CHAPTER 10:

Logistic Regression

Binary classification

- Two classes Y = {0,1}
- Goal is to learn how to correctly classify the input into one of these two classes
  - Class 0 – labeled as 0
  - Class 1 – labeled as 1
- We would like to learn $f : X \rightarrow \{0,1\}$
  - Since the output must be 0 or 1, we cannot directly use an unlimited linear model to estimate $f(x_i)$.

- Very much like a simple neuron with a sigmoid activation function, we will let:

$$f(\mathbf{x}) = g(\mathbf{w}^{\mathrm{T}}\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^{T}\mathbf{x}}}$$

where **w** forms the weights to be determined and **x** is the input vector.

- $g(z) = \dfrac{1}{1+e^{-z}}$ is the logistic function (also called sigmoid)

- We would also like to interpret f(**x**) as P(y=1|x)

$$P(y=1 \mid \mathbf{x}) = \frac{1}{1+e^{-(w_0+w_1x_1+\ldots+w_mx_m)}}$$

- In fact, we learn the log odds of P(y=1|x) as a linear function of the input variables:

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

Odds of y=1

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?

- If $f(x) = P(y=1|x) > 0.5$ then choose class C1
- Otherwise choose class C0

- Linear decision boundary.

Learning w for logistic regression

- We can use Maximum Likelihood Estimation to find the parameters (w) that maximizes (log) likelihood of the class labels in the training data.

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

where the superscript i is an index to the examples in the training set.

- No closed form => Iterative solution.
  - Iteratively reweighted least squares
  - Gradient descent

- Let p_i=Pr(y_i=1|x_i). We would like to estimate pi as f(x).
- We model the log odds of the probability p_i as:

  $\ln (p_i/(1-p_i)) = \beta.x_i$ where $\ln (p_i/(1-p_i))=g(p_i)$ is the logit function

- By applying the *inverse of logit* (the logistic function), we get back p_i:

  $$\text{logit}^{-1} [\ln (p_i/(1-p_i))] = p_i$$

- Applying it on the RHS as well, we get

  $$p_i = \text{logit}^{-1} (\beta.x_i) = 1 / (1 + e^{-\beta.xi})$$

Logistic Regression vs Perceptron

- Logistic Regression learns a linear decision boundary like the perceptron
  - What is the decision boundary?

- Logistic Regression is trained to produce probability estimations.

Logistic Regression vs Naive 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 θ_i by maximizing $P(X|y=v_i, \theta_i)$, and while doing so assumes conditional independence among attributes
    - Logistic regression estimates $\mathbf{w}$ by maximizing $P(y|x, \mathbf{w})$ and make no conditional independence assumption.

Logistic Regression vs Naive Bayes

- 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