Logistic Regression and Perceptron

Instructor: Wei Xu

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

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

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

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

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

features

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

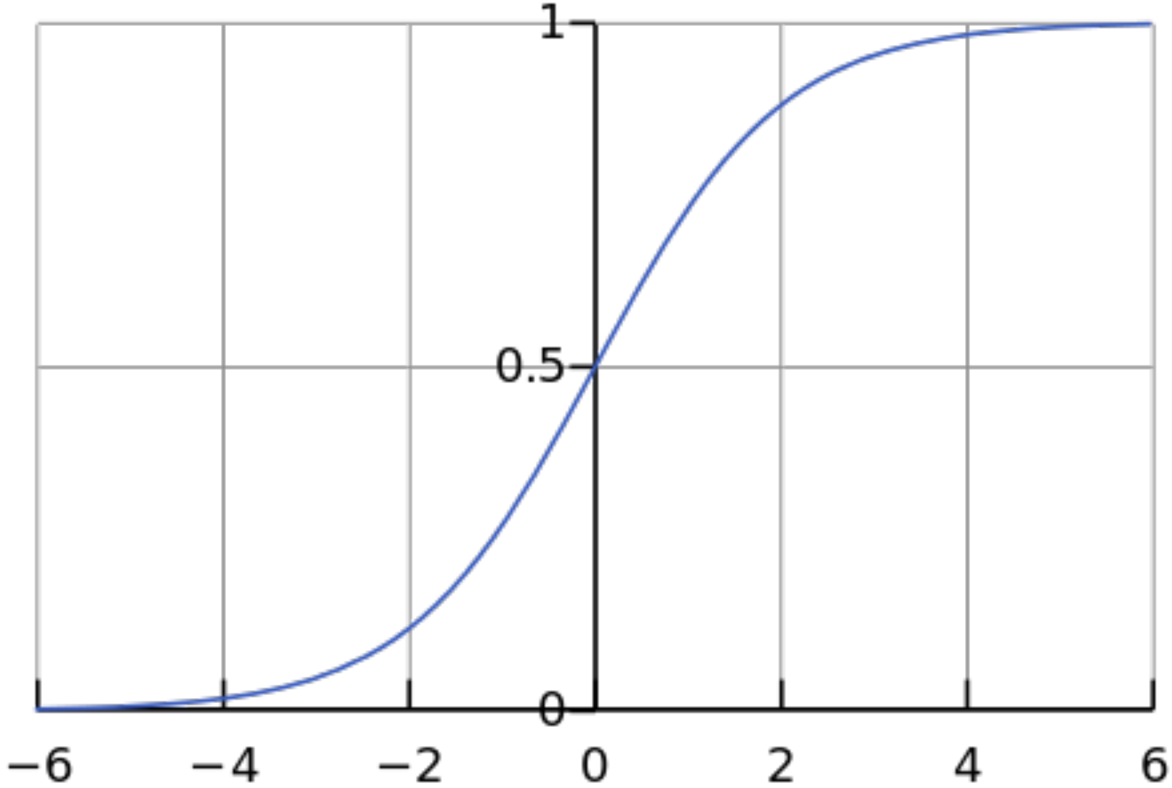
- Compute the logistic function:

convert into probabilities between [0, 1]

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

exponential/log space

The Logistic function



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

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

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

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

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

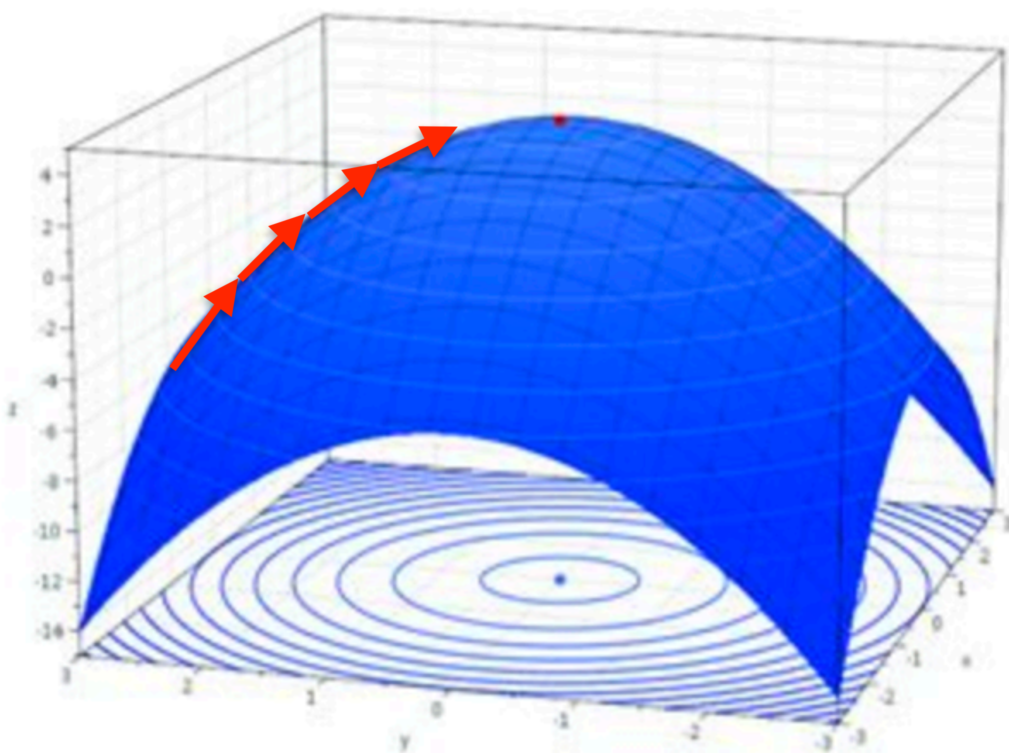
$$= \operatorname{argmax}_w \sum_i y_i \log p_i + (1-y_i) \log(1-p_i)$$

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

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

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

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

Exercise

Derivative of Sigmoid

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

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

MultiClass Classification

- Q: what if we have more than 2 categories?
  - Sentiment: Positive, Negative, Neutral
  - Document topics: Sports, Politics, Business, Entertainment, …

Q: How to easily do Multi-label classification?

Two Types of MultiClass Classification

- Multi-label Classification
  - each instance can be assigned more than one labels
- Multinominal Classification
  - each instance appears in exactly one class (classes are exclusive)

Multinominal Classification

- Pretty straightforward with Naive Bayes.

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

Log-Linear Models

$$P(y|x) \propto e^{w \cdot f(x,y)}$$

$$P(y|x) = \frac{1}{Z(w)} e^{w \cdot f(x,y)}$$

Multinominal Logistic Regression

$$P(y|x) \propto e^{w \cdot f(x,y)}$$

$$P(y|x) = \frac{1}{Z(w)} e^{w \cdot f(x,y)}$$

$$\boxed{P(y|x) = \frac{e^{w \cdot f(x,y)}}{\sum_{y' \in Y} e^{w \cdot f(x,y')}}}$$

Multinominal Logistic Regression

- Binary (two classes):
  - We have one feature vector that matches the size of the vocabulary
- Multi-class in practice:
  - one weight vector for each category

w_{pos} $\quad$ w_{neg} $\quad$ w_{neut}

Can represent this in practice with one giant weight vector and repeated features for each category.

Maximum Likelihood Estimation

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

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

$$= \operatorname{argmax}_w \sum_i \log \frac{e^{w \cdot f(x_i, y_i)}}{\sum_{y' \in Y} e^{w \cdot f(x_i, y')}}$$

(a.k.a) Softmax Regression

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Softmax function

From Wikipedia, the free encyclopedia

In mathematics, the **softmax function**, or **normalized exponential function**,[1]:198 is a generalization of the logistic function that "squashes" a K-dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\sigma(\mathbf{z})$ of real values in the range (0, 1) that add up to 1. The function is given by

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

(a.k.a) Maximum Entropy Classifier

- or MaxEnt

- Math proof of “LR=MaxEnt”:
  - [Klein and Manning 2003]
  - [Mount 2011]

http://www.win-vector.com/dfiles/LogisticRegressionMaxEnt.pdf

Multiclass LR Gradient

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^{D} f_j(y_i, d_i) - \sum_{i=1}^{D} \sum_{y \in Y} f_j(y, d_i) P(y|d_i)$$

Perceptron Algorithm

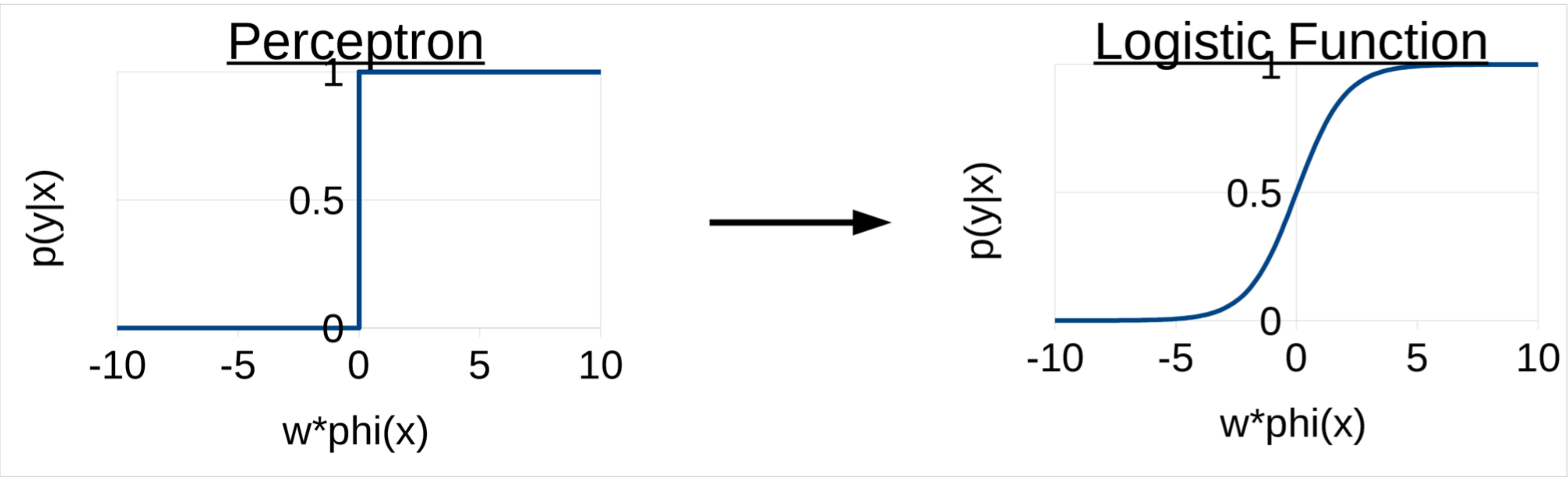
- Very similar to logistic regression
- Not exactly computing gradient

[Rosenblatt 1957]

http://www.peterasaro.org/writing/neural_networks.html

Perceptron Algorithm

- Very similar to logistic regression
- Not exactly computing gradient



$$P(y=1|x)=1 \text{ if } \mathbf{w}\cdot\boldsymbol{\varphi}(x)\geq 0$$
$$P(y=1|x)=0 \text{ if } \mathbf{w}\cdot\boldsymbol{\varphi}(x)<0$$

$$P(y=1|x)=\frac{e^{\mathbf{w}\cdot\boldsymbol{\varphi}(x)}}{1+e^{\mathbf{w}\cdot\boldsymbol{\varphi}(x)}}$$

Online Learning

- Update parameters for each training example (when predication is wrong)

```
for I iterations
    for each labeled pair x, y in the data
        phi = CREATE_FEATURES(x)
        y' = PREDICT_ONE(w, phi)
        if y' != y
            UPDATE_WEIGHTS(w, phi, y)
```

Online Learning

- The Perceptron is an online learning algorithm.
- Logistic Regression is not:

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

Perceptron Algorithm

- Very similar to logistic regression
- Not exactly computing gradient

```
Initalize weight vector w = 0
Loop for K iterations
    Loop For all training examples x_i
        if sign(w * x_i) != y_i
            w += (y_i - sign(w * x_i)) * x_i
```

Perceptron Notes

- Guaranteed to converge if the data is linearly separable

- Only hyperparameter is maximum number of iterations

- Parameter averaging will greatly improve performance

Differences between LR and Perceptron

- Online learning vs. Batch

- Perceptron doesn’t always make updates