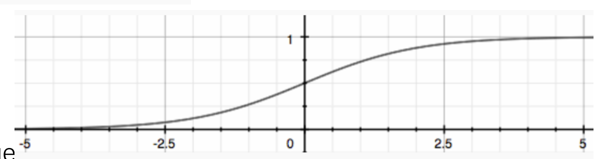
machine learning_week3

Notebook:	artificial intelligence		
Created:	18/11/08 06:04	**Updated:**	18/11/09 22:22
Author:	2440027575@qq.com		
URL:	https://www.coursera.org/learn/machine-learning/supplement/fDCQp/classification		

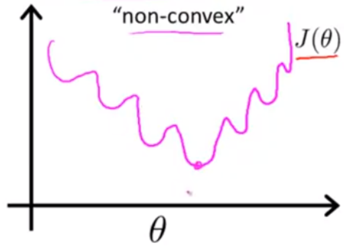
week_3 logistic regression

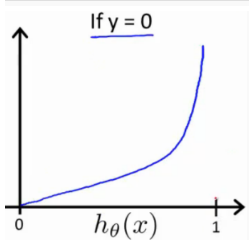
classification and representation

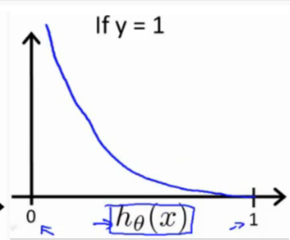
- **Introduction**
  - **task**
    - facing discrete data, we usually need to make a **classification.**
    - begin with **binary classification problem** 👉 further **multiclass**
    - Hence, y∈{0,1}. 0 is also called the negative class, and 1 the positive class
  - method
    - linear regression doesn't work
    - use **logistic regression**
- **hypothesis representation**
  - **Sigmoid function (Logistic function)**
    - $$h_\theta(x) = g(\theta^T x)$$
      $$z = \theta^T x$$
      $$g(z) = \frac{1}{1+e^{-z}}$$
    - image 
  - hθ(x) give us the **probability** that our output is 1. Just interpret it in **Conditional probability.**
    - $$h_\theta(x) = P(y=1|x;\theta) = 1 - P(y=0|x;\theta)$$
      $$P(y=0|x;\theta) + P(y=1|x;\theta) = 1$$
  - when can we get hθ(x)>0.5? studying the hypothesis closer.
    - we got this: $$\theta^T x \geq 0 \Rightarrow y = 1$$
      $$\theta^T x < 0 \Rightarrow y = 0$$
    - and take the term: **Decision Boundary**
  - **Decision Boudary**
    - the line that separates the area where y = 0 and where y = 1.
    - the property of hθ(x), once the theta decided, we got decision boundary. It doesn't come from data set.
    - **Image!** remember that decision boundary comes from $\theta^T X$, it can be linear or more complicated base on what polynomial function u chosen.

Logistic Regression Model

- **Cost Function**
  - review:
    - **what cost function work.** Cost function is to measure the accuracy of hypothesis. So if our hypothesis comes up with the output that **close to** the actual label, cost function should give value close to 0; while hypothesis gives wrong output, cost function should **penalize** it. That is give high value.
    - **how to use cost function to get better theta: Gradient Descent.** When we use GD, we want the cost function *has global optimum* and better *not have local optimum* so that it can converge to the theta we want.

- **convex:**
  - non-convex: use the same cost function in linear regression we would get many local optimum. That's not we want. Plot below:
  - 

- we use:
  - $$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
    $$\mathrm{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if y = 1}$$
    $$\mathrm{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if y = 0}$$
  - how it work
    - $$\mathrm{Cost}(h_\theta(x), y) = 0 \text{ if } h_\theta(x) = y$$
      $$\mathrm{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 0 \text{ and } h_\theta(x) \to 1$$
      $$\mathrm{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 1 \text{ and } h_\theta(x) \to 0$$
    - 



- **compress it:**
  - $$h = g(X\theta)$$
    $$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1-y)^T \log(1-h)\right)$$

- ## Gradient Descent
  - $$\begin{aligned} & Repeat \; \{ \\ & \quad \theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \\ & \} \end{aligned}$$
  - vectored version: $$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$$
  - further: how to make partial derivative?
    - link: https://math.stackexchange.com/questions/477207/derivative-of-cost-function-for-logistic-regression

The reason is the following. We use the notation

$$\theta x^i := \theta_0 + \theta_1 x_1^i + \cdots + \theta_p x_p^i.$$

Then

$$\log h_\theta(x^i) = \log \frac{1}{1+e^{-\theta x^i}} = -\log(1+e^{-\theta x^i}),$$

$$\log(1-h_\theta(x^i)) = \log(1-\frac{1}{1+e^{-\theta x^i}}) = \log(e^{-\theta x^i}) - \log(1+e^{-\theta x^i}) = -\theta x^i - \log(1+e^{-\theta x^i}),$$

[this used: $1 = \frac{(1+e^{-\theta x^i})}{(1+e^{-\theta x^i})}$, the 1's in numerator cancel, then we used: $\log(x/y) = \log(x) - \log(y)$]

The reason is the following. We use the notation

$$\theta x^i := \theta_0 + \theta_1 x_1^i + \cdots + \theta_p x_p^i.$$

Then

$$\log h_\theta(x^i) = \log \frac{1}{1+e^{-\theta x^i}} = -\log(1+e^{-\theta x^i}),$$

$$\log(1-h_\theta(x^i)) = \log(1-\frac{1}{1+e^{-\theta x^i}}) = \log(e^{-\theta x^i}) - \log(1+e^{-\theta x^i}) = -\theta x^i - \log(1+e^{-\theta x^i}),$$

[this used: $1 = \frac{(1+e^{-\theta x^i})}{(1+e^{-\theta x^i})}$, the 1's in numerator cancel, then we used: $\log(x/y) = \log(x) - \log(y)$]

- ## Advanced Optimization
  - require: Advanced Mathematics. Know how to make **partial derivative.**
  - "Conjugate gradient", "BFGS", and "L-BFGS" are more sophisticated, faster ways to optimize θ
  - how: use the library.

```
% write a single function that returns both of these

function [jVal, gradient] = costFunction(theta)
  jVal = [...code to compute J(theta)...];
  gradient = [...code to compute derivative of J(theta)...];
end

% give to the function "fminunc()" our cost function
% for more details, refer to official document.
```

- ## Multi-class Classification: One vs All
  - Supposed we have N class. Then train a logistic regression classifier hθ(X) for each class to predict the probability that y = i . That's say now we have N classifier. Each classifier can predict whether the sample is class(j) or not.
  - To make a prediction on a new x, pick the class that **maximizes** hθ(X)

Regularization

- ## over-fitting/under-fitting
  - hypothesis fit training set perfectly but fail to predict / map poorly to the trend of data set.
  - solution:
    - **Reduce features**
      1. manually select features to remain
      2. use a model selection algorithm
    - **Regularization**
- ## modify cost function
  - add regularization parameter λ. It's a **penalty** for theta. It determines how much the costs of our theta parameters are inflated.

- $$min_\theta \ \frac{1}{2m} \ \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \boxed{\lambda} \sum_{j=1}^n \theta_j^2$$
- Proper lambda can make hypothesis smoother so that avoid over-fitting. But
  - too large: penalize too strong that make theta close to 0. Finally we may get a flat line (assumed that θ0 not been regularized) .
  - too small: help little to fix over-fitting.
- for linear regression:
  - new GD here:
    - $$\begin{aligned} & \text{Repeat } \{ \\ & \quad \theta_0 := \theta_0 - \alpha \ \frac{1}{m} \ \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)} \\ & \quad \theta_j := \theta_j - \alpha \ \left[\left(\frac{1}{m} \ \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \right) + \frac{\lambda}{m}\theta_j \right] \qquad\qquad j \in \{1, 2...n\} \\ & \} \end{aligned}$$
    - separate out θ_0 from the rest of the parameters because we do not want to penalize θ_0
    - and merge θ: $$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$
  - new normal equation:
    - $$\theta = \left(X^TX + \lambda \cdot L \right)^{-1} X^Ty$$ $$\text{where } \ L = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$
    - benefit: add the term λ·L, then X^TX + λ·L becomes always **invertible**.
      - when X_{m*n}, has m<n, the X^TX would be non-invertible.
      - because:
        - a invertible matrix is full rank.
        - $R(X_{m*n})$<=min(m,n)=m
        - X^TX is n*n matrix, which full rank should equal to n.
        - further explanation:
          - Suppose matrix A is of size vxw. Then rank(A)<=min(v,w).
          - Also suppose matrices A and B with rank(A)=a and rank(B)=b. Then rank(A*B)=min(a,b).

          Remember X is a mxn - matrix, now with m<n.

          Multiplying out X'*X is of dimension nxn, but rank(X'*X)<=m<n.

          Invertibility of a matrix requires full rank, which is by now not given. A square matrix that has full rank is also called **regular**, which is maybe why the name **regularization** arises.
- also work for logistic regression:
  - new cost function: (just add θ_j^2 at the end)
    - $$J(\theta) = -\frac{1}{m}\sum_{i=1}^m [y^{(i)} \ \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \ \log(1 - h_\theta(x^{(i)}))] + \boxed{\frac{\lambda}{2m}\sum_{j=1}^n \theta_j^2}$$
  - by solving partial derivative : new GD

Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \underbrace{\left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \boxed{\frac{\lambda}{m} \theta_j} \right]}_{\frac{\partial}{\partial \theta_j} J(\theta)}$$

$$(j = \cancel{0}, \underbrace{1, 2, 3, \ldots, n}_{\theta_1 \ldots \theta_n})$$

}

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

- remember to separate θ0 out.