Lecture 9 Logistic regression

10-17-2008

Review

- Training a Naïve Bayes classifier
  - $p(y)$, $p(x_i|y)$ for $i=1,\ldots, m$
- Predicting with Naïve Bayes Classifier
  - $p(y|X) = p(X|y)p(y)/p(X)$
  - predict y that maximizes $p(y|X)$
- Zero probabilities cause headache for Bayes classifiers
  - Laplace smoothing
- Generative vs. discriminative approaches
  - Naïve Bayes is a generative approach

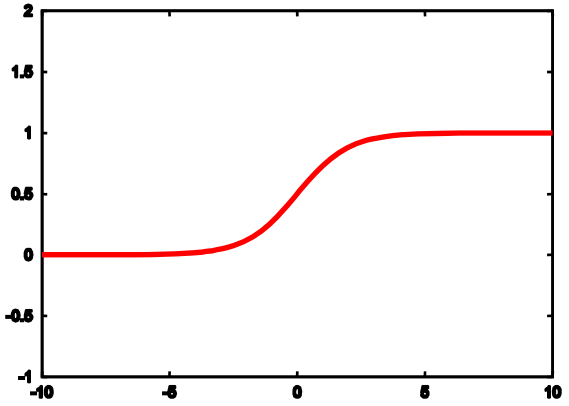
Logistic Regression

- Assume that the log odds of y=1 is a linear function of x :

$$\log \frac{P(y=1|x)}{P(y=0|x)} = w_0 + w_1 x_1 + \ldots + w_m x_m$$

- Or equivalently, we have:

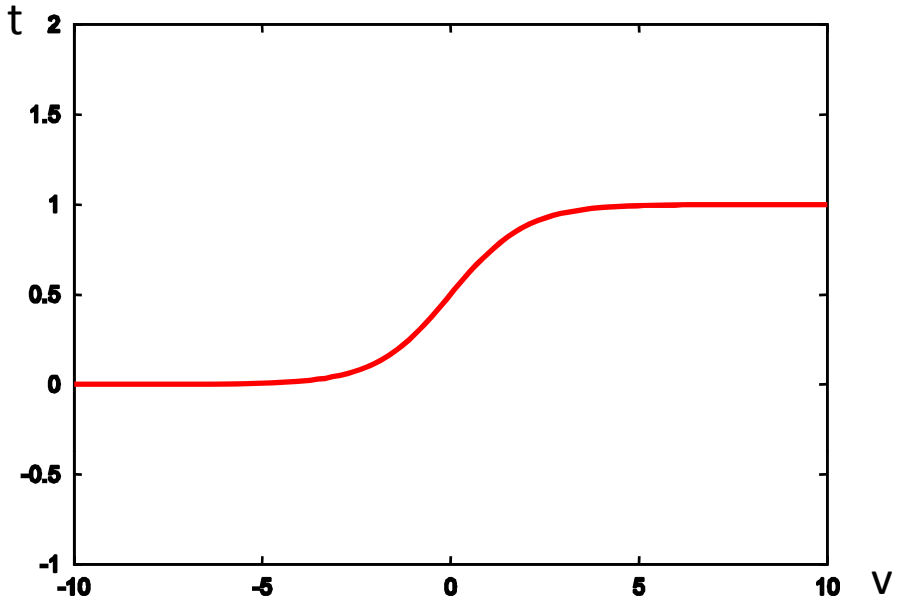
$$P(y=1|x) = \frac{1}{1+e^{-(w_0 + w_1 x_1 + \ldots + w_m x_m)}}$$



Sigmoid function

Side Note:
the odds in favor of an event are the quantity p / (1 − p), where p is the probability of the event
If I toss a fair dice, what are the odds that I will have a six?

Learning **w** for logistic regression



$$t = \frac{1}{1 + e^{-v}}$$

- Given a set of training data points, we would like to find a weight vector ***w*** such that $P(y = 1 \mid X) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \ldots + w_m x_m)}}$

  is large (e.g. 1) for positive training examples, and small (e.g. 0) otherwise

- In other words, a good weight vector W should satisfy the following:

  if we plot (v = $\mathbf{W} \cdot \mathbf{X}^i$, t=$y^i$), i=1, …, n, they should be in the area close to t=0 (for y^i=0) and t=1 (for y^i=1)

Learning **w** for logistic regression

- This can be captured in the following objective function:

$$L(\mathbf{w}) = \sum_i \log P(y^i \mid \mathbf{x}^i, \mathbf{w})$$
$$= \sum_i [y^i \log P(y^i = 1 \mid \mathbf{x}^i, \mathbf{w}) + (1 - y^i) \log(1 - P(y^i = 1 \mid \mathbf{x}^i, \mathbf{w}))]$$

Note that the superscript *i* is an index to the examples in the training set

This is call the likelihood function of **w,** and by maximizing this objective function, we perform what we call "maximum likelihood estimation" of the parameter **w.**

Maximum Likelihood Estimation

Goal: estimate the parameters given data

Assuming the data is i.i.d (identically independently distributed)

For example, given the results of n coin tosses, we like to estimate the probability of head p.

Likelihood function:

$$L(\theta) = \log P(D \mid \theta) = \log \prod_{i=1}^{n} P(x^i, y^i \mid \theta) = \sum_{i=1}^{n} \log P(x^i, y^i \mid \theta)$$

MLE estimator:

$$\theta_{MLE} = \arg\max_{\theta} L(\theta)$$

Example

- Data: n iid coin toss: D={0, 0, 1, 0,…1}
- Parameter $\theta = P(x = 1)$
- Binary distribution $P(x) = \theta^x (1-\theta)^{1-x}$
- Likelihood function?

- MLE estimate?

Example

- Data: n iid coin toss: D={0, 0, 1, 0,…1}
- Parameter $\theta = P(x=1)$
- Binary distribution $P(x) = \theta^x (1-\theta)^{1-x}$
- Likelihood function?

$$L(\theta) = \log \theta^{n_1} (1-\theta)^{n_0} = n_1 \log \theta + n_0 \log(1-\theta)$$

- MLE estimate?

$$\frac{dL}{d\theta} = \frac{n_1}{\theta} - \frac{n_0}{(1-\theta)} = 0 \Rightarrow$$

$$\frac{n_1}{\theta} = \frac{n_0}{(1-\theta)} \Rightarrow n_1(1-\theta) = n_0\theta \Rightarrow$$

$$n_1 = n_1\theta + n_0\theta \Rightarrow \theta = \frac{n_1}{n_1 + n_0}$$

MLE for logistic regression

$$L(\theta) = \log P(D \mid W) = \log \prod_{i=1}^{n} P(X^i, y^i \mid W) = \sum_{i=1}^{n} \log P(X^i, y^i \mid W)$$

$$= \sum_{i=1}^{n} \log(P(y^i \mid W, X^i) P(X^i \mid W)) = \sum_{i=1}^{n} \log P(y^i \mid W, X^i) + \sum_{i=1}^{n} P(X^i \mid W)$$

$$W_{MLE} = \arg\max_W L(W) = \arg\max_W \sum_{i=1}^{n} \log P(y^i \mid W, X^i)$$

$$= \arg\max_W \sum_{i=1}^{n} \log(y^i P(y^i = 1 \mid W, X^i) + (1 - y^i)(1 - P(y^i = 1 \mid W, X^i)))$$

Equivalently, given a set of training data points, we would like to find a weight vector ***w*** such that $P(y = 1 \mid W, X)$ is large (e.g. 1) for positive training examples, and small (e.g. 0) otherwise – the same as our intuition

Optimizing L(w)

- Unfortunately this does not have a close form solution
- Instead, we iteratively search for the optimal w
- Start with a random w, iteratively improve w (similar to Perceptron)

Logistic regression learning

Given : training examples $(\mathbf{x}^i, y^i),\ i = 1,\ldots,N$

Let $\mathbf{w} \leftarrow (0,0,0,\ldots,0)$

Repeat until convergence

$\mathbf{d} \leftarrow (0,0,0,\ldots,0)$

For $i = 1$ to N do

$$\hat{y} \leftarrow \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}^i}}$$

$$error = y^i - \hat{y}$$

$$\mathbf{d} = \mathbf{d} + error \cdot \mathbf{x}^i$$

$$\mathbf{w} \leftarrow \mathbf{w} + \eta d$$

Learning rate

Batch Learning for Logistic Regression

Given : training examples $(\mathbf{x}^i, y^i)$, $i = 1,\ldots,N$

Let $\mathbf{w} \leftarrow (0,0,0,\ldots,0)$

Repeat until convergence

$d \leftarrow (0,0,0,\ldots,0)$

For $i = 1$ **to** N **do**

$$\hat{y}^i \leftarrow \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}^i}}$$

$$error = y^i - \hat{y}^i$$

$$d = d + error \cdot \mathbf{x}^i$$

$$\mathbf{w} \leftarrow \mathbf{w} + \eta d$$

Note: y takes 0/1 here, not 1/-1

Logistic Regression Vs. Perceptron

- Note the striking similarity between the two algorithms
- In fact LR learns a linear decision boundary – how so?
  - We can show that mathematically, see board
- What are the difference?
  - Different ways to train the weights
  - LR produces a probability estimation
  - (a maybe not so interesting difference)LR by statistician and Perceptron by CS

There are more!

- If we assume Gaussian distribution for $p(x_i|y)$ in Naïve Bayes, $p(y=1|X)$ will take the same functional form of Logistic Regression
- What are the differences here?
  - Different ways of training
    - Naïve bayes estimates θ_i by maximizing $P(X|y=v_i, \theta_i)$, and while doing so assumes conditional independence among attributes
    - Logistic regression estimates **w** by maximizing $P(y|x, \mathbf{w})$ and make no conditional independence assumption.

Comparatively

- Naïve Bayes - generative model: P(X|y)
  - makes strong conditional independence assumption about the data attributes
  - When the assumptions are ok, naïve bayes can use small amount of training data and estimate a reasonable model
- Logistic regression-discriminative model: directly learn p(y|X)
  - has fewer parameters to estimate, but they are tied together and make learning harder
  - Makes no strong assumptions
  - May need large number of training examples

Bottom line: if the naïve bayes assumption holds and the probabilistic models are accurate (i.e., x is gaussian given y etc.), NB would be a good choice; otherwise, logistic regression works better

Summary

- We introduced the concept of generative vs. discriminative method
  - Given a method that we discussed in class, you need to know which category it belongs to
- Logistic regression
  - Assumes that the log odds of y=1 is a linear function of X (i.e., $W \cdot X$)
  - Learning goal is to learn a weight vector W such that examples with y=1 are predicted to have high $P(y=1|X)$ and vice versa
    - Maximum likelihood estimation is a approach that achieves this
    - Iterative algorithm to learn W using MLE
    - Similarity and difference between LR and perceptrons
  - Logistic regression learns a linear decision boundarys