HW3_problem5

April 29, 2015

```
In [3]: %matplotlib inline
        import scipy
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

        data_link = "http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data"
        import requests
        from cStringIO import StringIO
        data = StringIO(requests.get(data_link).text)
        df = pd.read_csv(data, sep=",", index_col=0)
```

```
In [88]: df.head()
```

Out[88]:

row.names	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
1	160	12.00	5.73	23.11	Present	49	25.30	97.20	52	1
2	144	0.01	4.41	28.61	Absent	55	28.87	2.06	63	1
3	118	0.08	3.48	32.28	Present	52	29.14	3.81	46	0
4	170	7.50	6.41	38.03	Present	51	31.99	24.26	58	1
5	134	13.60	3.50	27.78	Present	60	25.99	57.34	49	1

```
In [113]: x=df[["sbp", "tobacco", "ldl", "adiposity", "typea", "obesity", "alcohol", "age", "chd"]]
          y=df[["famhist"]]
          print x.shape
```

```
(462, 9)
```

The classification boundary is given as follows

$$w^T X + b = 0$$

where $w^T X^+ + b > 1$ and $w^T X^- + b < -1$

Now, error in classification for X^+ is given by

$$w^T X^+ + b = 1 - z$$

and for X^- it is

$$w^T X^- + b = -1 + l$$

Thus for best classification we need minimize z and l, Thus if p and q denote the size of X^+ and X^-, then the linear programming problem is stated as,

$$\min \frac{1}{p}(z) + \frac{1}{q}(l) \tag{1}$$
$$\text{such that} \tag{2}$$
$$w^T X^+ + b - 1 \geq -z \rightarrow -w^T X^+ - b + 1 \leq z \tag{3}$$
$$w^T X^- + b + 1 \leq l \tag{4}$$
$$z \geq 0 \tag{5}$$
$$l \geq 0 \tag{6}$$
$$\tag{7}$$

The constraints can be re-written as

$$-w^T X^+ - zI - b \leq -1 \tag{8}$$
$$w^T X^- - lI + b \leq -1 \tag{9}$$
$$\tag{10}$$

Now the constraints can be changed to the form $X'e' \leq b$, where X' is a horizontally stacked matrix of $[X^+, M, b * 1_{\text{m x 1}}]$ where M is a vertically stacked Matrix of $[I_p, 0_{q \text{ x } 1}]$

e' is a matrix of form $[w, z, l, b]$ with dimensions n+p+q+1 = n+m+1

```
In [132]: from sklearn.base import ClassifierMixin
          from scipy.optimize import linprog

          class LP_classifier(ClassifierMixin):
              def __init__(self):
                  self.w = None
                  self.b = None
                  self.slack = None

              def fit(self, X, y):
                  m, n  = X.shape
                  m, n = X.shape
                  mask = np.array(y)==1
                  p = X[mask].shape[0]
                  q = m - p
                  C = np.vstack((np.zeros((n,1)), np.ones((p,1))/float(p), np.ones((q,1))/float(q), 0))
                  M = np.hstack((np.identity(p), np.zeros((p,q))))
                  X_dash = np.hstack((-1*X[mask], M, -1*np.ones((p,1))))
                  M2 = np.hstack((np.zeros((q,p)), -1*np.identity(q),))
                  X_dash2 = np.hstack((X[~mask], M2, np.ones((q,1))))
                  #stack both these matrixes vertically so that both constraints can be checked in one
                  #print X_dash.shape, X_dash2.shape
                  lhs = np.vstack((X_dash, X_dash2))
                  for i in range(n+1, n+p+q):
                      lhs = np.vstack((lhs, np.zeros((1, lhs.shape[1]))))
                      lhs[-1,i] = -1
```

```
            print lhs.shape
            #similarly combine both RHS of the constraints
            rhs = -1*np.ones((lhs.shape[0], 1))
            #rhs = np.vstack((-1*np.ones((p,1)), -1*np.ones((q,1))))
            for i in range(p+q+1, lhs.shape[0]):
                rhs[i] = 0
            #bounds = np.empty((n,1)), np.zeros()
            self.lp_result = linprog(C.flatten(), lhs, rhs)
            res = self.lp_result.x
            #print res.success
            self.w = res[0:n]
            self.b = res[-1]
            #self.slack = res[n+1:n+m]
            return self

        def predict(self, X):
            return (np.dot(x, self.w) + self.b)>0
```

In [133]:
```
print x.shape
s = LP_classifier().fit(x,y)
```

```
(462, 9)
(923, 472)
```

In [134]:
```
s.predict(x)
```

Out[134]:
```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
```

```
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False], dtype=bool)
```

```
In [ ]:
```