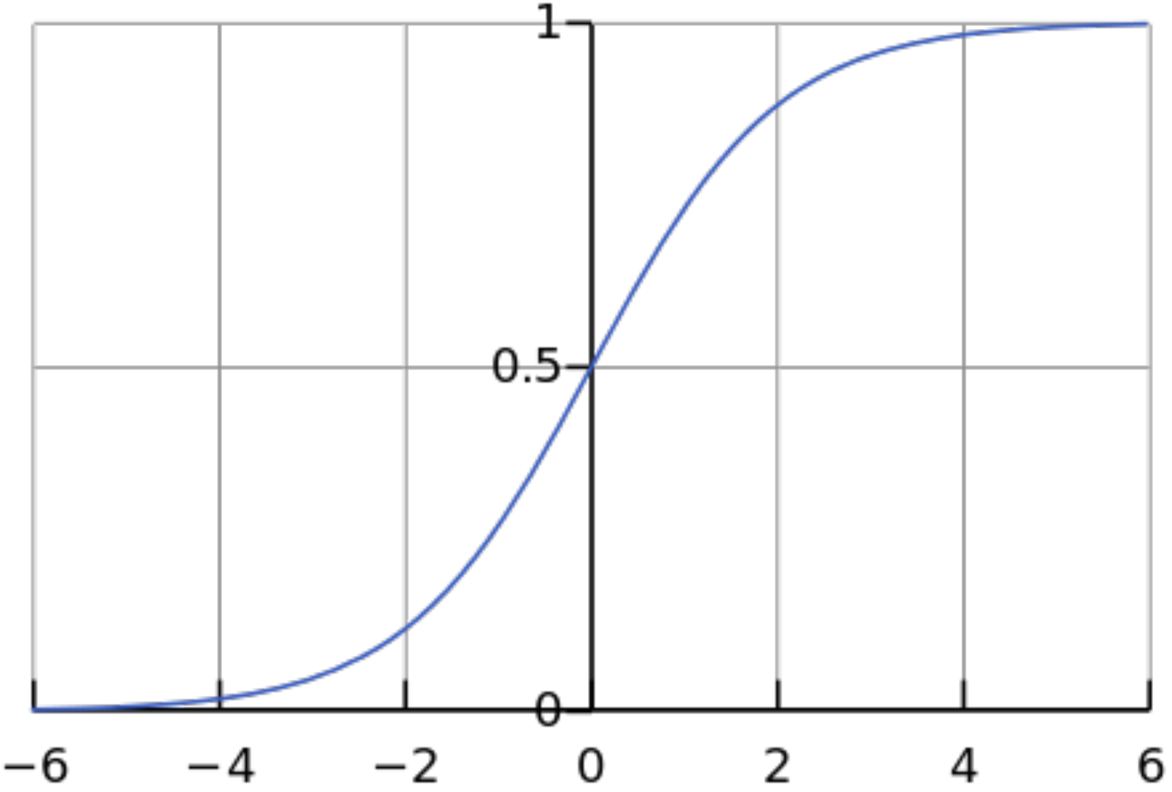
More Logistic Regression

Instructor: Wei Xu

Some slides adapted from Dan Jurfasky, Brendan O’Connor and Marine Carpuat

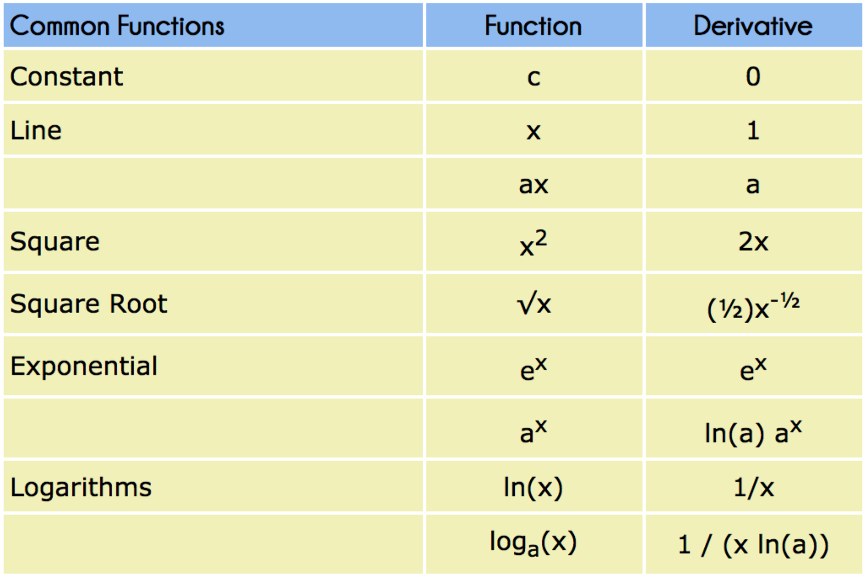
Warm Up

The Logistic function

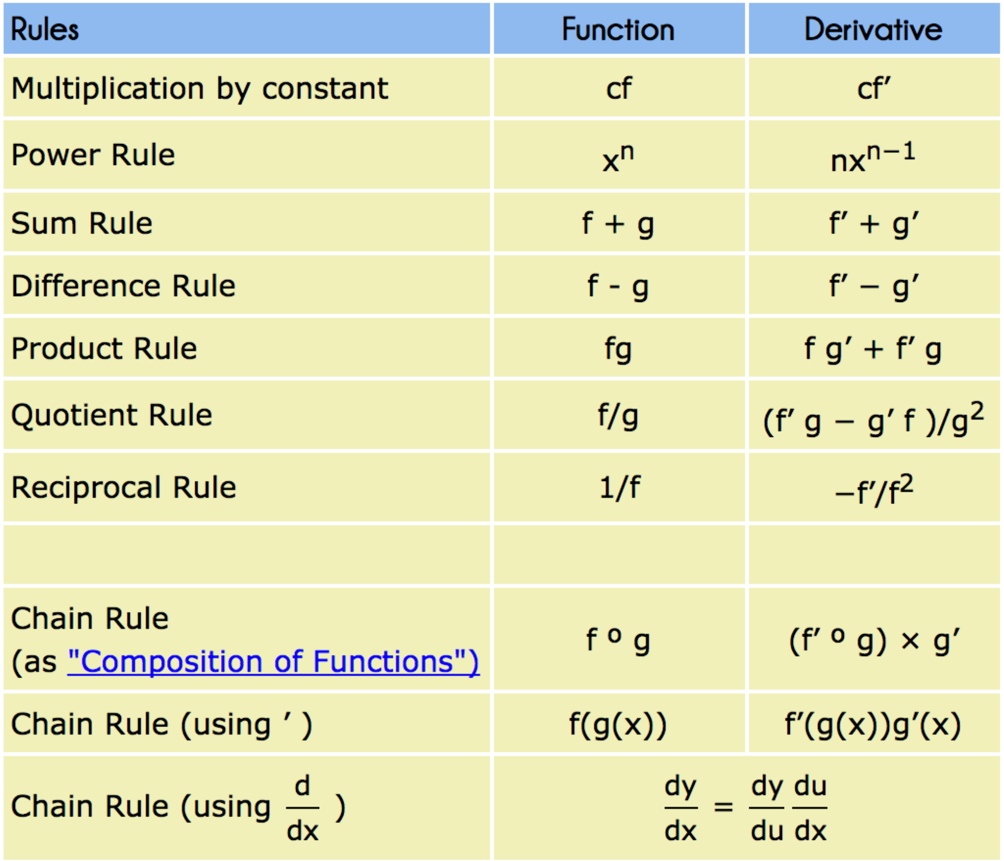


$$\sigma(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

Derivative Rules

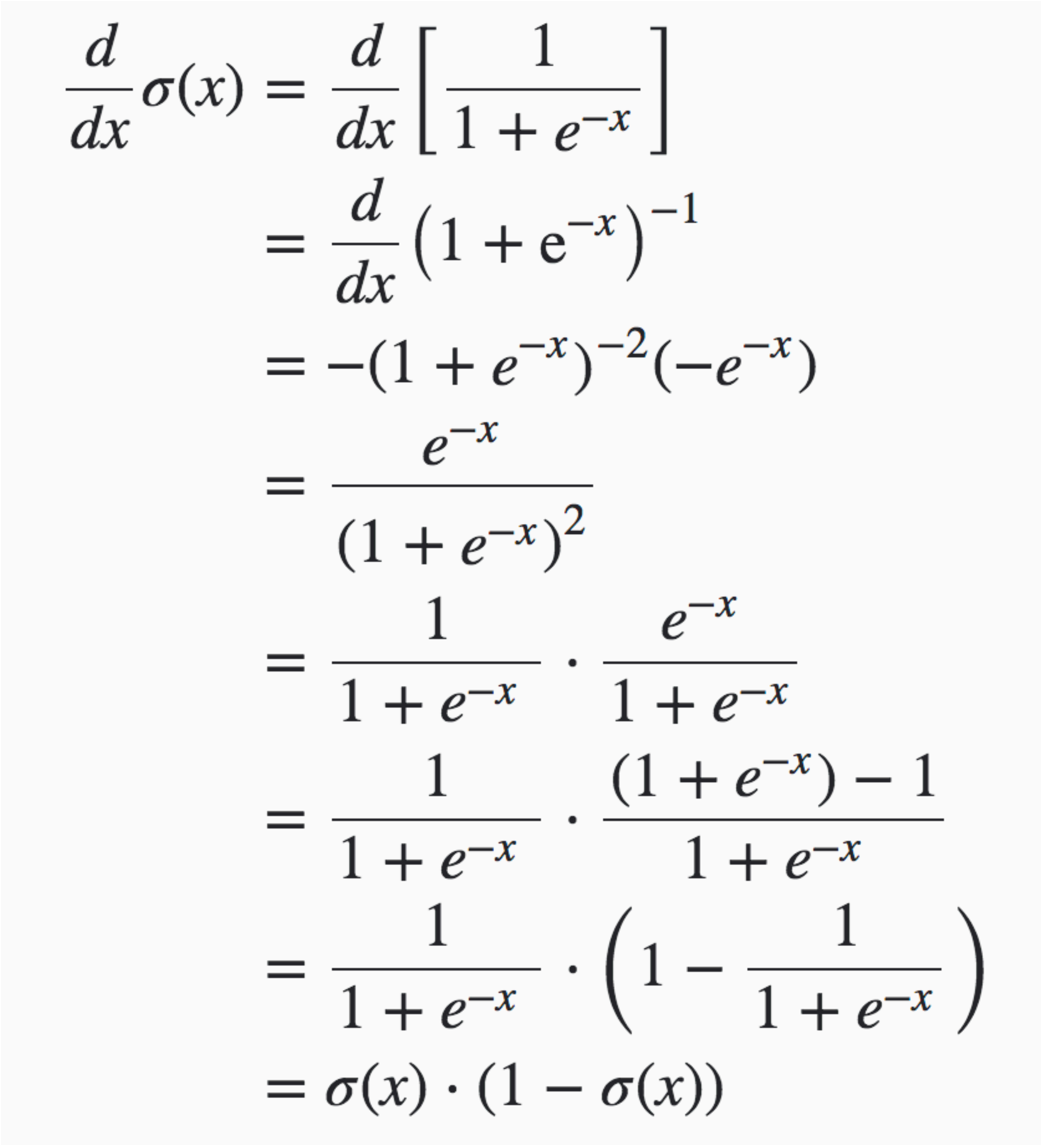


Common Functions	Function	Derivative
Constant	c	0
Line	x	1
	ax	a
Square	x^2	2x
Square Root	√x	$(½)x^{-½}$
Exponential	e^x	e^x
	a^x	$\ln(a)\ a^x$
Logarithms	ln(x)	1/x
	$\log_a(x)$	1 / (x ln(a))



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")	$f \circ g$	$(f' \circ g) \times g'$
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



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

NB & LR

- Both are linear models

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

- Training is different:
  - – NB: weights are trained independently
  - – LR: weights trained jointly

Linear Models

- 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:

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

Linear Models

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

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

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

Naïve Bayes as a Log-Linear Model

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

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

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

Logistic Regression

- (Log) Linear Model – similar to Naïve Bayes
- Doesn’t assume features are independent
- Correlated features don’t “double count”

Logistic Regression

- 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]

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

exponential/log space

NB vs. LR

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

NB vs. LR: Parameter Learning

- NB: Learn conditional probabilities **independently** by counting
- LR: Learn feature weights **jointly**

LR: Learning Weights

- Given: a set of feature vectors and labels

- Goal: learn the weights

LR: Learning Weights

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

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

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

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

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

logistic function

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

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

Maximum Likelihood Estimation

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

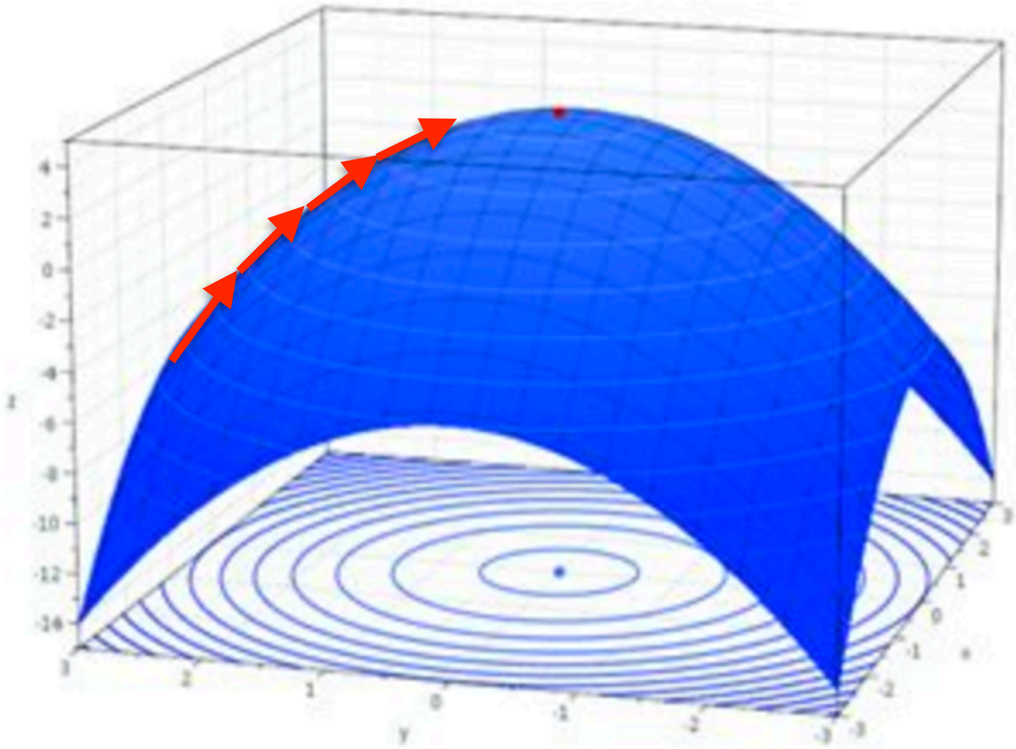
$$= \text{argmax}_w \sum_i y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

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

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

LR Gradient

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

logistic function

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

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

Exercise

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