Stony Brook University
CSE512 – Machine Learning – Spring 17
Homework 2, Due: March, 7, 2017, 11:59PM
Sudeshna Pal(110938222)

Instructions

- The homework is due on March 7, 2017. Anything that is received after the deadline will not be considered.
- The write-up **must** be prepared in Latex, including the Matlab code and figures in the report.
- We can use any Latex class you like, just report question number and your answer.
- If the question requires you to implement a Matlab function, the answer should be your code. Make sure it is sufficiently well documented that the TAs can understand what is happening.
- Each Question, regardless of how many sub-questions contains, is worth 10 points.

1 Ridge Regression

1.1 Question 1

In class, you learned about using k-folds cross validation as a way to estimate the true error of a learning algorithm and to tune parameters. The preferred solution is *Leave-One-Out Cross Validation* (LOOCV), which provides an almost unbiased estimate of this true error, but it can take a really long time to compute. In this problem, you will derive an algorithm for efficiently computing the LOOCV error for a particular regression algorithm.

We will now introduce a simple extension of the Least Square algorithm. Given a set of m data points and associated labels $\{\boldsymbol{x}_i, y_i | \boldsymbol{x}_i \in \Re^d, y_i \in \Re\}_{i=1}^m$. **Ridge Regression** finds the weight vector $\boldsymbol{w}$ and a bias term b to optimize the following:

$$\min_{\boldsymbol{w},b} \ \lambda\|\boldsymbol{w}\|^2 + \sum_{i=1}^{m}(\boldsymbol{w}^\top \boldsymbol{x}_i + b - y_i)^2 \ . \qquad (1)$$

Note that with $\lambda = 0$ Ridge Regression is exactly the Least Square formulation.

- Using the Matlab notation, let $\bar{\boldsymbol{w}} = [\boldsymbol{w}; b], \bar{X} = [X; \mathbf{1}^\top], \bar{I} = [I, \mathbf{0}; \mathbf{0}^\top, 0]$, $C = \bar{X}\bar{X}^\top + \lambda\bar{I}$, and $\boldsymbol{d} = \bar{X}\boldsymbol{y}$. Show that the solution of Ridge regression is:

$$\bar{\boldsymbol{w}} = C^{-1}\boldsymbol{d} \ . \qquad (2)$$

SOLUTION

Ordinary least squares (ols) minimizes the sum of squared errors. However if there is a strong correlation between two parameters ols fails. So, ridge regression minimizes the squared error satisfying additional constraint that is $\sum_{i=1}^{P} w_i^2 \leq c$.
If number of parameters, $P = 2$, minimum squared error for ridge regression is:

$$min \sum_{i=1}^{m} (y_i - w_0 + w_1 x_{1i} + w_2 x_{2i})^2 \tag{3}$$

$$\text{constrained to} \sum_{i=1}^{2} w_i^2 \leq c \tag{4}$$

By using Lagrange multiplier, the above equation becomes,

$$min \sum_{i=1}^{m} (y_i - w_0 + w_1 x_{1i} + w_2 x_{2i})^2 + \lambda\big(\sum_{j=1}^{2} w_i^2 - c\big) \tag{5}$$

λ and c are constants. So, we can write the above equation as

$$min \sum_{i=1}^{m} (y_i - w_0 + w_1 x_{1i} + w_2 x_{2i})^2 + \lambda\big(\sum_{j=1}^{2} w_i^2\big) \tag{6}$$

In matrix form,

$$min (Y - \bar{w}^T \bar{X})(Y - \bar{w}^T \bar{X})^T + \lambda \bar{w}^T \bar{w} \tag{7}$$

Conceptually, we understand that $\lambda \bar{w}^T \bar{w}$ would be the penalty paid highly correlated parameters. Now to find minimum we do partial derivative wrt. $\bar{w}$ and equating it to 0.

$$\frac{\partial}{\partial \bar{w}} = 0 \tag{8}$$

$$-2\bar{X}Y + 2\bar{X}\bar{X}^T \bar{w} + 2\lambda \bar{w} = 0 \tag{9}$$

$$\bar{w}(\bar{X}\bar{X}^T + \lambda \bar{I}) = \bar{X}Y \tag{10}$$

$$\bar{w} = (\bar{X}\bar{X}^T + \lambda \bar{I})^{-1} \bar{X}Y \tag{11}$$

According to the question $C = \bar{X}\bar{X}^T + \lambda \bar{I}$ and $d = \bar{X}Y$, then the above equation becomes,

$$\bar{w} = C^{-1} d \tag{12}$$

- Now suppose we remove $\boldsymbol{x}_i$ from the training data, let $C_{(i)}, \boldsymbol{d}_{(i)}, \bar{\boldsymbol{w}}_{(i)}$ be the corresponding matrices for removing $\boldsymbol{x}_i$. Express $C_{(i)}$ in terms of C and $\boldsymbol{x}_i$. Express $\boldsymbol{d}_{(i)}$ in terms of $\boldsymbol{d}$ and $\boldsymbol{x}_i$.

SOLUTION

After removing x_i from training data, we know the corresponding rows and columns from matrices will be not present. Hence,

$$d_{(i)} = d - x_i y_i.$$

$$C_{(i)} = C - x_i x_i^T$$

- Express $C_{(i)}^{-1}$ in terms of C^{-1} and $\boldsymbol{x}_i$. *Hint:* use the Sherman-Morrison formula

$$(A + \boldsymbol{u}\boldsymbol{v}^\top)^{-1} = A^{-1} - \frac{A^{-1}\boldsymbol{u}\boldsymbol{v}^\top A^{-1}}{1 + \boldsymbol{v}^\top A^{-1}\boldsymbol{u}} \ . \tag{13}$$

SOLUTION

$$\begin{aligned}
C_{(i)}^{-1} &= (C - x_i x_{(i)}^T)^{-1} \\
C_{(i)}^{-1} &= (C + (-x_i) x_{(i)}^T)^{-1} \\
&= \big(C^{-1} - \frac{C^{-1}(-x_{(i)})x_{(i)}^T C^{-1}}{1 + x_{(i)}^T C^{-1}(-x_{(i)})}\big) \qquad =
\end{aligned}$$

- Show that

$$\bar{\boldsymbol{w}}_{(i)} = \bar{\boldsymbol{w}} + (C^{-1}\bar{\boldsymbol{x}}_i)\frac{\bar{\boldsymbol{x}}_i^\top \bar{\boldsymbol{w}} - y_i}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} \ . \tag{14}$$

SOLUTION

Substituting $\bar{w}$ with just w, $\bar{x}_i$ with just x_i to make writing easy and the new solution after removing x_i will be,

$$C_i^{-1} d_i$$

Substituting values of C_i^{-1} and d_i from previous section we get,

$$w_i = (C^{-1} + \frac{C^{-1}x_i x_i^T C^{-1}}{1 - x_i^T C^{-1} x_i})(d - x_i y_i)$$

$$w_i = C^{-1}d + \frac{C^{-1}x_i x_i^T C^{-1}}{1 - x_i^T C^{-1} x_i} d - C^{-1}x_i y_i - \frac{C^{-1}x_i x_i^T C^{-1}}{1 - x_i^T C^{-1} x_i} x_i y_i$$

Replacing C^{-1} with w and taking $C^{-1}x_i$ common, we get,

$$w_i = w + C^{-1}x_i(\frac{x_i^T C^{-1}}{1 - x_i^T C^{-1} x_i} d - y_i - \frac{x_i^T C^{-1}}{1 - x_i^T C^{-1} x_i} x_i y_i)$$

$$w_i = w + C^{-1}x_i(\frac{x_i^T C^{-1} d - y_i + x_i^T C^{-1} x_i y_i - x_i^T C^{-1} x_i y_i}{1 - x_i^T C^{-1} x_i})$$

$$\boldsymbol{w_i = w + (C^{-1}x_i)(\frac{x_i^T w - y_i}{1 - x_i^T C^{-1} x_i})} \tag{*}$$

- Show that the leave-one-out error for removing the i^{th} training sample is

$$\bar{\boldsymbol{w}}_{(i)}^\top \bar{\boldsymbol{x}}_i - y_i = \frac{\bar{\boldsymbol{w}}^\top \bar{\boldsymbol{x}}_i - y_i}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} \ . \tag{15}$$

SOLUTION

We know that the error is given by

$$E = w_i^T x_i - y_i$$

Substituting the value of w_i from previous answer,

$$E = (w + (C^{-1}x_i)(\frac{x_i^T w - y_i}{1 - x_i^T C^{-1} x_i}))^T x_i - y_i$$

$$E = (w^T + ((C^{-1}x_i)(\frac{x_i^T w - y_i}{1 - x_i^T C^{-1} x_i}))^T x_i - y_i$$

$$E = (w^T + ((\frac{x_i^T w - y_i}{1 - x_i^T C^{-1} x_i})^T (C^{-1}x_i)^T) x_i - y_i$$

$$E = (w^T + (\frac{(x_i^T w - y_i)^T}{1 - x_i^T C^{-1} x_i})(x_i^T (C^{-1})^T) x_i - y_i$$

$$E = (w^T + (\frac{(x_i^T w)^T - y_i^T}{1 - x_i^T C^{-1} x_i})(x_i^T (C^{-1})^T) x_i - y_i$$

$$E = (w^T + (\frac{w^T x_i - y_i^T}{1 - x_i^T C^{-1} x_i})(x_i^T (C^{-1})^T) x_i - y_i$$

$$E = \frac{(w^T(1 - x_i^T C^{-1} x_i) + (w^T x_i - y_i^T)(x_i^T (C^{-1})^T) x_i - y_i(1 - x_i^T C^{-1} x_i)}{1 - x_i^T C^{-1} x_i}$$

We know that C is symmetric and hence we can replace C^T with C and also, we know that $(C^{-1})^T = (C^T)^{-1} = C^{-1}$. Therefore,

$$E = \frac{(w^T(1 - x_i^T C^{-1} x_i) + (w^T x_i - y_i^T)(x_i^T C^{-1}) x_i - y_i(1 - x_i^T C^{-1} x_i)}{1 - x_i^T C^{-1} x_i}$$

$$E = \frac{w^T x_i - y_i}{1 - x_i^T C^{-1} x_i} \qquad (*)$$

Substituting the above value in error equation, we get

$$\boldsymbol{w_i^T - y_i = \frac{w^T x_i - y_i}{1 - x_i^T C^{-1} x_i}} \qquad (*)$$

- The LOOCV is defined as: $\sum_{i=1}^{m}(\bar{\boldsymbol{w}}_{(i)}^{\top}\bar{\boldsymbol{x}}_i - y_i)^2$. What is the algorithmic complexity of computing LOOCV error using the formula given in the previous point? How is it compared with the usual way of computing LOOCV? Note that the complexity of inverting a $k \times k$ matrix is $O(k^3)$.

If we use the normal approach , then at every step we will end up spending minimum of $O(n^3)$ time for computing inverse of matrix. Therefore the total complexity will be $O(n^3)$. However, in the ridge regression all we need to do is to do matrix multiplications and addition. Therefore the time complexity of the new approach will be $O(n^2)$.

1.2 Question 2

Implement the Ridge Regression algorithm that finds the solution of (1) through (2). The prototype of the function must be

```
[w,w_0] = train_rr(X, y, lambda)
```

where $X \in \mathbb{R}^{m\times d}$ is the matrix of m input vectors in d dimension, i.e. each input $\boldsymbol{x}_i$ is a row of X, y is a column vector of m columns containing the labels associated to the training samples. lambda is the value of λ in (1). You can assume $\lambda > 0$.

SOLUTION

```
function [w, w_0] = train_rr (X, Y, lambda)


    IdentityMatrix = eye(size(X,2));
    C = X' * X + (lambda * IdentityMatrix);
    d = X' * Y;

    w = pinv(C) * d;

    w_0 = w(1);
    w = w(2:size(w, 1));
end
```

2 Perceptron

Given a sequence of m samples $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and the labels $y_i \in \{-1, 1\}$, the Perceptron algorithm runs as follows:

Algorithm 1 Perceptron pseudocode

```
Initialize: $\boldsymbol{w} = 0$
for $i = 1, \cdots, T$ do
    if $y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle < 0$ then
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_i\boldsymbol{x}_i$
    end if
end for
```

For simplicity, we have put the bias b into $\boldsymbol{w}$, that is $\boldsymbol{w} \leftarrow [\boldsymbol{w}; b]$ and $\boldsymbol{x}_i \leftarrow [\boldsymbol{x}_i; 1]$. So it is no necessary to consider the bias term in next two questions. Assume $\|\boldsymbol{x}_i\| \le R$ for all i. We have shown that if there exists some w^* such that $\|w^*\| = 1$ and for all i $y_i\langle w^*, \boldsymbol{x}_i\rangle \ge \gamma$, then the number of mistakes is upper bounded by

$$\frac{R^2}{\gamma^2} \ . \tag{16}$$

2.1 Question 3

Consider the generalized Perceptron updates $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y_i \boldsymbol{x}_i$ with *learning rate* $\eta > 0$. The Perceptron algorithm is the special case $\eta = 1$. Prove a bound on the number of mistakes similar to (16). How does η affect this bound?

SOLUTION

Update rule given:

$$w_{(i+1)} = w_i + \eta x_i y_i$$

Therefore

$$||w_{(i+1)}||_2^2 = ||w_i||^2 + 2\eta y_i < w_i, x_i > + ||x_i||_2^2$$

$$\leq ||w_i||^2 + \eta^2 R^2$$

Assuming M mistakes are made, we can write

$$||w_{(T+1)}||_2^2 \leq M\eta^2 R^2$$

Consider w^* to achieve maximum in the definition of B

$$< w^*, w_T + 1 > \leq ||w^*||_2 ||W_{T+1}||_2 \leq \eta R\sqrt{M}$$

Also consider

$$< w^*, w_T + 1 >= (w^*, + \sum_{t:mistakesonx_t} \eta y_t < w^*, x_t >) \geq \eta\gamma M$$

$$< w^*, w_T + 1 > \geq \eta\gamma M$$

putting it together

$$\eta\gamma M \leq \eta R\sqrt{M}$$

Canceling η and simplification gives us

$$M \leq \frac{R^2}{\gamma^2}$$

Thus the η factor does not affect the bound.

2.2 Question 4

Now let's drop the assumption that samples are linear separable. Let $\boldsymbol{u}$ be any vector with $\|\boldsymbol{u}\| = 1$ and let $\gamma > 0$. Define $\ell_i(\boldsymbol{u}) = \max(0, \gamma - y_i\langle \boldsymbol{u}, \boldsymbol{x}_i\rangle)$ and and $L_q(\boldsymbol{u}) = \sum_{i=1}^m \ell_i(\boldsymbol{u})^q$ for $1 \leq q \leq 2$.

- Building on the previous proof, use the fact that you can express the updates *on all rounds* as $\boldsymbol{w} \leftarrow \boldsymbol{w} + \tau_i y_i \boldsymbol{x}_i$ for $\tau_i \in \{0, 1\}$, and show that the number of mistakes M of the Perceptron algorithm satisfies the *implicit* (i.e. M appears on both sides) inequality

$$\gamma M \leq M^{1-\frac{1}{q}} L_q^{\frac{1}{q}}(\boldsymbol{u}) + R\sqrt{M}, \ \forall 1 \leq q \leq 2 \tag{17}$$

  Hint: Start from the proof seen in class, and use the Hölder's inequality:

$$\sum_{i=1}^m a_i b_i \leq \left(\sum_{i=1}^m |a_i|^q\right)^{\frac{1}{q}} \left(\sum_{i=1}^m |b_i|^p\right)^{\frac{1}{p}}, \ \forall p, q \in [1, +\infty] \text{ such that } \frac{1}{p} + \frac{1}{q} = 1 \ .$$

  Note that the Cauchy-Schwarz inequality is a special case of Hölder's inequality with $p = q = 2$.

  SOLUTION

$$
\begin{aligned}
\|w_i+1\|^2 &= \|w_i\|^2 + 2y_i\langle w_i, x_i\rangle + \|x\|^2 \\
&\leq \|w_i\|^2 + R^2, \text{ since } \ 2y_i\langle w_i, x_i\rangle \leq 0 \text{ and } \|x\|^2 \leq R^2 \\
\langle w^*, w_T+1\rangle &\leq R\sqrt{M} \\
\\
\langle w^*, w_T+1\rangle &= \langle w^*, \sum_t y_t x_t\rangle \\
&= \sum_t y_t \langle w^*, x_t\rangle \\
&= \sum_{i=1}^{M} \gamma - \ell_i(\boldsymbol{u}) \\
&= M\gamma - \sum_{i=1}^{M} \ell_i(\boldsymbol{u}) \\
&= M\gamma - \sum_{i=1}^{M} \ell_i(\boldsymbol{u}) * 1 \\
&\geq M\gamma - \left(\sum_{i=1}^{m} \ell_i(\boldsymbol{u})^q\right)^{\frac{1}{q}} \left(\sum_{i=1}^{m} |1|^p\right)^{\frac{1}{p}}, \ \forall p, q \in [1, +\infty] \text{ such that } \frac{1}{p} + \frac{1}{q} = 1\ . \\
&\geq M\gamma - \left(\sum_{i=1}^{m} \ell_i(\boldsymbol{u})^q\right)^{\frac{1}{q}} \left(\sum_{i=1}^{m} |1|^{1-\frac{1}{q}}\right)^{1-\frac{1}{q}}, \ \forall 1 \leq q \leq 2 \\
&\geq M\gamma - L_q^{\frac{1}{q}}(\boldsymbol{u}) M^{1-\frac{1}{q}}, \ \forall 1 \leq q \leq 2 \\
R\sqrt{M} &\geq M\gamma - L_q^{\frac{1}{q}}(\boldsymbol{u}) M^{1-\frac{1}{q}}, \ \forall 1 \leq q \leq 2 \\
M\gamma &\leq R\sqrt{M} + L_q^{\frac{1}{q}}(\boldsymbol{u}) M^{1-\frac{1}{q}}, \ \forall 1 \leq q \leq 2
\end{aligned}
$$

- Setting $q = 2$ in above, find an *explicit* upper bound on M, that is M must be only on the left hand side of the inequality.
  SOLUTION

$$
\begin{aligned}
M\gamma &\leq R\sqrt{M} + L_q^{\frac{1}{q}}(\boldsymbol{u}) M^{1-\frac{1}{q}}, \ \forall 1 \leq q \leq 2 \\
M\gamma &\leq R\sqrt{M} + \sqrt{L_2(\boldsymbol{u})}\sqrt{M}, \ \text{when q = 2} \\
M\gamma &\leq \sqrt{M}(R + \sqrt{L_2(\boldsymbol{u})}) \\
\sqrt{M} &\leq \frac{R + \sqrt{L_2(\boldsymbol{u})}}{\gamma} \\
M &\leq \left(\frac{R + \sqrt{L_2(\boldsymbol{u})}}{\gamma}\right)^2
\end{aligned}
$$

2.3 Question 5

Implement the Perceptron algorithm in Algorithm 1. The prototype of the function must be

```
[w,w_0] = train_perceptron(X, y)
```

where $X \in \mathbb{R}^{m\times d}$ is the matrix of m input vectors in d dimension, i.e. each input $\boldsymbol{x}_i$ is a row of X, y is a column vector of m columns containing the labels associated to the training samples. Learn `w_0` using the usual trick of augmenting the input data with a constant feature equal to 1.

```
function [w, w_0] = train_perceptron(X,Y)

  w_0 = 0;
  w = zeros(1, size(X,2));

  %Number of samples
  m = size(X,1);

  %One pass of the sample
  for sample = 1 : m
      if(sign(w * X(sample,:)') ~= Y(sample))
        w = w + X(sample,:) * Y(sample);
      end
  end

  %First element is the bias
  w_0 = w(1);
  w = w(2:size(w,2));
end
```

3 Support Vector Machines

3.1 Question 6

Consider training a SVM on a linearly separable dataset consisting of m points. Let n be the number of support vectors obtained by training on the entire set. Show that the LOOCV error is upper bounded by $\frac{n}{m}$. *Hint:* Consider two cases: (1) removing a support vector datapoint and (2) removing a non-support vector datapoint.

SOLUTION

Leave one out cross validation is particular version of Cross Validation. In this we held out a training data point and train the classifier on the remaining training data points. Test the resulting classifier on the held out data point. Let superscript (i) represent the parameters we would obtain while finding the linear separator without the i^{th} training example.

$$LOOCV = \frac{1}{m}\sum_{i=1}^{m} loss(y_i, f(x_i; w^{(i)}, w^{(i)}))$$

$$LOOCV = \frac{1}{m}(\sum_{i=1}^{n} loss(y_i, f(x_i; w^{(i)}, w^{(i)})) + \sum_{i=1}^{q} loss(y_i, f(x_i; w^{(i)}, w^{(i)})))$$

$$LOOCV = \frac{1}{numberofsamples}(\text{total loss of support vectors} + \text{total loss of non-support vectors})$$

Sample data points that lie outside the margin will be classified correctly. It does not depend on whether they are part of the training set or not. But support vectors define the linear separator because if the support vectors are removed from training set, then it may be miss-classified. Hence LOOCV is upper bounded by number from support vectors.

$$LOOCV \leq \frac{number of support vectors}{number of samples}$$
$$LOOCV \leq \frac{n}{m}$$

Hence proved.

3.2 Question 7

Here you have to implement the soft SVM classification problem in the primal form using Matlab quadratic solver.

Quadratic programs refer to optimization problems in which the objective function is quadratic and the constraints are linear. Many Machine Learning algorithms are reduced to solving quadratic programs. In this question, we will use the quadratic program solver of Matlab ('quadprog') to optimize the primal objective of a linear SVM.

For the primal objective function

$$\begin{aligned} &\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ &\text{s.t. } y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i,\ i = 1, \dots, m \\ &\qquad \xi_i \geq 0,\ i = 1, \dots, m\ . \end{aligned} \tag{18}$$

1. Cast the SVM primal objective as a quadratic program: Look at the `quadprog` function of Matlab, and write down what `H, f, A, b, Aeq, beq, lb, ub` are.

   Assuming the target variable is a vector containing first d entries as weights, then bias and finally last m entries as epsilons, where d is the dimension of data points and m is the number of samples.

   SOLUTION

   Following are values of variables

   $$F = [0_{1*d}, 0, 1_{1*m}]^T$$
   $$A = [X, 1_{m*1}]$$

   Then y_i is multiplied with its corresponding row.

   $$A = [x_1y_1, x_2y_2, x_3y_3, ...]$$

   Finally we append an identity matrix to A of size m to match matrix dimensions.

   $$A = [A, I_{d*d}]$$

   and finally for adjusting the sign of constant, -1 is multiplied

$$A = -1 * A$$

b is a column vector of all 1s.

$$b = -1 * 1_{m*1}$$

Aeq and Beq are null since there are no equalities in our system of constraints. lb is a column vector with $-\infty$ as first $d+1$ entries and 0 as last m entries.

$$lb = [-\infty_{1*(d+1)}, 0_{1,m}]^T$$

ub is all positive ∞

$$ub = [\infty_{1*(d+m+1)}]^T$$

2. Using the above answer, implement a function

   `[w,w_0] = train_svm_primal(X, y, C)`

   that solves the primal problem for SVMs, where $X \in \mathbb{R}^{m\times d}$ is the matrix of m input vectors in d dimension, i.e. each input $\boldsymbol{x}_i$ is a row of X, y is a column vector of m columns containing the labels associated to the training samples. C is the constant in the primal formulation (18).

```
function [w,w_0] = train_svm_primal(X, y, C)

[m, d] = size(X);

% create a (d+m+1) X (d+m+1) matrix for H
H = [ eye(d) zeros(d, m+1) ; zeros(m+1, d+m+1)];

% f represents the summation of xi and we multiply it with C as given in
% the equation of SVM primal form
f = C * [ zeros(d+1, 1) ; ones(m,1)];

% A is a (d+m+1) X (d+m+1) matrix
A = - [y* ones(1,d) .* X, y, eye(m)];

% lower bound on xi is 0
lb = [-inf*ones(d+1,1); zeros(m,1)];

% upper bound = inf
ub = inf * ones(d+m+1,1);

% b is -1
b = -1 * ones(m,1);

w = quadprog(H, f, A, b, [], [], lb, ub, []);

% extract bias
w_0 = w(d+1);
```

```
% extract w
w = w(1:d);
end
```

4 Empirical Results

4.1 Question 8

In the following we will use the synthetic data in synthdata.mat. It is a 2-d classification dataset composed by 2 classes.

We will use the implementations of Rigde Regression, SVM, and the Perceptron to train linear classifiers. Note that, while Ridge Regression is typically used for regression, nothing prevents us to use it as a classification algorithm. In this case, the training procedure goes on as usual, and in the testing phase we predict with the sign of the prediction.

- Set $\lambda = 1$, train Ridge Regression using the training data in synthdata.mat. Test the obtained solution on test data in synthdata.mat, and report the accuracy.

  The accuracy of ridge regression is 99.7985%.

- Set $C = 0.01$ in (18), use your implementation of the SVM and the training data in synthdata.mat. Test the obtained solution on test data in synthdata.mat, and report the accuracy, the objective value of the SVM, and the number of support vectors.

  The accuracy of SVM with $C = 0.01$ is 99.9664%.

- Repeat the above question with $C = 100$.

  The accuracy of SVM with $C = 100$ is 100%. The number of support vectors as we see from the figures plotted below is 3.

- Run the Perceptron with one pass over the training data. Test the obtained solution on test data in synthdata.mat, and report the accuracy.

  The accuracy of Perceptron is 100%.

- Plot in 2d the separating hyperplanes obtained in all the above cases and the training points.

Figure 1: Ridge Regression

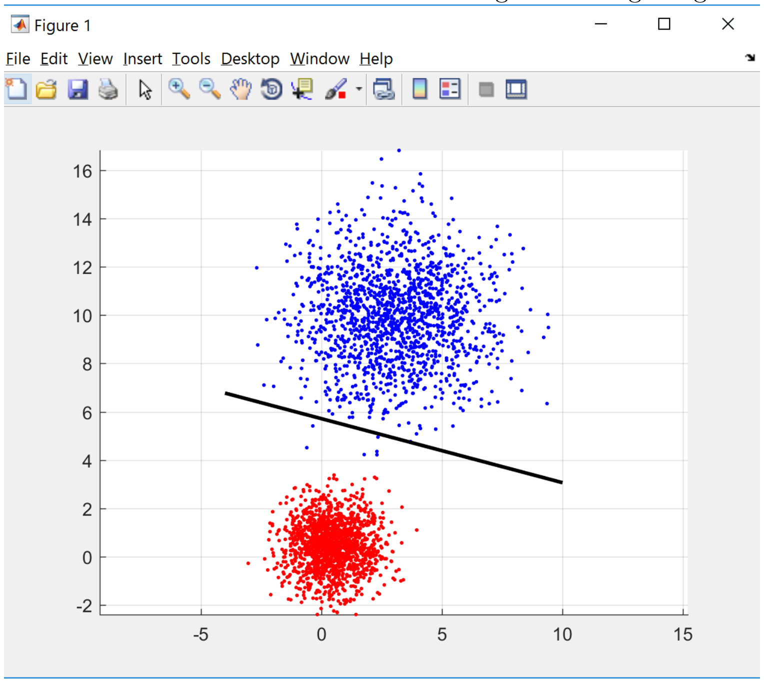

Figure 2: SVM C = 0.01

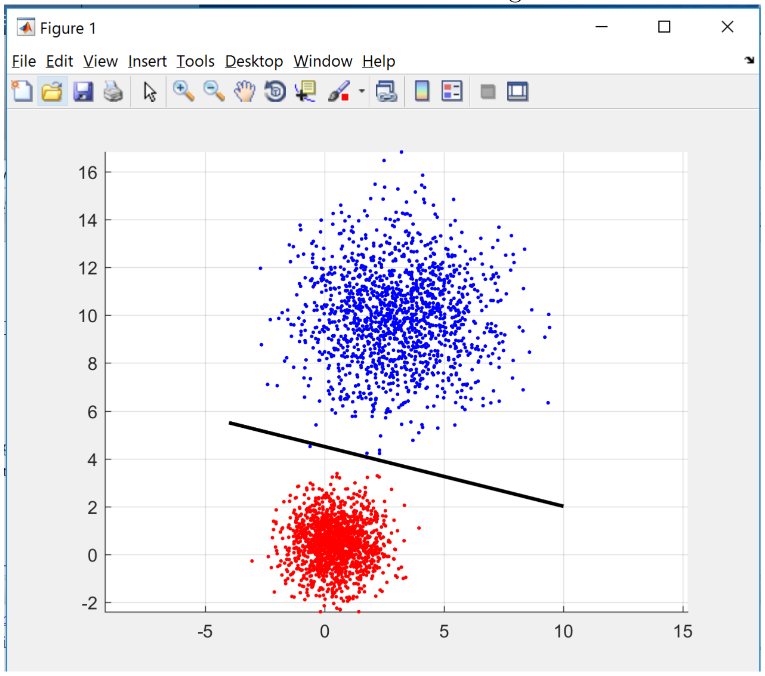

Figure 3: SVM C = 100

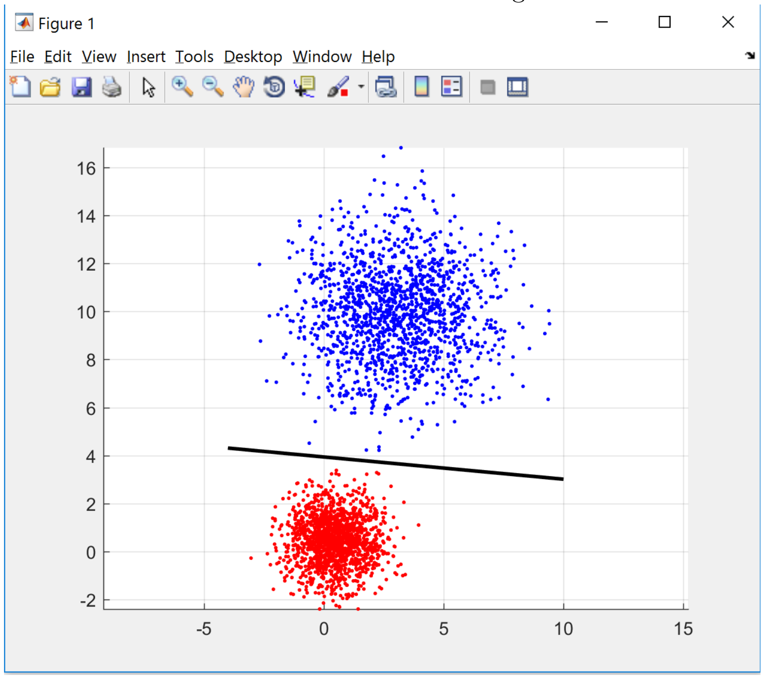

Figure 4: Perceptron

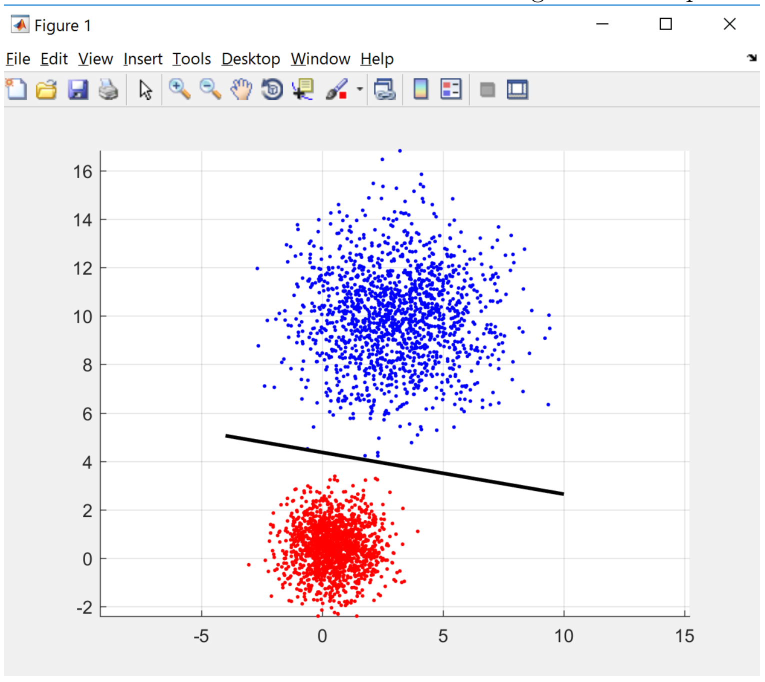
Figure 1
File Edit View Insert Tools Desktop Window Help