CS 4340 - Logistic Regression

Austin Hester

November 5, 2017

Introduction

We will use logistic regression to obtain probabilities of passing a course given a number of weeks inactive.

Our training data is:

Training Data	
Weeks Inactive	**Pass/Fail**
1	0
2	1
3	0
4	1
5	0
6	1
7	1
8	1

0 = pass, 1 = fail

The Code

```
# Austin Hester
# Logistic Regression
# CS 4340 - Intro to Machine Learning
# 11.05.17

import numpy as np
# define x_0
x0 = 1

# ln L = sum_1_n{ x_i^j * ( y^j - ( e^{w0x0+w1x1} / (1 + e^{w0x0+w1x1} )))
    }

# compute d/dwi ln L with given x, y, weights, and step size
def ddw(i, x, y, w_, c):
    s = 0
    # for d/dw1 ln L
    for j in range(1,9):

        eexp = np.exp( ( w_[0] * x0 ) + ( w_[1] * x[j-1] ) )
        if (i == 0):
            pointmult = 1
        else:
            pointmult = x[j-1]
        point = pointmult * (y[j-1] - ( eexp / ( 1 + eexp ) ) )
        s = s + point
    w = w_[i] + (c * s)
    return w

# compute the passing chance given x weeks of inactivity
def passingchance(w_, x):
    chance = 1 / ( 1 + np.exp(x0 * w_[0] + (x * w_[1])))
    return chance

# input data [1-8], "weeks of inactivity"
#x = np.arange(1, 9)
x = np.array( [1,2,3,4,5,6,7,8] )
# output, 0 = "pass", 1 = "fail"
y = np.array( [0,1,0,1,0,1,1,1] )
# step size
c = 0.01
# initial weight vector
w_ = np.array( [1., 1.] )
```

```
# iterate T times
T = 2000
for t in range(T):
    new_w0 = ddw(0,x,y,w_,c)
    new_w1 = ddw(1,x,y,w_,c)
    w_[0] = new_w0
    w_[1] = new_w1

# print weight vector
print("\nWeight vector: ", w_)
print("\nWeeks of Inactivity\tChances of passing")
for i in range(0,13):
    print("\t",i, "\t\t", round(passingchance(w_, i),4)*100, "%")
```

Notes

(a) Logistic regression results

```
Weight vector:  [-1.8142984   0.5594476]

Weeks of Inactivity     Chances of passing
     0                  85.99 %
     1                  77.81 %
     2                  66.72 %
     3                  53.39 %
     4                  39.57 %
     5                  27.23 %
     6                  17.62 %
     7                  10.89 %
     8                   6.53 %
     9                   3.84 %
     10                  2.23 %
     11                  1.29 %
     12                  0.74 %
```

I started with a step size, $c = 0.1$, which gave shaky results for the weight vector. It would bounce from $< -2.24, 1.02 >$ to $< -2.36, 0.50 >$ every other iteration.
So I decided to lower the step size to $c = 0.01$ to avoid the jumpiness.

Running logistic regression over our input data gives us a weight vector of $< -1.81, 0.56 >$.

At 3 weeks of inactivity, a student has a 53.4% chance of passing the course.
At 5 weeks of inactivity, a student has a 27.2% chance of passing the course.

(b) Can logistic regression be used for classification?

Logistic regression can very well be used for classification.

We can classify using:

if [$P(Y = 0|X) > P(Y = 1|X)$] then pass
else fail

(c) Suppose two or more rows in the training data set have the same x-value, but different y-values (e.g. (x=3,y=1) and (x=3,y=0)).

Would we still be able to obtain a valid logistic regression of Y on X?

Yes. Here is a run with both 0 and 1 at x = 3:

```
Weight vector:   [-1.08989798   0.43602457]

Weeks of Inactivity      Chances of passing
     0                     74.84 %
     1                     65.79 %
     2                     55.42 %
     3                     44.57 %
     4                     34.2 %
     5                     25.16 %
     6                     17.85 %
     7                     12.32 %
     8                      8.33 %
     9                      5.55 %
     10                     3.66 %
     11                     2.4 %
     12                     1.56 %
```

Having x-values with different y-values adds a degree of uncertainty, which leads to less confident predictions.