Logistic Regression

—— Algorithm implemented in Octave
—— Simple data were applied with the algorithm

—— Same data were analyzed in R
—— Same data were analyzed in Python

Algorithm

Hypothesis $h_\theta(x) = g(\theta^T x)$

$g(z) = \frac{1}{1+e^{-z}}$ Sigmoid function

Parameters: $\cancel{\theta_0, \theta_1, \ldots, \theta_n}$ θ

cost function

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} \boxed{-\log(h_\theta(x))} & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

Want $\min_\theta J(\theta)$:

Repeat {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

} (simultaneously update all θ_j)

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

Advanced optimization

Optimization algorithms:
- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:
- No need to manually pick α
- Often faster than gradient descent.

Disadvantages:
- More complex

Implemented in Octave

logistic_regression.m

Sigmoid Function ← training.txt, $g(z) = \frac{1}{1+e^{-z}}$, sigmoid.m

↓

Gradient Descent ← $\theta_j := \theta_j - \alpha \boxed{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}}$, costFunction.m

↓

Predict Testing Data ← testing.txt, predict.m

↓

prediction.txt

Data Analyzed in R

File Name:
logistic_regression.R

Usage:
/usr/bin/Rscript logistic_regression.R training.txt testing.txt prediction.txt

Core Functions{Packages} Used:
build model
glm{stats}
make prediction on testing data
predict{stats}

Data Analyzed in Python

File Name:
logistic_regression.py

Usage:
python logistic_regression.py training.txt testing.txt prediction.txt

Core Functions{Modules} Used:
build model
LogisticRegression().fit(){sklearn}
make prediction on testing data
LinearRegression().predict_proba(){sklearn}