Machine Learning Homework 1

Sang Woo Jun

September 23, 2011

1 Online Perceptron

1.1

The number of mistakes would be d. This is because each mistake and update of the perceptron does not affect the next orthogonal unit vector. A mistake has to be made on each d unit vectors for everything to be classified correctly.

1.2

The largest value of γ_g is :

$$\frac{1}{\sqrt{d}}$$

This is because the perceptron algorithm makes at most

$$\frac{R^2}{\gamma_g^2}$$

which equals d, and R equals 1 for orthogonal unit vectors. It is trivial to see that this holds in 2 and 3 dimensions.

1.3

An example of linearly independant unit vectors with a relatively large margin can be given with two unit vectors on a two-dimentional plane, each pointing to $(0 \times \pi, 1)$ and $(0.999 \times \pi, 1)$ in polar coordinates. Similarly, vector sequences with large margins can be generated with vectors clustered around a $d-1$ dimension plane. This would be a one dimensional line in the above example.

1.4

The dimension of the subspace spanned is always 1 if $\gamma_g = 1$. In order for margin to be 1, all unit vectors must be located on either $||\theta||$ or $-||\theta||$.

2 Regression with class targets

2.1

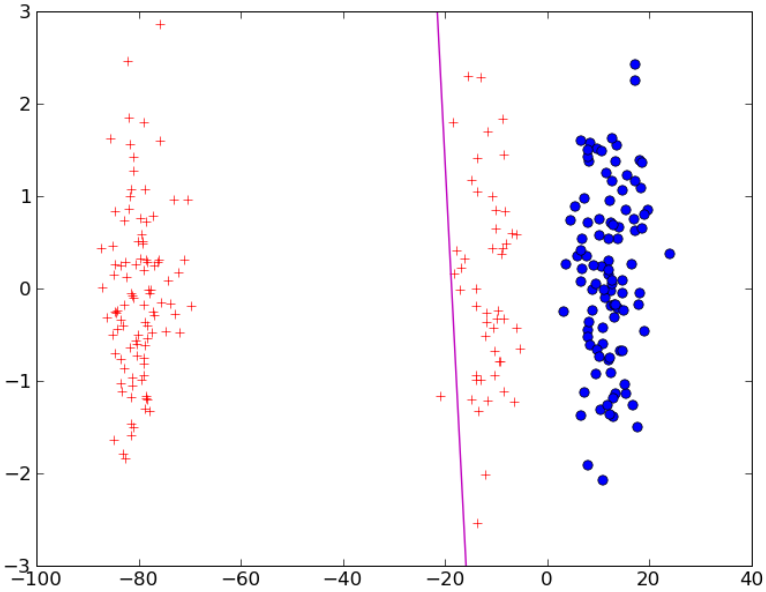
The code segment for the linear least-square regression is listed below :

```
from numpy import *

def linreg (X, y):
        cX = c_[X, ones((X.shape[0],1))];
        tcX = transpose(cX);

        b = dot(dot(mat(dot(tcX, cX).copy()).I, tcX),y);
        t = ravel(b[:b.size-1]);
        rb = ravel(b[b.size-1]);
        return (t, rb);
```

2.2

A simple situation would be when the data is not roughly balanced. Because least-square linear regression attempts to minimize sum of least squared errors, the optimal fitting is largely affected by outlying data points. A similar affect can be observed by locating a few data points extremely far from the rest of the data points.

The following graph is an example of attempting to classify an imbalanced data set.



2.3

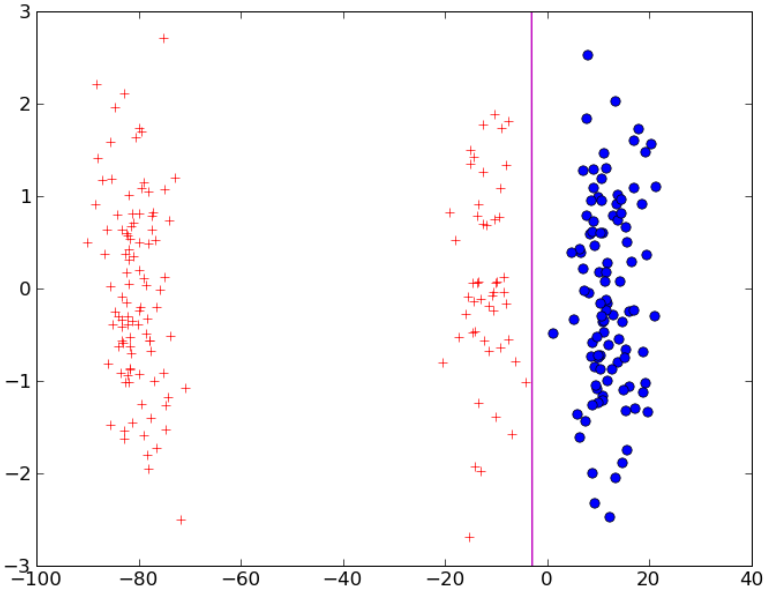
For the purpose of a separator, data points near the separator are more important. Therefore for regression to find a more suitable separator, the regression targets of outlying data points can be adjusted to have larger absolute values to less disturb the regression. The problem then, is recognizing the outlying data points, and deciding the the re-weighing principle.

Intuitively, efficient methods to recognize outliers would include dividing points into few percentile classes according to their distance from (1) origin or (2) average location of all data points. A more resource requiring method would be to first determine the separator vector, and classifying data points according to their distance from it.

Methods to decide weighs could include linear or non-linear proportional weights according to distance percentile.

However, such methods generally do not guarantee safe separation, and therefore be only used to ameliorate error or by empirically adjusting parameters.

The following graph is the saparation of the data from the previous question, with the data points in the left hand cluster given y values of 4.



3 Linear support vector machine

3.1

The code for the dual form of SVM with slack variables is listed below. A peculiarity was that zero values resulting from the optimizer seemed to be represented using infinitesimally small numbers (order of e-12). This resulted in even outlying data points to be classified as support vector points, so a tiny margin was given when determining zero.

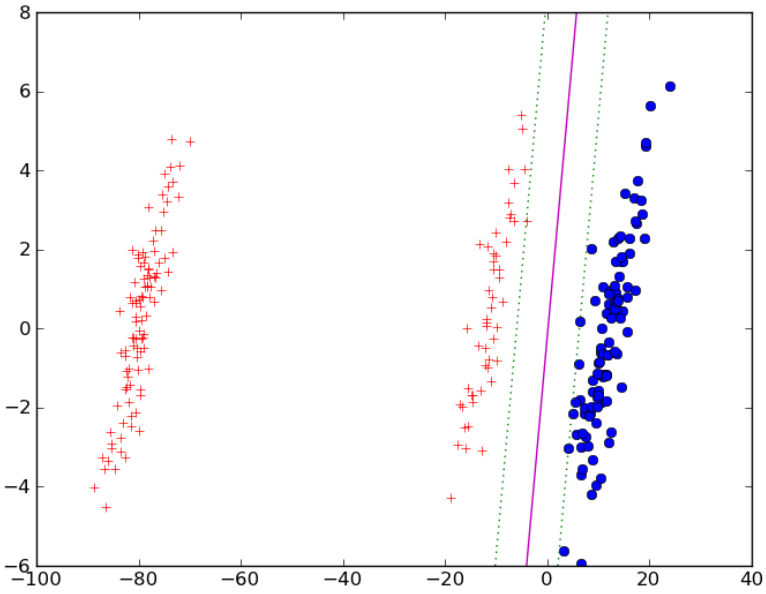
```
from numpy import *
from cvxopt import *

def linsvm_prim_train(X, y, C):
        shapeX = X.shape
        H = ones((shapeX[0],shapeX[0]));

        for i in range(shapeX[0]):
                for j in range(shapeX[0]):
                        d = X[i] * X[j];
                        H[i,j] = y[i] * y[j] * (d[0]+d[1]);
```

```
P = matrix(H);
q = matrix(ones(shapeX[0]) * -1);

G = matrix(zeros((shapeX[0]*2, shapeX[0])));
h = matrix(zeros(shapeX[0]*2));
for i in range(shapeX[0]):
        G[i,i] = -1;
        G[i+shapeX[0],i] = 1;
        h[shapeX[0]+i] = C;


h = matrix(zeros(shapeX[0]));

A = matrix(y.transpose(), tc = 'd');
print A;
b = matrix(zeros(1));

sol = solvers.qp(P, q, G, h, A, b);

w = matrix(zeros((1,shapeX[1])));
i = 0;
S = [];
countS = 0;
a = sol['x'];
for x in a:
        w = w + x * y[i] * X[i];

        # Something wrong with zero representation?
        if x > 0.000001 and x <= C:
                S.append(i);
                countS = countS + 1;

        i = i + 1;

tts = 0;
for s in S:
        ts = 0;
        for m in S:
                d = X[m] * X[s];
                ts = ts + a[m] * y[m] * (d[0] + d[1]);
        tts = tts + y[s] - ts;

if countS > 0:
        tts = tts / countS;
else:
        tts = 0;
```

```
return (ravel(w),tts, a);
```

3.2

As can be seen from figure below, the SVM correctly separated the two classes



3.3

3.3.1 a

When $C > 1$, the margin becomes 1, which is also the hard margin of the data set. There are no more support vectors, and therefore no slack variables to penalize with C. Increasing C will have no effect beyond this point.

3.3.2 b

Because margin $\frac{1}{||\theta||}$ is 2, θ is $\frac{1}{2}$. (Not $-\frac{1}{2}$ because the data point to the right has value $+1$)

Because the separator goes through (0,0), b is 0.

And from $x_i \cdot w + b \geq +1 - \xi_i$ for $y_i = +1$ and $x_i \cdot w + b \leq -1 + \xi_i$ for $y_i = -1$, it can be seen that $\xi_1 = \xi_2 = \frac{1}{2}$.

This can be trivially confirmed from the fact the margin is 2 and the support vectors are of length 1. The difference is filled up by slack variables which amount to $margin \times \xi$

All of the steps in calculating the paramters were deterministic. Therefore the answer must be unique.

3.4

3.4.1 a

Margin decreases as C increases. Results are as follows:

C	margin
0.01	3.50
0.1	2.64
1	1.69
10	1.69
100	1.19

If the data is not linearly separable, margin will always decrease as C increases in an attempt to minimize slack penalty. If it is linearly separable, margin will decrease until the hard margin limit is met and there are no support vectors to penalize.

3.4.2 b

The number of support vectors decrease as C increases. Results are as follows:

C	no. SVs
0.01	18
0.1	11
1	10
10	10
100	9

3.4.3 c

In order to increase the margin, the value of C should be made as small as possible. However, this would enforce too little penalty on slack variables and could result in more misclassifications. This could also result in misclassifications even when the classes are separable, because the effects of data points outside the margin can be arranged so to override and ignore the small slack penalty.

Another criterion for selecting C could be the number of misclassified data points in the training set.

3.5

Since support vectors are prependicular to the decision boundary, the value of ξ_i is directly proportional to its distance from the decision boundary.

When ξ_i is 0, the data point is correctly classified, and the support vector points $\frac{1}{|w|}$ away from the decision boundary. The distance linearly decreases as ξ_i increases, until ξ_i becomes 1 and the data point is located on the decision boundary. Afterwards, the distance linearly increases as ξ_i increases, at a rate of $\frac{1}{|w|}$.

3.6

optional.

N/A