CSE512 HW2

Tim Zhang (110746199)

1 Ridge Regression

1.1 Question 1

1.1.1 $\bar{\boldsymbol{w}} = C^{-1}\boldsymbol{d}$

We want to minimize $\lambda\|\bar{\boldsymbol{w}}\|^2 + \sum_{i=1}^{m}(\bar{\boldsymbol{w}}^\top \boldsymbol{x_i} - y_i)^2$ with respect to $\bar{\boldsymbol{w}}$.

$$
\begin{aligned}
&\nabla_{\bar{\boldsymbol{w}}}\ \lambda\|\bar{\boldsymbol{w}}\|^2 + \sum_{i=1}^{m}(\bar{\boldsymbol{w}}^\top \boldsymbol{x_i} - y_i)^2 \\
&= \nabla_{\bar{\boldsymbol{w}}}\ \lambda\|\bar{\boldsymbol{w}}\|^2 + \|\bar{\boldsymbol{w}}^\top \bar{X} - \boldsymbol{y}\|^2 \\
&= 2\lambda\bar{\boldsymbol{w}} + 2\bar{X}(\bar{\boldsymbol{w}}^\top \bar{X} - \boldsymbol{y}) \\
&= 2(\lambda\bar{\boldsymbol{w}} + \bar{X}(\bar{\boldsymbol{w}}^\top \bar{X} - \boldsymbol{y})) \\
&= 2(\lambda\bar{\boldsymbol{w}} + (\bar{X}\bar{\boldsymbol{w}}^\top \bar{X} - \bar{X}\boldsymbol{y}))
\end{aligned}
$$

Finding the minimum of the gradient we compute:

$$
\begin{aligned}
2(\lambda\bar{\boldsymbol{w}} + (\bar{X}\bar{\boldsymbol{w}}^\top \bar{X} - \bar{X}\boldsymbol{y})) &= 0 \\
\lambda\bar{\boldsymbol{w}} + \bar{X}\bar{\boldsymbol{w}}^\top \bar{X} - \bar{X}\boldsymbol{y} &= 0 \\
\lambda\bar{\boldsymbol{w}} + \bar{X}\bar{\boldsymbol{w}}^\top \bar{X} &= \bar{X}\boldsymbol{y} \\
(\lambda\bar{I} + \bar{X}\bar{X}^\top)\bar{\boldsymbol{w}} &= \bar{X}\boldsymbol{y} \\
(\bar{X}\bar{X}^\top + \lambda\bar{I})\bar{\boldsymbol{w}} &= \bar{X}\boldsymbol{y} \\
\bar{\boldsymbol{w}} &= (\bar{X}\bar{X}^\top + \lambda\bar{I})^{-1}\bar{X}\boldsymbol{y} \\
\bar{\boldsymbol{w}} &= C^{-1}\boldsymbol{d}\ \blacksquare
\end{aligned}
$$

1.1.2 $C_{(i)}$ and $d_{(i)}$

$C_{(i)} = C - x_i x_i^\top$
$\boldsymbol{d}_{(i)} = \boldsymbol{d} - x_i y_i$

1.1.3 $C_{(i)}^{-1}$

Since $C_{(i)} = C - x_i x_i^\top$ and using the Sherman-Morrison formula we compute:

$$(C - x_i x_i^\top)^{-1} = C^{-1} - \frac{C^{-1}(-x_i x_i^\top)C^{-1}}{1 + (-x_i^\top C^{-1} x_i)}$$
$$C_{(i)}^{-1} = C^{-1} + \frac{C^{-1} x_i x_i^\top C^{-1}}{1 - x_i^\top C^{-1} x_i} \blacksquare$$

1.1.4 $\bar{w}_{(i)}$

$$\begin{aligned}
\bar{\boldsymbol{w}}_{(i)} &= C_{(i)}^{-1} d_{(i)} \\
&= C_{(i)}^{-1} \bar{X} y - \bar{\boldsymbol{x}}_i y_i \\
&= \left(C^{-1} + \frac{C^{-1}\bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^\top C^{-1}}{1 - \boldsymbol{x}_i^\top C^{-1} \bar{\boldsymbol{x}}_i} \right) \left(\bar{X} y - \bar{\boldsymbol{x}}_i y_i \right) \\
&= C^{-1}\bar{X} y - C^{-1}\bar{\boldsymbol{x}}_i y_i + \frac{C^{-1}\bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{X} y}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} - \frac{C^{-1}\bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i y_i}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} \\
&= \bar{\boldsymbol{w}} - C^{-1}\bar{\boldsymbol{x}}_i y_i + \frac{C^{-1}\bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{X} y}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} - \frac{C^{-1}\bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i y_i}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} \\
&= \bar{\boldsymbol{w}} - \frac{C^{-1}\bar{\boldsymbol{x}}_i y_i (1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i)}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} + \frac{C^{-1}\bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{X} y}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} - \frac{C^{-1}\bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i y_i}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} \\
&= \bar{\boldsymbol{w}} - \frac{(C^{-1}\bar{\boldsymbol{x}}_i y_i - C^{-1}\bar{\boldsymbol{x}}_i y_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i)}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} + \frac{C^{-1}\bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{X} y}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} - \frac{C^{-1}\bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i y_i}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} \\
&= \bar{\boldsymbol{w}} - \frac{C^{-1}\bar{\boldsymbol{x}}_i y_i - C^{-1}\bar{\boldsymbol{x}}_i y_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i + C^{-1}\bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{X} y - C^{-1}\bar{\boldsymbol{x}}_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i y_i}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} \\
&= \bar{\boldsymbol{w}} - \frac{C^{-1}\bar{\boldsymbol{x}}_i (y_i - y_i \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i + \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{X} y - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i y_i)}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} \\
&= \bar{\boldsymbol{w}} - \frac{C^{-1}\bar{\boldsymbol{x}}_i (y_i (1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i) + \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{X} y)}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} \\
&= \bar{\boldsymbol{w}} - \frac{C^{-1}\bar{\boldsymbol{x}}_i (y_i + \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{X} y)}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_i} \\
&= \bar{\boldsymbol{w}} - (C^{-1}\bar{\boldsymbol{x}}_i) \frac{y_i + \bar{\boldsymbol{x}}_i^\top \bar{\boldsymbol{w}}}{1 - \bar{\boldsymbol{x}}_i^\top C^{-1}\bar{\boldsymbol{x}}_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} \blacksquare
\end{aligned}$$

1.1.5 $\bar{w}_{(i)}^{\top}\bar{\boldsymbol{x}}_i - y_i$

$$
\begin{aligned}
\bar{w}_{(i)}^{\top}\bar{\boldsymbol{x}}_i - y_i &= \left(\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}\right)^{\top}\bar{\boldsymbol{x}}_i - y_i \\
&= \bar{\boldsymbol{w}}^{\top}\bar{\boldsymbol{x}}_i + \left((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}\right)^{\top}\bar{\boldsymbol{x}}_i - y_i \\
&= \bar{\boldsymbol{w}}^{\top}\bar{\boldsymbol{x}}_i + \left(\frac{\bar{\boldsymbol{w}}^{\top}\bar{\boldsymbol{x}}_i - y_i}{1 - \bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i}(\bar{\boldsymbol{x}}_i^{\top}(C^{-1})^{\top})\right)\bar{\boldsymbol{x}}_i - y_i \\
&= \bar{\boldsymbol{w}}^{\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}(\bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i) \\
&= \frac{(\bar{\boldsymbol{w}}^{\top}\bar{\boldsymbol{x}}_i - y_i)(1 - \bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i)}{1 - \bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i} + \frac{(\bar{\boldsymbol{w}}^{\top}\boldsymbol{x}_i - y_i)(\bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i)}{1 - \bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i} \\
&= \frac{(\bar{\boldsymbol{w}}^{\top}\bar{\boldsymbol{x}}_i - y_i)(1 - \bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i) + (\bar{\boldsymbol{w}}^{\top}\boldsymbol{x}_i - y_i)(\bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i)}{1 - \bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i} \\
&= \frac{(\bar{\boldsymbol{w}}^{\top}\bar{\boldsymbol{x}}_i - y_i) - (\bar{\boldsymbol{w}}^{\top}\bar{\boldsymbol{x}}_i - y_i)(\bar{\boldsymbol{x}}_i^{\top}C^{-1}\boldsymbol{x}_i) + (\bar{\boldsymbol{w}}^{\top}\bar{\boldsymbol{x}}_i - y_i)(\bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i)}{1 - \bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i} \\
&= \frac{\bar{\boldsymbol{w}}^{\top}\bar{\boldsymbol{x}}_i - y_i}{1 - \bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i} \blacksquare
\end{aligned}
$$

1.1.6 Analysis

Asymptotically $\bar{w}_{(i)}^{\top}\bar{\boldsymbol{x}}_i - y_i = \frac{\bar{\boldsymbol{w}}^{\top}\bar{\boldsymbol{x}}_i - y_i}{1 - \bar{\boldsymbol{x}}_i^{\top}C^{-1}\bar{\boldsymbol{x}}_i}$ is of complexity $O((d+1)^3)$ since inversion is the most expensive operation and $C \in \mathbb{R}^{d+1 \times d+1}$, this is the complexity we pay during the first computation. In each subsequent iteration we pay $O((d+1))$ for vector operations. Thus the total complexity is $O((d+1)^3)$.

In the normal case of LOOCV since we are computing the inverse over each training sample the total complexity is $O(m \times (d+1)^3)$.

1.2 Question 2

```
function [w,w_0] = train_rr(X, y, lambda)
    [m, d] = size(X);  % It is assumed that X does not have x_0 = 1 already
    I = eye(d + 1);    % Make the identity matrix
    one = ones(m, 1);  % Make x_0 = 1 vector
    X_bar = [one X];   % Append vector to input X

    % Compute minimum weights
    % NOTE: y is assumed to be input as a row vector from the command prompt
    w = inv((X_bar' * X_bar) + (lambda * I)) * (X_bar' * y);

    % Remove bias term from w
    w_0 = w(1);
    w = w(2:length(w));
end
```

2 Perceptron

2.1 Question 3

Assume that $\|\boldsymbol{x_i}\| \leq R$, $\forall i$ and that the data is linearly separable by some hyperplane defined by some $\boldsymbol{w}^*$ where $\|\boldsymbol{w}^*\| = 1$ and $y_i\langle \boldsymbol{w}^*, \boldsymbol{x}_i\rangle \geq \gamma$.

Let $\boldsymbol{w}_i$ be the weight vector at update i. So, we start with $\boldsymbol{w}_0 = \boldsymbol{0}$ and after the first update we have $\boldsymbol{w}_1$.

Then we can bound the rate at which $\|\boldsymbol{w}\|$ changes over each update as follows:

$$
\begin{aligned}
\|\boldsymbol{w}_{i+1}\|^2 &= \|\boldsymbol{w}_i + \eta y_i \boldsymbol{x}_i\|^2 \\
&= \sum_{i=1}^{d+1} (\boldsymbol{w}_i + \eta y_i \boldsymbol{x}_i)^2 \\
&= \sum_{i=1}^{d+1} \boldsymbol{w}_i^2 + 2\eta y_i \boldsymbol{w}_i \boldsymbol{x}_i + \eta^2 y^2 \boldsymbol{x}_i^2 \\
&= \sum_{i=1}^{d+1} \boldsymbol{w}_i^2 + 2\eta y_i \sum_{i=1}^{d+1} \boldsymbol{w}_i \boldsymbol{x}_i + \eta^2 y^2 \sum_{i=1}^{d+1} \boldsymbol{x}_i^2 \\
&= \|\boldsymbol{w}_i\|^2 + 2\eta y_i \langle \boldsymbol{w}_i \boldsymbol{x}_i\rangle + \eta^2 y^2 \|\boldsymbol{x}_i\|^2 \\
&\leq \|\boldsymbol{w}_i\|^2 + \eta^2 R^2
\end{aligned}
$$

Where the inequality follows since $y^2 = 1$ and $\|\boldsymbol{x}_i\| \leq R$ and the fact that $2\eta y_i \langle \boldsymbol{w}_i \boldsymbol{x}_i \rangle$ is negative since the example is misclassified.

So we have that $\|\boldsymbol{w}_{i+1}\|^2 \leq \|\boldsymbol{w}_i\|^2 + \eta^2 R^2$ which implies that $\|\boldsymbol{w}_M\|^2 \leq M\eta^2 R^2$ since each update increases $\|\boldsymbol{w}_i\|^2$ by at most $\eta^2 R^2$. Thus far we have concluded that $\|\boldsymbol{w}_M\| \leq \eta R\sqrt{M}$.

Now $\langle \boldsymbol{w}^*, \boldsymbol{w}_M \rangle \geq \gamma M$ since $\boldsymbol{w}^*$ defines a separating hyperplane which means that the PLA terminates in at least M steps and each margin is at least γ. Shown explicitly:

$$\langle \boldsymbol{w}^*, \boldsymbol{w}_M \rangle = \langle \boldsymbol{w}^*, \left(\sum_{i=1}^{M} y_i x_i \right) \rangle = \sum_{i=1}^{M} y_i \langle \boldsymbol{w}^*, x_i \rangle \geq \gamma M$$

Furthermore, since $\|\boldsymbol{w^*}\| = 1$ it is the case that $\langle \boldsymbol{w}^*, \boldsymbol{w}_M \rangle \leq \|\boldsymbol{w}_M\|$.

Thus, $\gamma M \leq \langle \boldsymbol{w}^*, \boldsymbol{w}_M \rangle \leq \|\boldsymbol{w}_M\| \leq \eta R\sqrt{M}$.

Solving for M we find:

$$\begin{aligned} \gamma M &\leq \eta R\sqrt{M} \\ \frac{\gamma M}{\sqrt{M}} &\leq \eta R \\ \sqrt{M} &\leq \frac{\eta R}{\gamma} \\ M &\leq \frac{\eta^2 R^2}{\gamma^2} \blacksquare \end{aligned}$$

As η increases the bound on M becomes looser and more iterations may be required. As η decreases the bound becomes tighter and less iterations may be required.

2.2 Question 4

2.2.1

In the separable case we have shown previously that $\gamma M \leq R\sqrt{M}$. By the definition of $\ell_i(\boldsymbol{u})$ it is the case that $y_i \langle \boldsymbol{u}, \boldsymbol{x}_i \rangle \geq \gamma$ and hence $\ell_i(\boldsymbol{u}) = 0$ if the sample $(\boldsymbol{x}_i, y_i)$ was correctly classified.

Then:

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

2.2.2 $q = 2$

$$\gamma M \leq M^{1/2} L_2^{1/2}(\boldsymbol{u}) + R\sqrt{M}$$
$$\gamma M \leq M^{1/2}(L_2^{1/2}(\boldsymbol{u}) + R)$$
$$\frac{M}{M^{1/2}} \leq \frac{(L_2^{1/2}(\boldsymbol{u}) + R)}{\gamma}$$
$$M^{1/2} \leq \frac{(L_2^{1/2}(\boldsymbol{u}) + R)}{\gamma}$$
$$M \leq \frac{(L_2^{1/2}(\boldsymbol{u}) + R)^2}{\gamma^2} \blacksquare$$

2.3 Question 5

```
%
% Implements the PLA running until convergence.
% The data is assumed to be linearly seperable.
%
function [w,w_0] = train_perceptron(X, y)
    [m, d] = size(X);     % It is assumed that X does not have x_0 = 1 already
    one = ones(m, 1);     % Make x_0 = 1 vector
    X_bar = [one X];      % Append vector to input X
    w = zeros(d + 1, 1);  % Initialize w = 0 vector

    while ~converged(w, X_bar, y, m)
        for i = 1:m
            if dot(w, X_bar(i, :)) * y(i) <= 0
                w = w + (y(i) * X_bar(i, :)');
            end
        end
    end

    % Remove bias term from w
    w_0 = w(1);
    w = w(2:length(w));
end

%
% Function iterates over all training examples to determine
% if PLA has converged.
%
function done = converged(w, X, y, m)
    for i = 1:m
        if dot(w, X(i, :)) * y(i) <= 0
            done = 0;
            return;
        end
    end

    done = 1;
    return;
end
```

3 Support Vector Machines

3.1 Question 6

If we remove a non-support vector datapoint the solution to SVM is not altered and the datapoint will still be correctly labeled. So it will be assumed that we only remove support vector datapoints. Denote by N the set of all support vectors and take $\mathcal{L}_{0-1}(\cdot)$ to be the 0-1 loss over $\bar{\boldsymbol{w}}, \bar{\boldsymbol{x}}, y$. Then we find:

$$\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}_{0-1}(\bar{\boldsymbol{w}}_{(i)}, \bar{\boldsymbol{x}}_i, y_i) = \frac{1}{m}\sum_{i\in N}\mathcal{L}_{0-1}(\bar{\boldsymbol{w}}_{(i)}, \bar{\boldsymbol{x}}_i, y_i) \leq \frac{n}{m} \blacksquare$$

3.2 Question 7

3.2.1 Quadratic Programming Formulation

First we will rewrite the primal optimization problem in standard form as

$$\begin{gathered}\min_{w} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} -(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) + \xi_i) \leq -1,\ i = 1, ..., m \\ -\xi_i \leq 0,\ i = 1, ..., m\end{gathered}$$

Then the arguments for quadprog are:

```
% The first d + 1 represents w and w_0 while the last m are the slack variables
H = diag([ones(1, d), zeros(1, m + 1)]);
% The f term only affects the slack variables
f = [zeros(1, d + 1), C * ones(1, m)];

% Break up the constraints into parts
A = -1 * [diag(y) * X, y, eye(m)];
b = -1 * ones(m, 1);

% There are no linear constraints
Aeq = [];
beq = [];

% Constrain slack variables to be greater than or equal to 0
lb = [-1 * inf(1, d + 1), zeros(1, m)];
ub = inf(1, d + 1 + m);
```

3.2.2 SVM Implementation

```
function [w,w_0] = train_svm_primal(X, y, C)
    [m, d] = size(X);

    % The first d + 1 represents w and w_0 while the last m are the slack variables
    H = diag([ones(1, d), zeros(1, m + 1)]);

    % The f term only affects the slack variables
    f = [zeros(1, d + 1), C * ones(1, m)];

    % Break up the constraints into parts
    A = -1 * [diag(y) * X, y, eye(m)];
    b = -1 * ones(m, 1);

    % There are no linear constraints
    Aeq = [];
    beq = [];

    % Constrain slack variables to be greater than or equal to 0
    lb = [-1 * inf(1, d + 1), zeros(1, m)];
    ub = inf(1, d + 1 + m);

    solution = quadprog(H, f, A, b, Aeq, beq, lb, ub);
    w = solution(1:d);
    w_0 = solution(d + 1);
end
```

4 Empirical Results

4.1 Question 8

Answers are directly copied from MATLAB Command Window session (implementation included in the last section).

4.1.1 Observed Metrics

```
Perceptron Loss
     0

Ridge Regression Loss
    0.0020

SVM C = .01 Loss
   3.3580e-04

SVM C = .01 Support Vector Count
     2

SVM C = 100 Loss
     0

SVM C = 100 Support Vector Count
     3
```

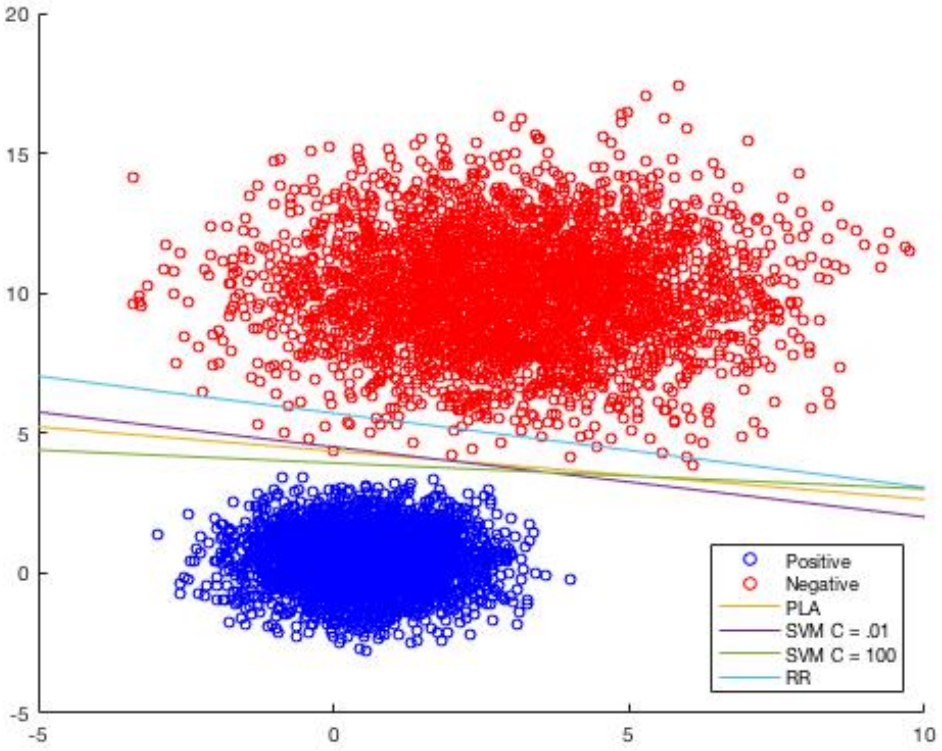

Figure 1: Training Data and Separating Hyperplanes

4.1.2 Implementation

```
%
% Trains algorithms and plots hyperplanes
%
function empirical_observations(Xtest, Xtrain, ytest, ytrain)
    % Split training data into classes
    [d, m] = size(ytrain);
    pos = [];
    neg = [];

    for i = 1: d
        if ytrain(i) == 1
            pos = [pos; Xtrain(i,:)];
        else
            neg = [neg; Xtrain(i,:)];
        end
    end

    % Train algorithms
    [w_perceptron, w_0_perceptron] = train_perceptron(Xtrain, ytrain);
    [w_rr, w_0_rr] = train_rr(Xtrain, ytrain, 1);
    [w_svm_01, w_0_svm_01] = train_svm_primal(Xtrain, ytrain, 0.01);
    [w_svm_100, w_0_svm_100] = train_svm_primal(Xtrain, ytrain, 10000);

    % Test algorithms
    loss_perceptron = test_algorithm(Xtest, ytest, w_perceptron, w_0_perceptron);
    loss_rr = test_algorithm(Xtest, ytest, w_rr, w_0_rr);
    loss_svm_01 = test_algorithm(Xtest, ytest, w_svm_01, w_0_svm_01);
    loss_svm_100 = test_algorithm(Xtest, ytest, w_svm_100, w_0_svm_100);

    % Display empirical errors
    disp('Perceptron Loss ');
    disp(loss_perceptron);

    disp('Ridge Regression Loss ');
    disp(loss_rr);

    disp('SVM C = .01 Loss ');
    disp(loss_svm_01);
    disp('SVM C = .01 Support Vector Count ');
```

```
    disp(count_support_vectors(Xtrain, w_svm_01, w_0_svm_01));

    disp('SVM C = 100 Loss ');
    disp(loss_svm_100);
    disp('SVM C = 100 Support Vector Count ');
    disp(count_support_vectors(Xtrain, w_svm_100, w_0_svm_100));

    % Define hyperplanes
    a_p = -1 * w_perceptron(1)/w_perceptron(2);
    b_p = -1 * w_0_perceptron/w_perceptron(2);
    x_p = -5:10;
    y_p = a_p * x_p + b_p;

    a_rr = -1 * w_rr(1)/w_rr(2);
    b_rr = -1 * w_0_rr/w_rr(2);
    x_rr = -5:10;
    y_rr = a_rr * x_rr + b_rr;

    a_svm_01 = -1 * w_svm_01(1)/w_svm_01(2);
    b_svm_01 = -1 * w_0_svm_01/w_svm_01(2);
    x_svm_01 = -5:10;
    y_svm_01 = a_svm_01 * x_svm_01 + b_svm_01;

    a_svm_100 = -1 * w_svm_100(1)/w_svm_100(2);
    b_svm_100 = -1 * w_0_svm_100/w_svm_100(2);
    x_svm_100 = -5:10;
    y_svm_100 = a_svm_100 * x_svm_100 + b_svm_100;

    % Plot points
    figure,
    scatter(pos(:,1), pos(:,2), 25, 'b'),
    hold on,
    scatter(neg(:,1), neg(:,2), 25, 'r'),
    plot(x_p, y_p),
    plot(x_svm_01, y_svm_01),
    plot(x_svm_100, y_svm_100),
    plot(x_rr, y_rr),
    hold off,
    legend('Positive', 'Negative', 'PLA', 'SVM C = .01', 'SVM C = 100', 'RR',
           'Location', 'southeast');
end
```

```
%
% Calculates the empirical risk of trained hypothesis
%
function loss = test_algorithm(X, y, w, w_0)
    [m, d] = size(X);  % Size of input matrix
    loss = 0;          % Value of loss

    % Sum over each training example
    for i = 1:m
        if y(i) * (dot(w, X(i, :)) + w_0) <= 0
            loss = loss + 1;
        end
    end

    loss = loss / m;
end

%
% Calculates the number of support vectors
%
function count = count_support_vectors(X, w, b)
    [m, d] = size(X);  % Size of input matrix
    count = 0;         % Number of support vectors
    min = 100;

    % Calculate the margin of each sample
    for i = 1:m
        margin = abs(dot(w, X(i, :)) + b);

        % Check in double precision range
        if margin <= 1.0001 && margin >= .9999
            count = count + 1;
        end
    end
end
```