INF5830 – 2017 FALL
NATURAL LANGUAGE PROCESSING

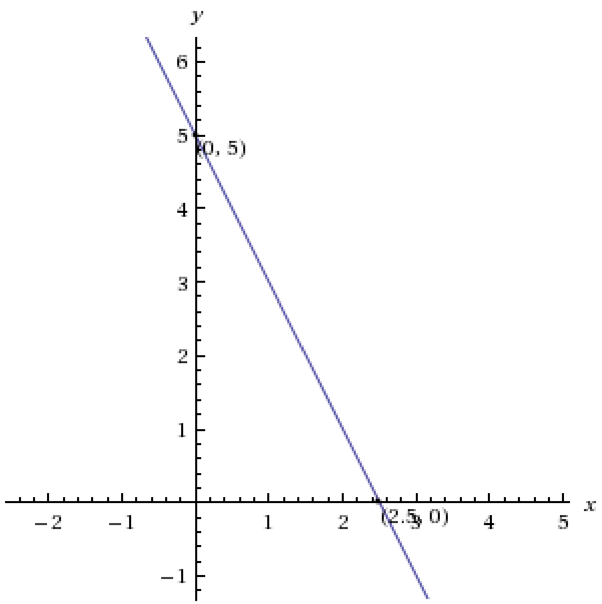
Jan Tore Lønning, Lecture 10, 24.10

Today

- Linear classifiers
- Naive Bayes is log linear
- Logistic Regression
- Multinomial Logistic Regression = Maximum Entropy Classifiers
- Comparing Naïve Bayes and Logistic regression

Geometry: lines

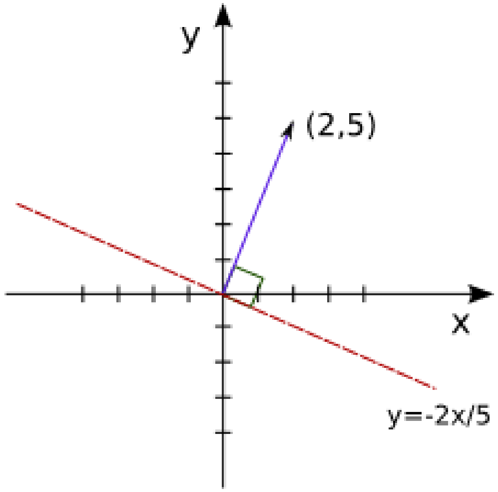
- Descartes
  - (1596-1650)
- Line:
- $ax + by + c = 0$
- If $b \neq 0$:
  - $y = mx + n$
  - $n = -c/b$ is the intercept with the y-axis
  - $m = -a/b$ is the slope
- A point = intersection of two lines



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

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

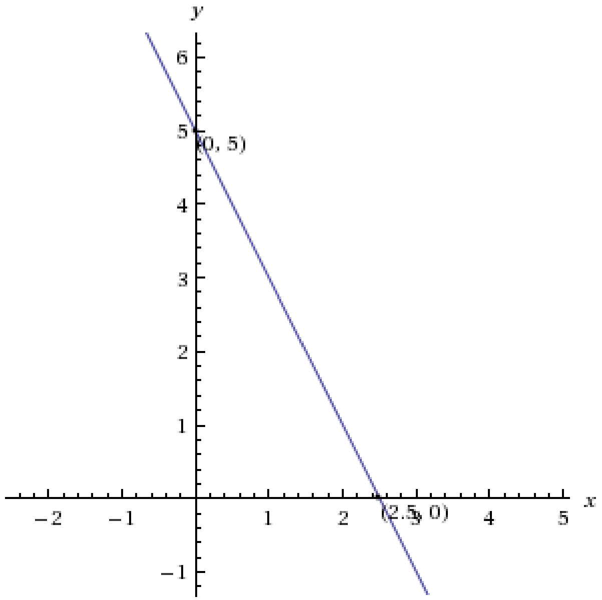


Vector (2,5) is normal to the line y=-2x/5

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

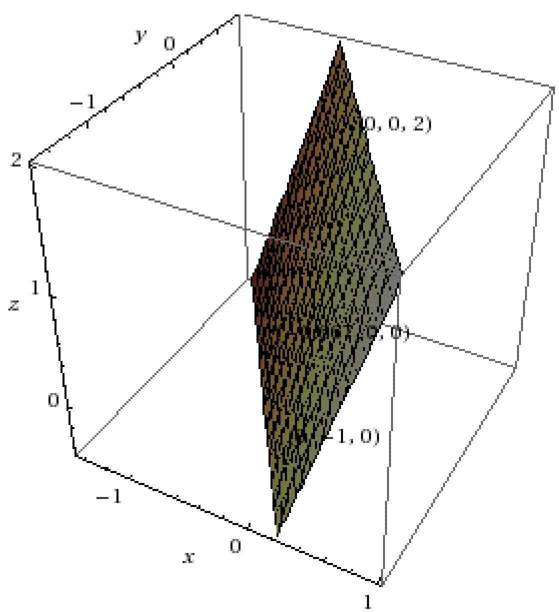
Lines not through (0,0)

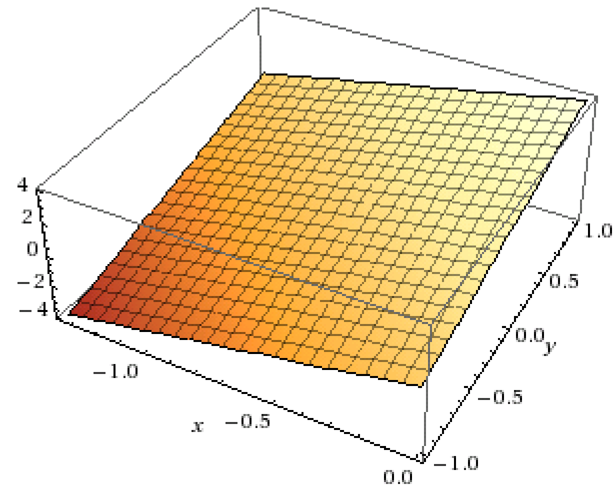
- $y = -2x + 5$
- $2x + y - 5 = 0$
- $(x,y) \bullet (2,1) = 5$
- $((x,y)-(2.5, 0) \bullet (2,1) = 0$
- $((x,y)-(0,5)) \bullet (2,1) = 0$
- $((x,y)-(x_0, y_0)) \bullet (2,1) = 0$
  - For any (x_0, y_0) on the line



Geometry: planes

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



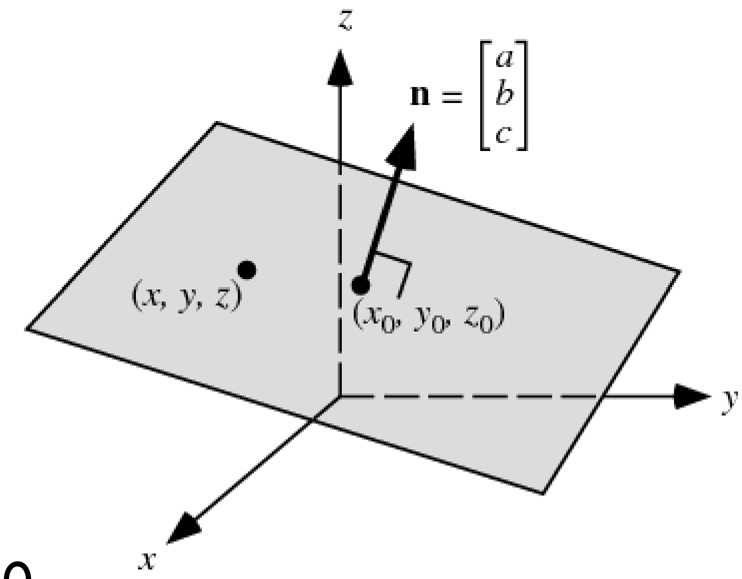


- $3x + 2y - z + 2 = 0$
- $z = 3x + 2y + 2$

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

Normal vector of a plane

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



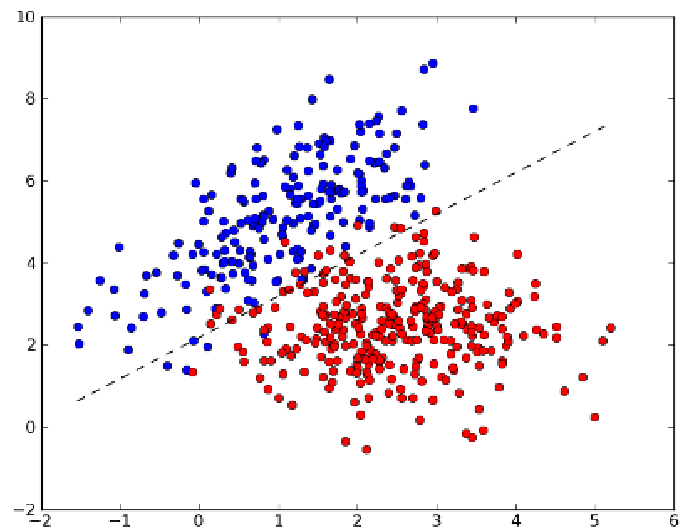
- Hyperplane
  - $w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = 0$
  - $(w_1, w_2, \ldots, w_n) \bullet (x_1, x_2, \ldots x_n) = -w_0$
- Sometimes (n+1 dimensions):
  - $(w_0, w_1, w_2, \ldots, w_n) \bullet (1, x_1, x_2, \ldots x_n) = 0$

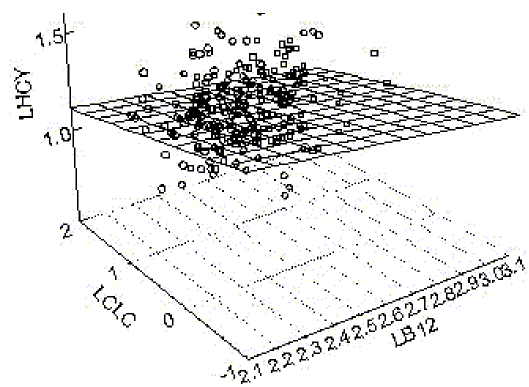
Hyperplanes

- 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

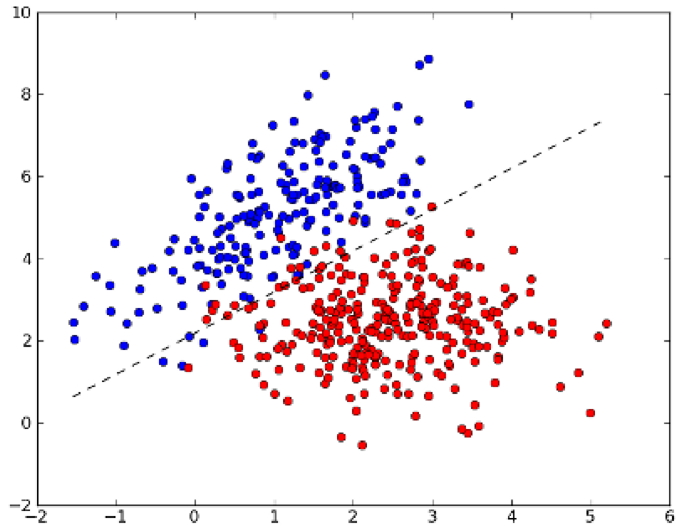
- Assume:
  - All features are numerical (including Boolean)
  - Two classes
- The two classes are <u>linearly separable </u>if they can be separated by a hyperplane
- In 2 dimensions that is a line:
  - $ax + by < c$ for red points
  - $ax + by > c$ for blue points

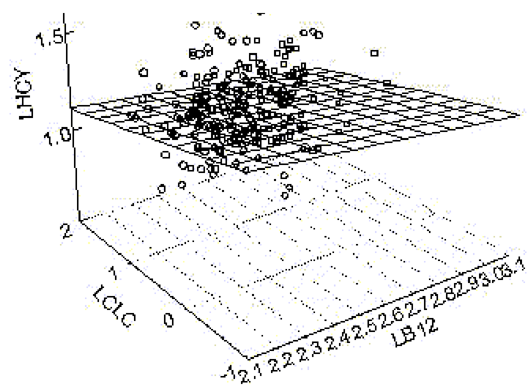




Linear classifiers

- A linear classifier introduces a hyperplane and classifies accordingly
- If the data aren't linearly separable, the classifier will make mistakes.
- Then: goal to make as few mistakes as possible



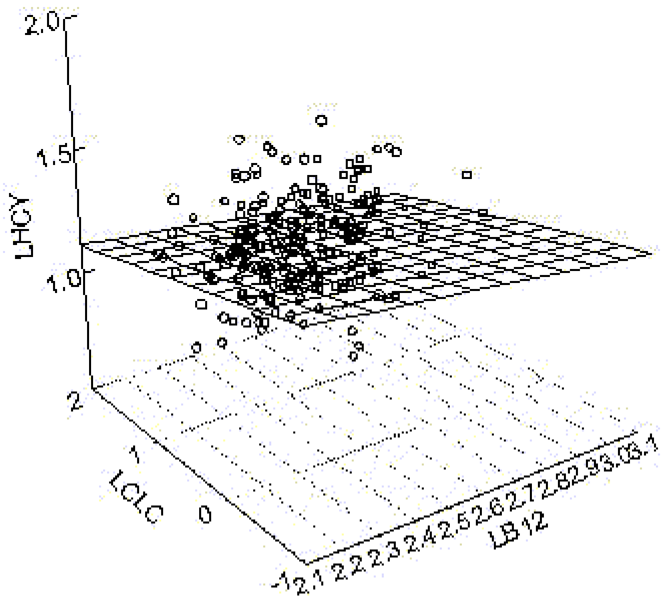


Linear classifiers – general case

- 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

- Linear classifiers
- Naive Bayes is log linear
- Logistic Regression
- Multinomial Logistic Regression = Maximum Entropy Classifiers
- Comparing Naïve Bayes and Logistic regression

Towards logistic regression

- Two ways to approach logistic regressions

1. Start with linear classifier and try to derive probabilistic classifier (e.g., J&M)
2. Start with NB and show that it is log-linear
   Consider log.reg. as a generalization

- We choose alt. 2 to supplement J&M, and because
- the reasons for choosing the logistic function in (1) are not obvious.

Naive Bayes is a log linear classifier

- We start with a binary classifier, with classes: C_1, C_2= not C_1
- For a given feature vector $\vec{f}$, we choose C_1:
  - iff: $P(C_1|\vec{f}) > P(-C_1|\vec{f}) = P(C_2|\vec{f})$
  - iff: $P(c_1)\prod_{j=1}^{n} P(f_j | c_1) > P(c_2)\prod_{j=1}^{n} P(f_j | c_2)$ (by NB assumption)
  - Iff (by Bayes' formula):

$$\log P(c_1 | \vec{f}) - \log P(c_2 | \vec{f}) = \log P(c_1) + \sum_{j=1}^{n} \log P(f_j | c_1) - \log P(c_2) + \sum_{j=1}^{n} \log P(f_j | c2) > 0$$

- which is a linear expressions

Naive Bayes is a log linear classifier

- How does this linear expression

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

correspond to a linear classifier?

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

- What in the NB-classifier corresponds to w_i and x_i in the linear classifier?

$$\sum_{i=1}^{M} w_i x_i = \theta \qquad \log P(c_1) + \sum_{j=1}^{n} \log P(f_j \mid c_1) - \log P(c_2) + \sum_{j=1}^{n} \log P(f_j \mid c2) > 0$$

Example 1 (gender of names, NLTK),

the only feature registers the last letter of the name

- Original view:
  - One (categorical) feature f1
  - 26 possible different values: a, b, c, …,z

Indicator variables

Example 1 Current view:

- 52 different features x1, x2, …, x52
- Each corresponds to a pair of last letter and class, e.g.
- x1=f1(letter|class) = 1 if letter='a' and class='fem'
  - 0 otherwise
- x2=f31(letter|class) = 1 if letter='e' and class='masc'
  - 0 otherwise
- $w_{31} = P(last = e|C_2)$
- Exactly two of these xi-s will equal1, the rest equals 0
- In addition, two features, $x_{53}=x_{54}=1$, $w_{53}=P(C_1)$, $w_{54}=P(C_2)$

Example 2: Bernoulli text classif.

$$\log P(c_1) + \sum_{j=1}^{n} \log P(f_j \mid c_1) - \log P(c_2) + \sum_{j=1}^{n} \log P(f_j \mid c2) > 0 \qquad \sum_{i=1}^{M} w_i x_i = \theta$$

- 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
  - Four features x1, x2, …, x4 corresponding to
    - $w_1 = P(contains('bob') = TRUE|'pos')$
    - $w_2 = P(contains('bob') = FALSE|'pos')$
    - $w_3 = P(contains('bob') = TRUE|'neg')$
    - $w_4 = P(contains('bob') = FALSE|'neg')$
  - Two xi-s will be 1 (w1, w3 or w2, w4) and two will be 0

Example 3: the multinomial model

$$\log P(c_1) + \sum_{j=1}^{n} \log P(f_j \mid c_1) - \log P(c_2) + \sum_{j=1}^{n} \log P(f_j \mid c2) > 0 \qquad \sum_{i=1}^{M} w_i x_i = \theta$$

- Original view: one feature for each term e.g. f1 for 'alice' and f2 for 'bob'.
- Here two values for each term,
  - $w_1 = P('alice'|'pos')$
  - $w_2 = P('alice'|'neg')$
  - $w_3 = P('bob'|'pos')$
  - $w_4 = P('bob'|'neg')$
  - x1=x2=3, if *alice* occurs 3 times
  - x3=x4=0, if *bob* occurs 0 times

Today

- Linear classifiers
- Naive Bayes is log linear
- Logistic Regression
- Multinomial Logistic Regression = Maximum Entropy Classifiers
- Comparing Naïve Bayes and Logistic regression

NB and logistic regression

- The NB uses a linear expression to decide

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

- Where the w_i-s are determined by estimating probabilities of the type
  - $P(C_1)$, the class probability, and
  - $P\left(f_i \middle| C_j\right)$and the probability of seeing a feature for the given class
- 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 above?

Logistic regression – learning

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

$$\hat{w} = \arg\max_{w} \prod_{i=0}^{m} P(c_{k_i} \mid \vec{f}^{\,i}) = \arg\max_{w} \sum_{i=0}^{m} \log P(c_{k_i} \mid \vec{f}^{\,i})$$

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

Furthermore

- To estimate

$$\hat{w} = \arg\max_{w} \prod_{i=0}^{m} P(c_{k_i} \mid \vec{f}^{\,i}) = \arg\max_{w} \sum_{i=0}^{m} \log P(c_{k_i} \mid \vec{f}^{\,i})$$

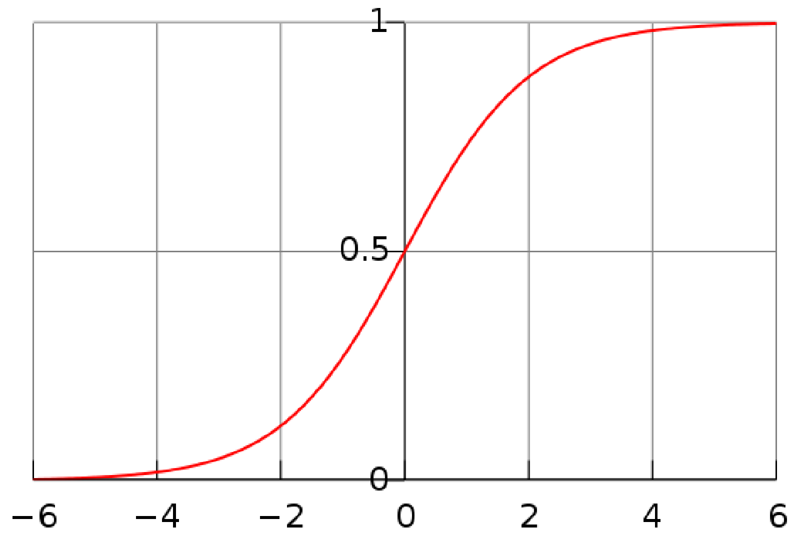
- we must find the relationship between **w** and $P(c^i \mid f^i)$

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

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

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

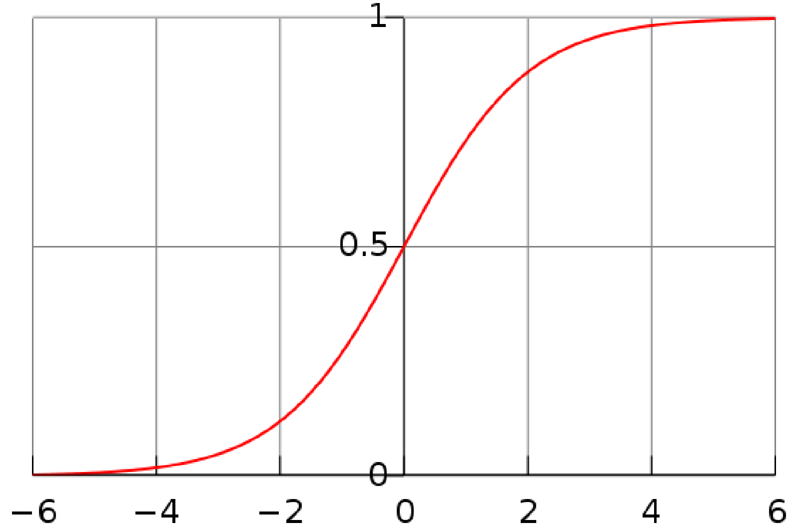
$$P(c_1 \mid \vec{f}) = \frac{1}{1 + e^{-\vec{w} \bullet \vec{f}}}$$



Logistic function

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

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



- Takes values between 0 and 1
- Can express probabilities.
- Useful for transforming discrete values to probs.
- Used e.g. in "deep learning"
- Mathematically "well-behaved"
- Used to model population growth, disease spreading etc.

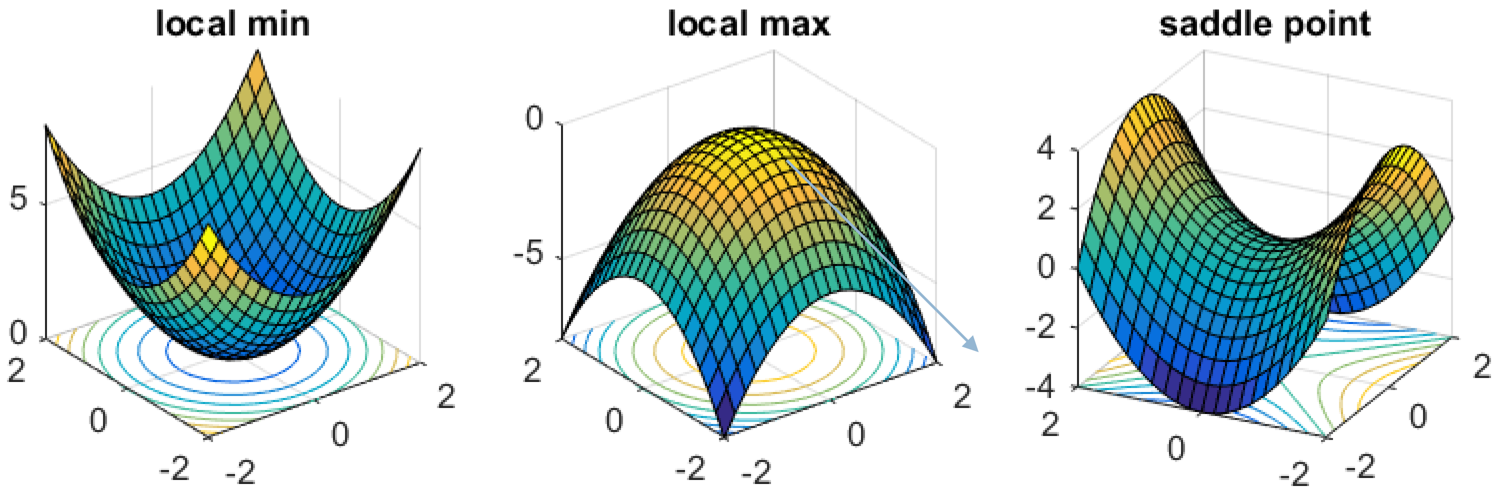
Learning algorithms

- 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 iterative algorithm:
  - start with an initial value for $\vec{w}$ and step by step try better candidates
- The problem is convex: There is a global optimum and we will not be caught in local – non-global – optima.
- Possible algorithms
  - (Hill climbing: optimize for one w_i after the other)
  - Some sort of gradient ascent: use derivatives to find optimal direction

Convex Gradient ascent



convex

Learning

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.

Regularization

- There is a tendency to overfitting, hence regularization

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

- The regularization punishes large weights
- Most common is L2-regularization $R(W) = \sum_0^n w_i^2$
- Alternative: L1-regularization $R(W) = \sum_0^n |w_i|$

scikit-learn – LogisticRegression

- `LogisticRegression(penalty=’l2’, …, C=1.0, …)`
- Uses L2-regularization as default
- Obligatory assignment 2.4:
  - Without regularization (C=10000), you loose ca 0.05
  - With C=0.1, I gained ca 0.005 on accuracy

Today

- Linear classifiers
- Naive Bayes is log linear
- Logistic Regression
- Multinomial Logistic Regression = Maximum Entropy Classifiers
- Comparing Naïve Bayes and Logistic regression

A slight reformulation

- We have formulated
- $P(c_1 \mid \vec{f}) > P(c_2 \mid \vec{f})$ as
- $\log P(c_1 \mid \vec{f}) - \log P(c_2 \mid \vec{f}) = \vec{w} \bullet \vec{f} = \sum_{i=0}^{M} w_i x_i > 0$
- But we could have stuck to the inequality and formulated is as $\vec{w}^1 \bullet \vec{f} = \sum_{i=0}^{M} w_i^1 x_i > \sum_{i=0}^{M} w_i^2 x_i = \vec{w}^2 \bullet \vec{f}$

  where $\log P(c_1 \mid \vec{f}) = \vec{w}^1 \bullet \vec{f} = \sum_{i=0}^{M} w_i^1 x_i$ $\quad \log P(c_2 \mid \vec{f}) = \sum_{i=0}^{M} w_i^2 x_i = \vec{w}^2 \bullet \vec{f}$

  (where we collect the indicator variables of the form $f(C_1, \ldots)$ to the left and they of the form $f(C_2, \ldots)$ to the right)

Reformulation, contd.

- $\log P(c_1 \mid \vec{f}) = \vec{w}^1 \bullet \vec{f} = \sum_{i=0}^{M} w_i^1 x_i$ and $\log P(c_2 \mid \vec{f}) = \sum_{i=0}^{M} w_i^2 x_i = \vec{w}^2 \bullet \vec{f}$

- in this notation

$$P(c_1 \mid \vec{f}) = \frac{P(c_1 \mid \vec{f})}{P(c_1 \mid \vec{f}) + P(c_2 \mid \vec{f})} = \frac{e^{\vec{w}^1 \bullet \vec{f}}}{e^{\vec{w}^1 \bullet \vec{f}} + e^{\vec{w}^2 \bullet \vec{f}}}$$

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

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 $a_i = e^{w_i^j}$

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

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

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

Today

- Linear classifiers
- Naive Bayes is log linear
- Logistic Regression
- Multinomial Logistic Regression = Maximum Entropy Classifiers
- Comparing Naïve Bayes and Logistic regression

Comparing NB and LogReg

- NB is a generative classifier:
  - It has a model of how the data are generated
  - $P(C)P\left(\vec{f}\middle|C\right) = P(\vec{f}, C)$
- LogReg is a discriminative classifier
  - It only considers the conditional probability $P\left(C|\vec{f}\right)$
- NB is an instance of LogReg,
  - i.e. one possible choice of weights
- LogReg will always do at least as well as NB on the <u>training</u> data

Comparing NB and LogReg

- LogReg will always do at least as well as NB on the training data
- When the independence assumptions of NB holds, NB will do as well as LogReg
- When the independence assumptions does not hold, NB may put too much weight on some features
- LogReg will not do this: If we add features that depend on other features, LogReg will put less weight on them

LogReg

- LogReg is prone to overfitting to the training data:
  - Use regularization.
- Adding more features will not disturb LogReg (on the training data.)
- To see which features are important for LogReg, use ablation:
  - Throw in all
  - Remove one after the other
  - But you may remove f1 or f2 but not both of f1 and f2