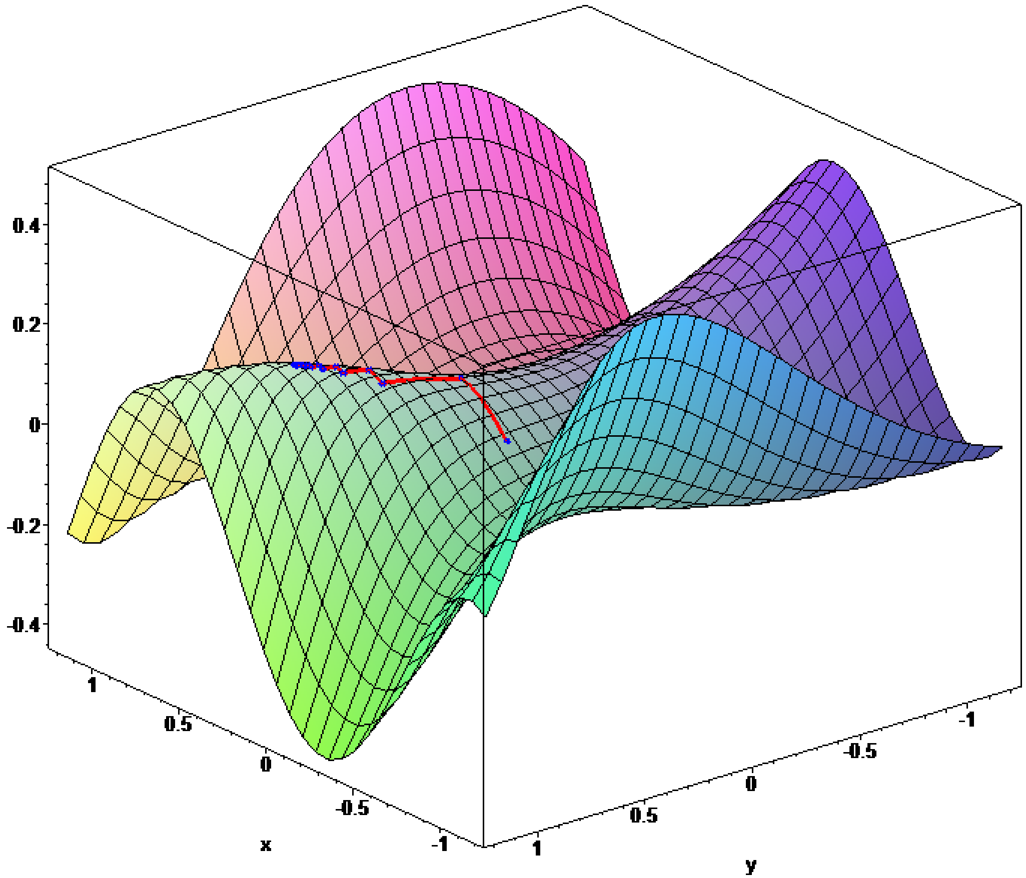
$$\hat{w} = \arg\max_{w} \sum_{i=1}^{m} \log P(c^i \mid \vec{f}^{\,i}) \qquad \text{where} \qquad P(c_1 \mid \vec{f}) = \frac{1}{1 + e^{-\vec{w} \bullet \vec{f}}}$$

- Use some numeric method which runs through a series of iterations
- e.g. gradient ascent (hill climbing)
  - There are partial derivatives (gradient) which points out the direction of the ascent
  - There is a global optimum: convergence
  - But we cannot predict how far to go.
- There is a tendency to overfitting, hence regularization

$$\hat{w} = \arg\max_{w} \sum_{i=1}^{m} \log P(c^i \mid \vec{f}^{\,i}) - \alpha R(w)$$

- Don’t try this at home! Use a package

Gradient ascent



Today

- Feature selection 1(Oblig 2)
- Scikit-Learn from NLTK
- Linear classifiers
- Naive Bayes is log linear
- Logistic Regression
- Multinomial Logistic Regression = Maximum Entropy Classifiers

A slight reformulation

- We saw that for NB

$$P(c_1 \mid \vec{f}) > P(c_2 \mid \vec{f}) \qquad P(c_1)\prod_{j=1}^{n} P(f_j \mid c_1) > P(c_2)\prod_{j=1}^{n} P(f_j \mid c_2)$$

- iff

$$\log\left(\frac{P(c_1)}{P(c_2)}\right) + \sum_{j=1}^{n} \log\left(\frac{P(f_j \mid c_1)}{P(f_j \mid c_2)}\right) > 0$$

- This could also be written

$$\left(\log P(c_1) - \log P(c_2)\right) + \sum_{j=1}^{n} \left(\log P(f_j \mid c_1) - \log P(f_j \mid c2)\right) > 0$$

$$\log P(c_1) + \sum_{j=1}^{n} \log P(f_j \mid c_1) > \log P(c_2) + \sum_{j=1}^{n} \log P(f_j \mid c2)$$

Reformualtion, contd.

- $\log P(c_1) + \sum_{j=1}^{n} \log P(f_j \mid c_1) > \log P(c_2) + \sum_{j=1}^{n} \log P(f_j \mid c2)$
- has the form $\vec{w}^1 \bullet \vec{f} = \sum_{i=0}^{M} w_i^1 x_i > \sum_{i=0}^{M} w_i^2 x_i = \vec{w}^2 \bullet \vec{f}$
- where
  - $w_j^1 = \log(P(f_j \mid c_1))$
  - $w_j^2 = \log(P(f_j \mid c_2))$
  - and our earlier $w_j = w_j^1 - w_j^2$
- So the probability in this notation

$$P(c_1 \mid \vec{f}) = \frac{e^{\vec{w} \bullet \vec{f}}}{1 + e^{\vec{w} \bullet \vec{f}}} = \frac{e^{(\vec{w}^1 - \vec{w}^2) \bullet \vec{f}}}{1 + e^{(\vec{w}^1 - \vec{w}^2) \bullet \vec{f}}} = \frac{e^{\vec{w}^1 \bullet \vec{f}}}{e^{\vec{w}^2 \bullet \vec{f}} + e^{\vec{w}^1 \bullet \vec{f}}}$$

- and similarly for $P(c_2 \mid \mathbf{f})$

Multinomial logistic regression

- We may generalize this to more than two classes
  - For each class c^j for j = 1,..,k
    - a linear expression $\vec{w}^j \bullet \vec{f} = \sum_{i=0}^{M} w_i^j x_i$
    - and the probability of belonging to class c^j:

$$P(c^j \mid \vec{f}) = \frac{1}{Z}\exp\left(\vec{w}^j \bullet \vec{f}\right) = \frac{1}{Z} e^{\vec{w}^j \bullet \vec{f}} = \frac{1}{Z} e^{\sum_i w_i^j f_i} = \frac{1}{Z}\prod_i \left(e^{w_i^j}\right)^{f_i} = \frac{1}{Z}\prod_i a_i^{f_i}$$

    - where $Z = \sum_{j=1}^{k} \exp\left(\vec{w}^j \bullet \vec{f}\right)$
    - and $a_i = e^{w_i^j}$

$$\frac{\text{Multinomial regression}}{\text{Logistic regression}} \approx \frac{\text{Naive Bayes (Bernoulli)}}{\text{Binary NB as linear classifier}}$$

Footnote: Alternative formulation

- (In case you read other presentations, like Mitchell or Hastie et. al.:
- They use a slightly different formulation, corresponding to
  - where for i = 1, 2,…, k-1:

$$P(c^i \mid \vec{f}) = \frac{1}{Z}\exp\left(\vec{w}^i \bullet \vec{f}\right) = \frac{1}{Z}e^{\vec{w}^i \bullet \vec{f}} = \frac{1}{Z}e^{\sum_j w_j^i f_j} = \frac{1}{Z}\prod_j \left(e^{w_j^i}\right)^{f_j} = \frac{1}{Z}\prod_j a_j^{f_j}$$

  - But $Z = 1 + \sum_{i=1}^{k-1} \exp\left(\vec{w}^i \bullet \vec{f}\right)$ and $P(c^k \mid \vec{f}) = \dfrac{1}{1 + \sum_{i=1}^{k-1} \exp\left(\vec{w}^i \bullet \vec{f}\right)}$

- The two formulations are equivalent though:
  - In the J&M formulation, divide the numerator and denominator in each P(c^i|**f**) with $\exp\left(\vec{w}^k \bullet \vec{f}\right)$
  - and you get this formulation (with adjustments to Z and **w**.)

Indicator variables

$$P(c^j \mid \vec{f}) = \frac{1}{Z}\exp\left(\vec{w}^j \bullet \vec{f}\right) = \frac{\exp\left(\vec{w}^j \bullet \vec{f}\right)}{\sum_{l=1}^{k}\exp\left(\vec{w}^l \bullet \vec{f}\right)} = \frac{\exp\left(\sum_{i=0}^{n} w_i^j f_i\right)}{\sum_{l=1}^{k}\exp\left(\sum_{i=0}^{n} w_i^l f_i\right)} = \frac{\exp\left(\sum_{i=0}^{m} w_i\, f_i(c^j, x)\right)}{\sum_{l=1}^{k}\exp\left(\sum_{i=0}^{m} w_i\, f_i(c^l, x)\right)}$$

- Already seen: categorical variables represented by indicator variables, taking the values 0,1
- Also usual to let the variables indicate both observation and class

Examples – J&M

We would like to know whether to assign the class *VB* to *race* (or instead assign some other class like *NN*). One useful feature, we'll call it f_1, would be the fact that the current word is *race*. We can thus add a binary feature which is true if this is the case:

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

Another feature would be whether the previous word has the tag *TO*:

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

Two more part-of-speech tagging features might focus on aspects of a word's spelling and case:

$$f_3(c,x) = \begin{cases} 1 & \text{if suffix}(word_i) = \text{"ing"} \ \& \ c = \text{VBG} \\ 0 & \text{otherwise} \end{cases}$$

Why called "maximum entropy"?

NN	JJ	NNS	VB	NNP	IN	MD	UH	SYM	VBG	POS	PRP	CC	CD	...
$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	$\frac{1}{45}$	...

P(NN)+P(JJ)+P(NNS)+P(VB)=1

NN	JJ	NNS	VB	NNP	IN	MD	UH	SYM	VBG	POS	PRP	CC	CD	...
$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	0	0	0	0	0	0	...

P(NN)+P(NNS)=0.8

NN	JJ	NNS	VB	NNP	...
$\frac{4}{10}$	$\frac{1}{10}$	$\frac{4}{10}$	$\frac{1}{10}$	0	...

P(VB)=1/20

NN	JJ	NNS	VB
$\frac{4}{10}$	$\frac{3}{20}$	$\frac{4}{10}$	$\frac{1}{20}$

See NLTK book for a further example

Why called ”maximum entropy”?

- The multinomial logistic regression yields the probability distribution which
  - Gives the maximum entropy
  - Given our training data

Learning

- Similarly to the binary logistic regression,
- Regularization

NLTK: Some iterative optimization techniques are much faster than others.

- When training Maximum Entropy models, avoid the use of
  - Generalized Iterative Scaling (GIS) or
  - Improved Iterative Scaling (IIS),
- which are both considerably slower than the
  - Conjugate Gradient (CG) and
  - the BFGS optimization methods.

Line – Most frequen BoW-features

Number of word features	NaiveBayes	SklearnClassifier(LogisticRegression())
0	0.528	0.528
10	0.528	0.528
20	0.534	0.546
50	0.576	0.624
100	0.688	0.732
200	0.706	0.752
500	0.744	0.804
1000	0.774	0.838
2000	0.802	0.846
5000	0.826	0.850