Logistic regression

CS434
Bayes (Naïve or not) Classifiers: *Generative Approach*

- What do we mean by *Generative approach*:
  - Assuming that each data point is generated following a *generative process* governed by \( p(y) \) and \( p(X|y) \)
  - Learn \( p(y) \), \( p(x|y) \) and then apply Bayes rule to compute \( p(y|x) \) for making predictions
• Generative approach is just one type of learning approaches used in machine learning
  – Learning a correct generative model $p(x|y)$ is difficult – density estimation is a challenging problem in its own
  – And sometimes unnecessary

• In contrast, LTU, KNN and DT are what we call **discriminative methods**
  – They are not concerned about any generative models
  – They only care about finding a good discriminative function
  – LTU, KNN and DT learn deterministic functions, not probabilistic

• One can also take a probabilistic approach to learning discriminative functions
  – i.e., Learn $p(y|x)$ directly without learning $p(x|y)$
  – **Logistic regression** is one such approach
Logistic regression

- Recall the problem of regression
- Learns a mapping from input vector $\mathbf{x}$ to a continuous output $y$
- Logistic regression extends traditional regression to handle binary output $y$
- In particular, we assume that

\[
P(y = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \ldots + w_m x_m)}} = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}} = g(\mathbf{w}^T \mathbf{x})
\]
Logistic Regression

• Equivalently, we have the following:

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

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?

• In other words, LR assumes that the log odds is a linear function of the input features
Learning $\mathbf{w}$ for logistic regression

• Given a set of training data points, we would like to find a weight vector $\mathbf{w}$ such that

$$P(y = 1 | \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

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

• In other words, a good weight vector $\mathbf{w}$ should satisfy the following:

$\mathbf{w} \mathbf{x}$ should be large negative values for - points

$\mathbf{w} \mathbf{x}$ should be large positive value for + points
Learning $\mathbf{w}$ for logistic regression

• This can be captured by the log likelihood function:

$$L(w) = \sum \log P(y^i | x^i, w)$$

$$= \sum \left[ y^i \log P(y^i = 1 | x^i, w) + (1 - y^i) \log(1 - P(y^i = 1 | x^i, w)) \right]$$

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

This is call the likelihood function of $\mathbf{w}$, and by maximizing this objective function, we perform what we call “maximum likelihood estimation” of the parameter $\mathbf{w}$.
MLE for logistic regression

\[ l(w) = \log \prod_{i=1}^{n} P(y^i \mid x^i, w) = \sum_{i=1}^{n} \log P(y^i \mid x^i, w) \]

\[ w_{MLE} = \arg \max_w l(w) = \arg \max_w \sum_{i=1}^{n} \log P(y^i \mid x^i, w) \]

\[ = \arg \max_w \sum_{i=1}^{n} \left[ y^i \log P(y^i = 1 \mid x^i, w) + (1 - y^i) \log(1 - P(y^i = 1 \mid x^i, w)) \right] \]

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 $\ell(w)$

- 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) by moving toward the gradient direction (the fastest increasing direction) - $\nabla \ell$
Gradient Descend/Ascend Example

\[ F(x, y) = \sin \left( \frac{1}{2} x^2 - \frac{1}{4} y^2 + 3 \right) \cos(2x + 1 - e^y) \]

Start from a random initial point
Iteratively move toward the direction that improves the objective at maximal rate
Stop when reaching local optimal point (\( \nabla = 0 \))
Batch Learning for Logistic Regression

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

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

Let \(w \leftarrow (0,0,0,\ldots,0)\)

Repeat until convergence

\[
d \leftarrow (0,0,0,\ldots,0)
\]

For \(i = 1\) to \(N\) do

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

\[
error = y^i - \hat{y}^i
\]

\[
d = d + error \cdot x^i
\]

\[
w \leftarrow w + \eta d
\]

Gradient contribution from the \(i\)th example

Learning rate
Logistic Regression Vs. Perceptron

• Note the striking similarity between the two algorithms
• In fact LR learns a linear decision boundary – how so?
  – Home work assignment
• What are the difference?
  – Different ways to train the weights
  – LR produces a probability estimation
Logistic Regression vs. Naïve Bayes

• If we use Naïve Bayes and assume Gaussian distribution for $p(x_i|y)$, we can show that $p(y=1|X)$ takes the exact same functional form of Logistic Regression.

• What are the differences here?
  – Different ways of training
    • Naïve bayes estimates $\theta_i$ by maximizing $P(X|y=v_i, \theta_i)$, and while doing so assumes conditional independence among attributes.
    • Logistic regression estimates $w$ by maximizing $P(y|x, w)$ 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 a 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 weaker 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 boundaries