Machine Learning

2014/12/16

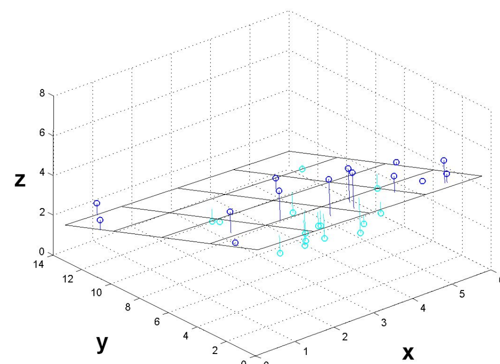
By 謝德威 (tewei)

安裝順序

1. Python 2.7
2. $python get-pip.py
3. $ pip install ipython, pyzmq, tornado, jinja2, numpy, matplotlib
4. $ipython notebook

Contents

1. Linear Basis Function Models



1.1. Linear Regression

$y(x, w) = w_0 + w_1 x_1 + ... + w_D x_D$

where $x = (x_1, ..., x_D)^T$

當然很多時候model沒有那麼簡單

1.2. Linear Combinations of Fixed Nonlinear Functions

$y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j \varphi_j(x)$ 把常數加進去

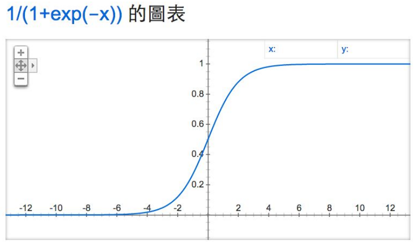
$y(x, w) = \sum_{j=0}^{M-1} w_j \varphi_j(x) = w^T \varphi(x)$

where $w = (w_0, ..., w_{M-1})^T$ and $\varphi = \varphi(\varphi_0, ..., \varphi_{M-1})^T$

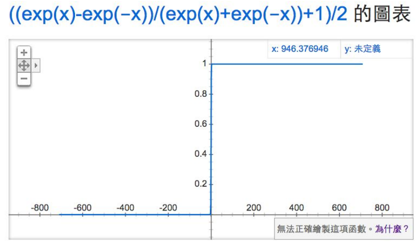
1.3. Examples of Basis Function

- ❏ Polynomial Function $\varphi_j(x) = x^j$
- ❏ Logistic Function $\varphi_j(x) = \sigma(\frac{x-\mu_j}{s})$, $\sigma(k)$ is a logistic (邏輯) sigmoid (S型) function $\sigma(k) = \frac{1}{1+exp(-k)}$ or $\sigma(k) = \frac{tanh(k)+1}{2}$

1/(1+exp(−x)) 的圖表



((exp(x)-exp(−x))/(exp(x)+exp(−x))+1)/2 的圖表



2. Maximum Likelihood

2.1. Sum-of-Squares Error Function

$E_D(w) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\varphi(x_n)\}^2$, where t is target variable

把它微分一下得到

$\sum_{n=1}^{N}\{t_n - w^T\varphi(x_n)\}\varphi(x_n)^T$

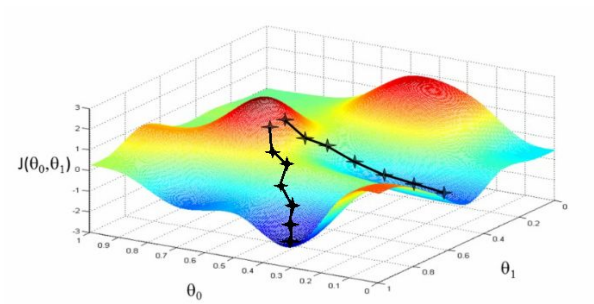
因為這個函數是convex，微分=0可得最小值

$\sum_{n=1}^{N} t_n\varphi(x_n)^T - w^T\left(\sum_{n=1}^{N}\varphi(x_n)\varphi(x_n)^T\right) = 0$

Φ是一個$N \times M$的矩陣，$\Phi_{nj} = \varphi_j(x_n)$

所求的$w = (\Phi^T\Phi)^{-1}\Phi^T$，是$\Phi$的Pseudo-Inverse

3. Gradient Descent



3.1. Optimization Problem

好Pseudo-Inverse不用嗎？如果沒辦法直接解呢？Gradient Descent是個最常見的最佳化演算法，可以求得一個函數的區域極大、極小值

假設有個cost function $J(\theta)$，我們想知道$J(\theta)$最小值時的θ

一直重複$\theta = \theta - \alpha\nabla J(\theta)$

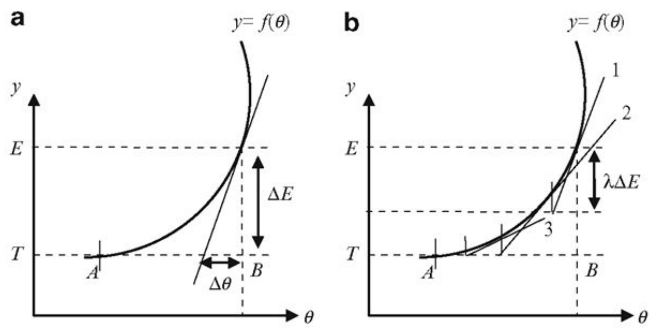
也就是向最陡(梯度$\nabla J(\theta)$)的方向走，α是一步的長度，好的情況下會趨近最小值時的θ

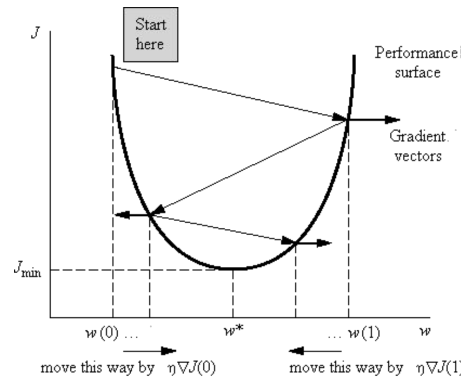
<u>範例：線性GradientDesent</u>

```
import numpy as np
def GradientDescent (x, y, theta, alpha, m, it):
    xT = x.transpose()
    for i in range(0, it):
        J = np.dot(x, theta)
        loss = J-y
        cost = np.sum(loss**2)/(2*m)
        grad = np.dot(xT, loss)/m
        theta = theta - alpha*grad
    return  theta
```

3.2. 不好的情況

有可能會z字型走，如果α太大有可能不收斂





4. Regularization

4.1. Control Over-Fitting

為避免模型過度複雜造成
over-fitting，所以加了
regularization term $E_D(w)+\lambda E_w(w)$
λ是控制它們之間重要性的係數
最簡單的$E_w(w)=\frac{1}{2}w^Tw$

4.2. $\frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$

通常q = 1叫lasso(L1-norm)
右圖為L1-norm與L2-norm

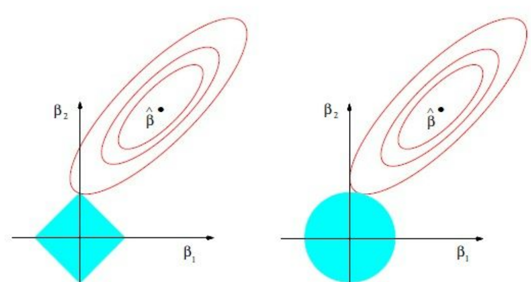


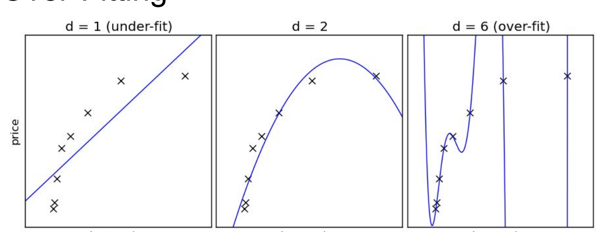
FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions $|\beta_1|+|\beta_2|\le t$ and $\beta_1^2+\beta_2^2\le t^2$, respectively, while the red ellipses are the contours of the least squares error function.*

5. Problems Applying Machine Learning

5.1. Cross Validation

拿一部份的資料用做測試而非訓練，以此觀察訓練結果的好壞

5.2. Over-Fitting



5.3. High Bias

dk 3 模型選得太簡單，造成Training跟Testing Error都偏高(Under-Fit)

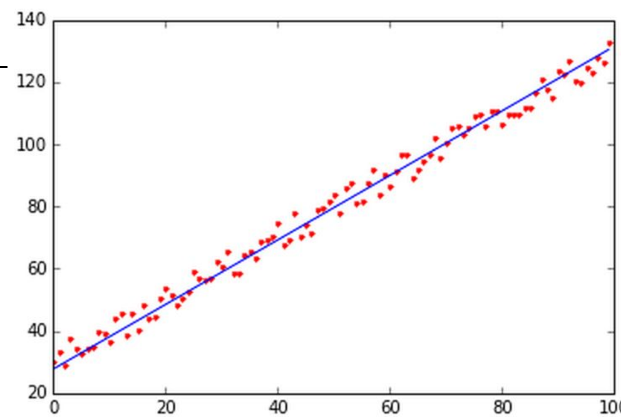
5.4. High Variance

選太多參數或是演算法、模型取不好，造成Testing Error上升但 Training Error下降(Over-Fit)

http://www.astroml.org/sklearn_tutorial/practical.html

Reference: Pattern Recognition and Machine Learning by Christopher M. Bishop

實驗：bias = 25，variance = 10 的100組隨機數據




```
import random
import matplotlib.pyplot as plt
%matplotlib inline

def DataGen(num, bias, var):
    x = np.zeros(shape = (num, 2))
    y = np.zeros(shape = num)
    for i in range(0, num):
        x[i][0] = 1
        x[i][1] = i
        y[i] = i + bias + random.uniform(0, 1)*var
    return x, y

x, y  = DataGen(100, 25, 10)
m, n = np.shape(x)
it = 30000
alpha = 0.0005
theta = np.ones(n)
theta = GradientDescent(x, y, theta, alpha, m, it)

print theta
plt.plot([i[1] for i in x], y, 'r.')
plt.plot(x, theta[1]*x+theta[0], 'b-')
```

Further Reading

https://github.com/nborwankar/LearnDataScience