Naïve Bayes Recap

- Bag of words (order independent)

- Features are assumed independent given class

$$P(x_1, \ldots, x_n|c) = P(x_1|c) \ldots P(x_n|c)$$

Q: Is this really true?

The problem with assuming conditional independence

- Correlated features -> double counting evidence
  - Parameters are estimated independently

- This can hurt classifier accuracy and calibration

Logistic Regression

- Doesn’t assume features are independent
- Correlated features don’t “double count”

What are “Features”?

- A feature function, f
  - Input: Document, D (a string)
  - Output: Feature Vector, X

What are "Features"?

$$f(d) = \begin{pmatrix} \text{count(“boring”)} \\ \text{count(“not boring”)} \\ \text{length of document} \\ \text{author of document} \\ \vdots \end{pmatrix}$$

Doesn't have to be just "bag of words"

Feature Templates

- Typically “feature templates” are used to generate many features at once

- For each word:
  - ${w}_count
  - ${w}_lowercase
  - ${w}_with_NOT_before_count

Logistic Regression: Example

- Compute Features:

$$f(d_i) = x_i = \begin{pmatrix} \text{count(“nigerian”)} \\ \text{count(“prince”)} \\ \text{count(“nigerian prince”)} \end{pmatrix}$$

- Assume we are given some weights:

$$w = \begin{pmatrix} -1.0 \\ -1.0 \\ 4.0 \end{pmatrix}$$

Logistic Regression: Example

- Compute Features
- We are given some weights
- Compute the dot product:

$$z = \sum_{i=0}^{|X|} w_i x_i$$

Logistic Regression: Example

- Compute the dot product:

$$z = \sum_{i=0}^{|X|} w_i x_i$$

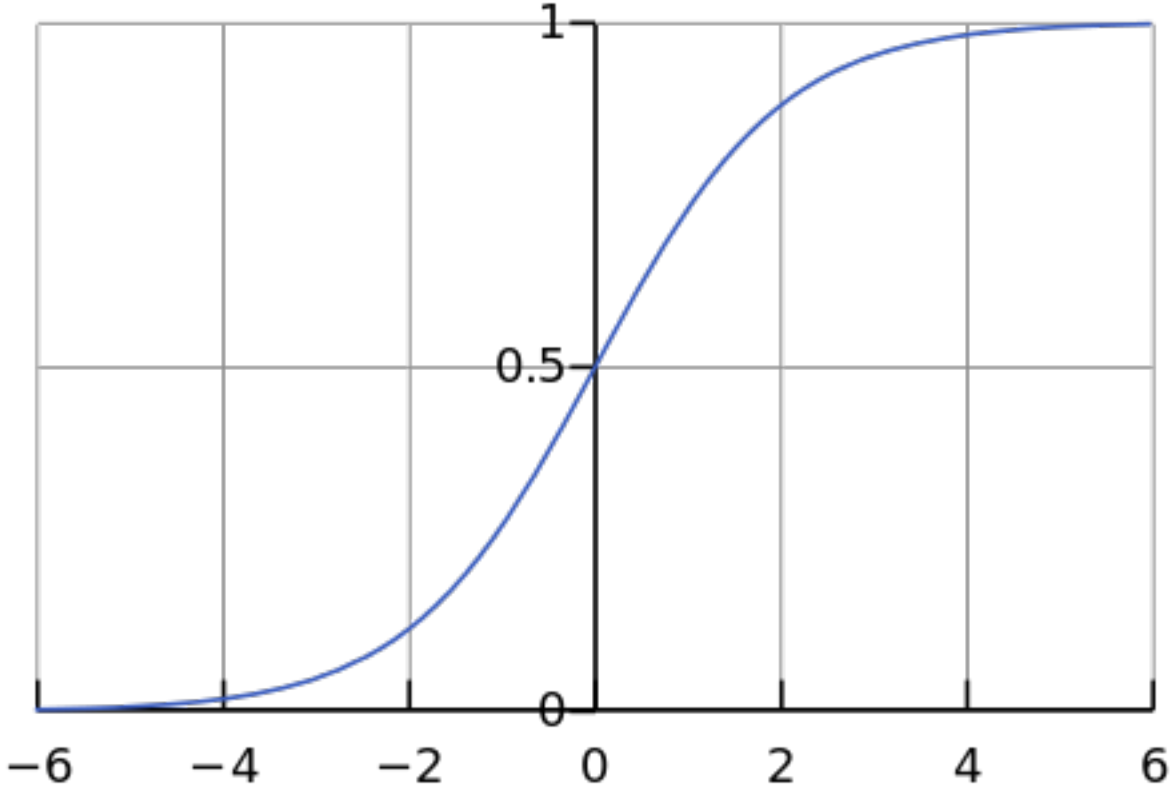
linear combination

- Compute the logistic function:

convert into probabilities between [0, 1]

$$P(\text{spam}|x) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

The Logistic function



$$P(\text{spam}|x) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

Logistic Regression

- (Log) Linear Model – similar to Naïve Bayes

The Dot Product

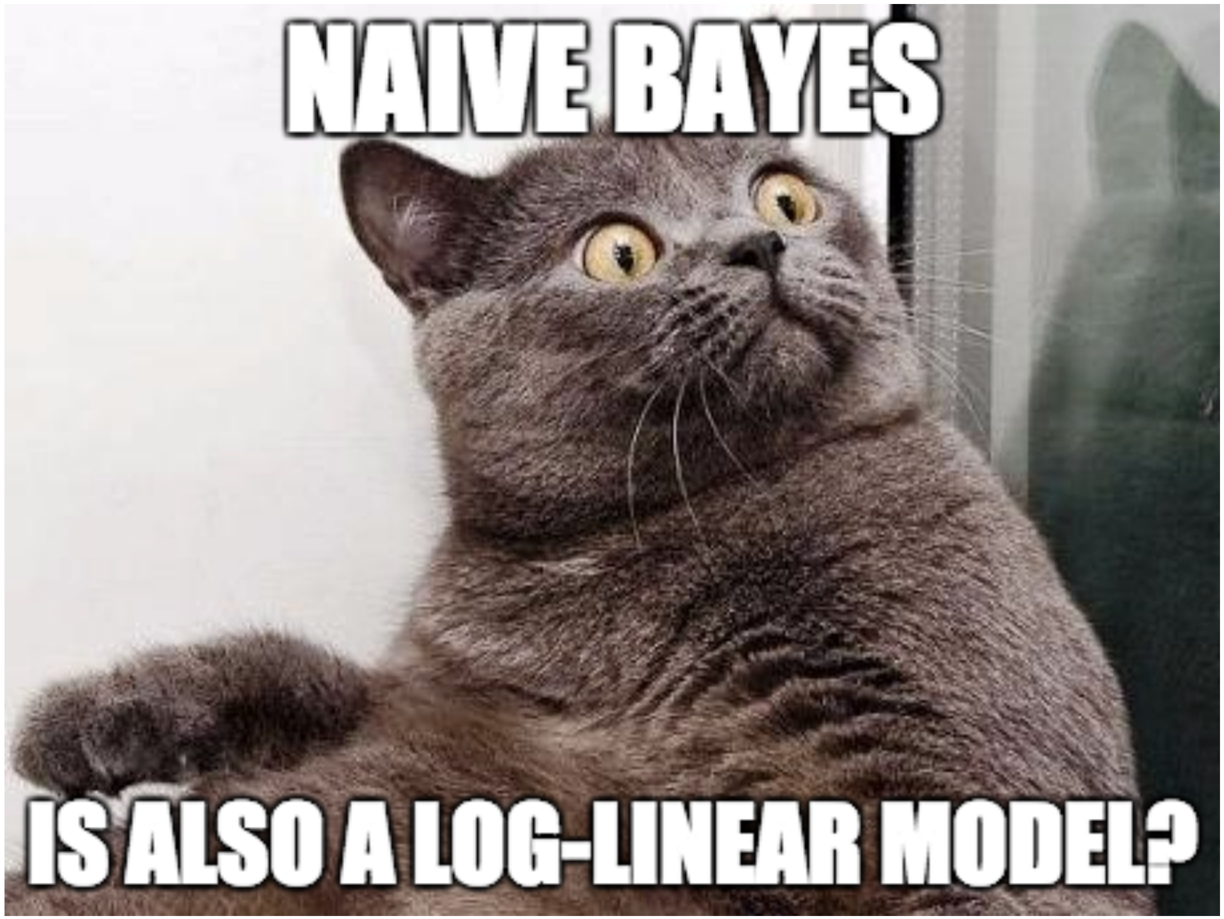
$$z = \sum_{i=0}^{|X|} w_i x_i$$

- Intuition: weighted sum of features
- All Linear models have this form

Log Linear Model

- A log-linear model is a mathematical model has the form a a function whose logarithm is a linear combination of the parameters.

$$\exp\left(\sum_{i=0}^{|X|} w_i x_i\right)$$


NAIVE BAYES
IS ALSO A LOG-LINEAR MODEL?

Naïve Bayes as a log-linear model

- Q: what are the features?

- Q: what are the weights?

Naïve Bayes as a Log-Linear Model

$$P(\text{spam}|D) \propto P(\text{spam}) \prod_{w \in D} P(w|\text{spam})$$

$$P(\text{spam}|D) \propto P(\text{spam}) \prod_{w \in \text{Vocab}} P(w|\text{spam})^{x_i}$$

$$\log P(\text{spam}|D) \propto \log P(\text{spam}) + \sum_{w \in \text{Vocab}} x_i \cdot \log P(w|\text{spam})$$

Naïve Bayes as a Log-Linear Model

$$\log P(\text{spam}|D) \propto \log P(\text{spam}) + \sum_{w \in \text{Vocab}} x_i \cdot \log P(w|\text{spam})$$

In both naïve Bayes and logistic regression we compute the dot product!

NB vs. LR

- Both compute the dot product
- NB: sum of log probabilities
- LR: logistic function

NB vs. LR: Parameter Learning

- Naïve Bayes:
  - Learn conditional probabilities **independently** by counting

- Logistic Regression:
  - Learn weights **jointly**

LR: Learning Weights

- Given: a set of feature vectors and labels
- Goal: learn the weights

LR: Learning Weights

$$\begin{array}{lcc}
 & \begin{matrix} \phantom{x_{11}} & \phantom{x_{12}} & \text{Feature} & & \phantom{x_{1n}} \end{matrix} & \text{Labels} \\
\text{Document} & \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & \cdots & x_{dn} \end{bmatrix} & \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}
\end{array}$$

Q: what parameters should we choose?

- What is the right value for the weights?

- Maximum Likelihood Principle:
  - Pick the parameters that maximize the probability of the y labels in the training data given the observations x.

Maximum Likelihood Estimation

$$w_{\mathrm{MLE}} = \operatorname{argmax}_w \log P(y_1, \ldots, y_d | x_1, \ldots, x_d; w)$$

$$= \operatorname{argmax}_w \sum_i \log P(y_i | x_i; w)$$

$$= \operatorname{argmax}_w \sum_i \log \begin{cases} p_i, & \text{if } y_i = 1 \\ 1 - p_i, & \text{if } y_i = 0 \end{cases}$$

logistic function

$$\boxed{p_i = \sigma(\textstyle\sum_j w_j x_j)}$$

$$= \operatorname{argmax}_w \sum_i \log p_i^{\mathbb{I}(y_i=1)} (1 - p_i)^{\mathbb{I}(y_i=0)}$$

Maximum Likelihood Estimation

$$= \operatorname{argmax}_w \sum_i \log p_i^{\mathbb{I}(y_i=1)} (1 - p_i)^{\mathbb{I}(y_i=0)}$$

$$= \operatorname{argmax}_w \sum_i y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

- Unfortunately there is no closed form solution
  - (like there was with naïve Bayes)

Closed Form Solution

- a Closed Form Solution is a simple solution that works instantly without any loops, functions etc
- e.g. the sum of integer from 1 to n

```
s= 0
for i in 1 to n
    s = s + i
end for
print s
```

Iterative Algorithm

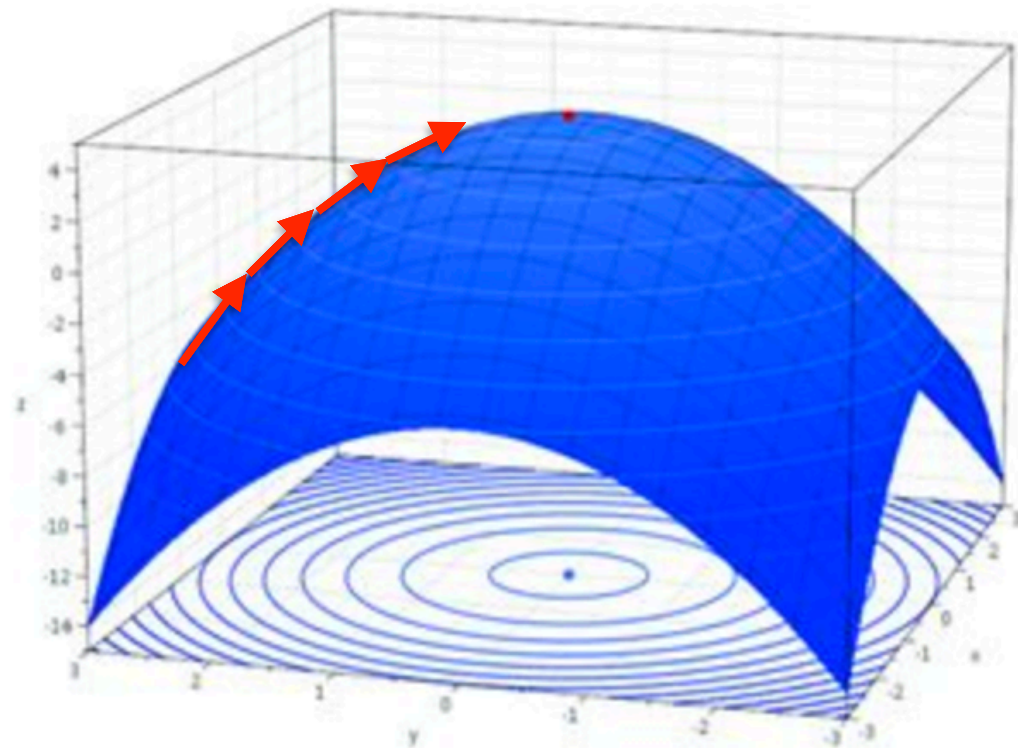
$$s = n(n+1)/2$$

Closed Form Solution

Maximum Likelihood Estimation

- Solution:
  - Iteratively climb the log-likelihood surface through the derivatives for each weight
- Luckily, the derivatives turn out to be nice

Gradient Ascent



Gradient Ascent

Loop While not converged:

For all features **j**, compute and add derivatives

$$w_j^{\text{new}} = w_j^{\text{old}} + \eta \frac{\partial}{\partial w_j} \mathcal{L}(w)$$

$\mathcal{L}(w)$: Training set log-likelihood

$$\left(\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \ldots, \frac{\partial \mathcal{L}}{\partial w_n} \right)$$: Gradient vector

Derivative Rules

Common Functions	Function	Derivative
Constant	c	0
Line	x	1
	ax	a
Square	x^2	2x
Square Root	√x	$(½)x^{-½}$
Exponential	e^x	e^x
	a^x	$\ln(a)\, a^x$
Logarithms	ln(x)	1/x
	$\log_a(x)$	1 / (x ln(a))

Rules	Function	Derivative
Multiplication by constant	cf	cf′
Power Rule	x^n	nx^{n-1}
Sum Rule	f + g	f′ + g′
Difference Rule	f - g	f′ − g′
Product Rule	fg	f g′ + f′ g
Quotient Rule	f/g	$(f' g - g' f)/g^2$
Reciprocal Rule	1/f	$-f'/f^2$
Chain Rule (as "Composition of Functions")	f º g	(f′ º g) × g′
Chain Rule (using ′)	f(g(x))	f′(g(x))g′(x)
Chain Rule (using $\frac{d}{dx}$)	$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$	

Derivative of Sigmoid

$$\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] \\
&= \frac{d}{dx}\left(1+\mathrm{e}^{-x}\right)^{-1} \\
&= -(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) \\
&= \sigma(x) \cdot (1-\sigma(x))
\end{aligned}$$

LR Gradient

$$w_{\text{MLE}} = \text{argmax}_w \underbrace{\sum_i y_i \log p_i + (1 - y_i) \log(1 - p_i)}_{\mathcal{L}}$$

logistic function

$$\boxed{p_i = \sigma(\textstyle\sum_j w_j x_j)}$$

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_i (y_i - p_i) x_j$$

Logistic Regression: Pros and Cons

- Doesn’t assume conditional independence of features
  - Better calibrated probabilities
  - Can handle highly correlated overlapping features

- NB is faster to train, less likely to overfit