Machine Learning

2014/12/27

By 謝德威 (tewei)

$pip install scikit-learn
http://scikit-learn.org/

Contents

1. RANSAC for Linear Regression
2. Basic Information Theory
3. Logistic Regression

1. Better Linear Regression: RANSAC

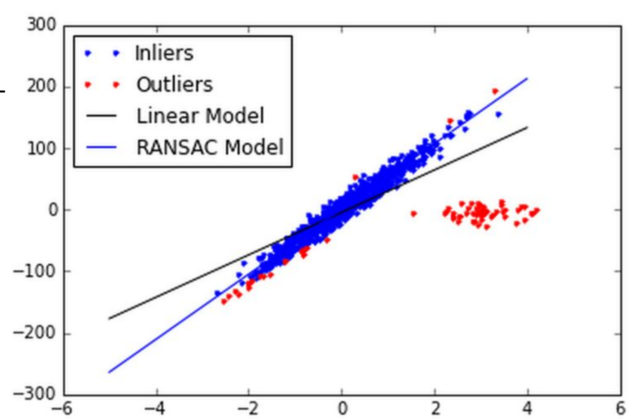
Linear Regression Problem，容易受到雜訊影響，訓練出不好的係數，所以就有一個新的方法叫做，所以有種叫做RANdom SAmple Consensus的東西可以做得更好

RANSAC演算法:

1. *隨機選取儘量少的點作回歸直線*
2. *計算有多少其餘的點近似於此直線*
3. *有愈多點近似則此直線欲佳，更新最佳直線*
   重複算k次，只要次數夠多，就有很大機率趨近最佳直線

k應該是多少呢?

設 $w = \frac{inlier}{samples}$，則執行k次得到的直線完全沒有outlier的機率 $P = 1-(1-w^n)^k$，得到 $k = \frac{log(1-P)}{log(1-w^n)}$，所以P愈高k愈大



實驗：使用sklearn的RANSAC

```
import numpy as np
from matplotlib import
pyplot as plt
from sklearn import
linear_model, datasets
%matplotlib inline

n_samples = 1000
n_outliers = 50
#產生資料(X, y)及正確係數cf
X, y, cf =
datasets.make_regression(n_samples = n_samples, n_features = 1,
n_informative = 1, noise = 10, coef = True)
#加入雜訊
X[:n_outliers] = 3 + 0.5 * np.random.normal(size=(n_outliers, 1))
y[:n_outliers] = -3 + 10 * np.random.normal(size=n_outliers)
#普通的回歸直線
LinearModel = linear_model.LinearRegression()
LinearModel.fit(X, y)
```

```
#RANSAC回歸直線
RANSACModel =
linear_model.RANSACRegressor(linear_model.LinearRegression())
RANSACModel.fit(X, y)
#把雜訊分開
in_mask = RANSACModel.inlier_mask_
out_mask = np.logical_not(in_mask)
print(cf, LinearModel.coef_, RANSACModel.estimator_.coef_)
#劃線
line_X = np.arange(-5, 5)
line_y = LinearModel.predict(line_X[:, np.newaxis])
line_y_ransac = RANSACModel.predict(line_X[:, np.newaxis])
#其實這是sklearn的sample code xP
plt.plot(X[in_mask], y[in_mask], '.b', label='Inliers')
plt.plot(X[out_mask], y[out_mask], '.r', label='Outliers')
plt.plot(line_X, line_y, '-k', label='Linear Model')
plt.plot(line_X, line_y_ransac, '-b', label='RANSAC Model')
plt.legend(loc='upper left')
plt.show()
```

2. Basic Information Theory

2.1. 身為資訊社怎能不會資訊理論呢? xP

假設有一個隨機變數x，它有八種狀態{a, b, c, d, e, f, g, h}，每種狀態出現的機率相等，需要用多少bit來表示它？

如果他們出現的機率為 $\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}\}$，是否可以換個編碼方式使得平均長度最小？注意必須是Noiseless Coding哦!!!

Ans: 如果我們用0, 10, 110, 1110, 111100, 111101, 111110, 111111來編碼，算一下期望值，可以得到平均2 Bit!!!

2.2. Entropy and Bit

要怎麼計算理論上平均編碼長度的最小值呢？

$H[x] = -\sum_i p(x_i) lg p(x_i)$ 單位是Bit，如果換成 $H[x] = -\sum_i p(x_i) ln p(x_i)$ 單位是Nat，當然在資料大時可以寫成積分 $H[x] = -\int p(x_i) ln p(x_i) dx$

2.3. 資料壓縮與 Huffman Coding

http://www.columbia.edu/~cs2035/courses/csor4231.F11/huff.pdf

2.4. Relative Entropy - KL Divergence

如果我們用一個近似機率分佈 $q(x_i)$ 對實際資料 $p(x_i)$ 編碼，則需要的額外長度是 $KL[p\|q] = -\int p(x_i) ln\{\frac{q(x_i)}{p(x_i)}\} dx$

3. Logistic Regression

3.1. 2 Class Classification

設兩個Class C_1 與 C_2，feature vector是 φ，要學的係數是 w

Sigmoid Function $\sigma(a) = \frac{1}{1+e^{-a}}$

Model $P(C_1|\varphi) = y(\varphi) = \sigma(w^T\varphi)$，當然 $P(C_2|\varphi) = 1 - P(C_1|\varphi)$

所以Logistic Regression 不是 Regression xP

3.2. Likelihood Function

對於一組 (φ_n, t_n)，Likelihood Function是 $\prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$(可以想像 如果正解是0預測愈接近0則 $(1-y_n)^{1-t_n}$ 愈大)

3.3. Error Function

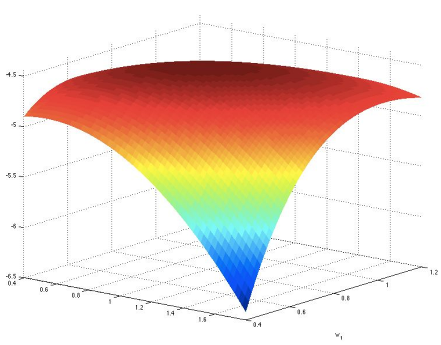
通常會用Negative Log Likelihood Function，或叫做Cross Entropy Error Function

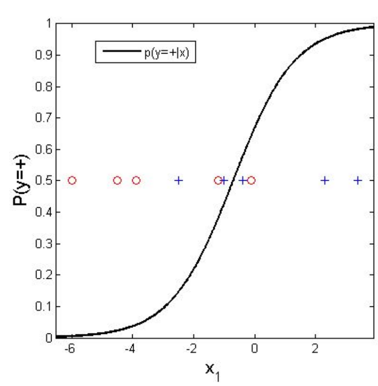
$$E(w) = -ln(\prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}) = -\sum_{n=1}^{N}\{t_n ln y + (1-t_n) ln(1-y_n)\}$$

就像Linear Regression，為了求 $min\ E(w)$，要求它的微分=0時的 w，這個 w 有可能使 $E(w)$ 是最小值，好心人告訴我們 $\nabla E(w) = \sum_{n=1}^{N}(y_n - t_n)\varphi_n$

問題是這個東西有辦法像Linear Regression一樣直接解嗎www?

答案是不行QQ，但是這個函數還是凸的，所以我們可以用Gradient Descent或Newton-Raphson Method來求最小值(或是不加負號則為Gradient Ascent，Log Likelihood Function與訓練結果如下圖)





https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=lectures.logistic

直接用Numpy實作Logistic Regression
http://www.mblondel.org/tlml/logreg.py.html
http://people.csail.mit.edu/jrennie/writing/lr.pdf

Further Reading

用Theano實作Logistic Regression
http://deeplearning.net/tutorial/logreg.html

範例：Pandas+Patsy
http://nbviewer.ipython.org/github/justmarkham/gadsdc1/blob/master/logistic_assignment/kevin_logistic_sklearn.ipynb

Stanford CS229 Notes
http://cs229.stanford.edu/materials.html