CIS 520, Machine Learning, Fall 2018: Assignment 4

Wentao He

Collaborator: N/A

1 Convolutional Neural Network

1. Number of weights $= 108 \times 163 \times 3 = \boxed{52488}$
2. Number of weights $= 18 \times 6 \times 3 = \boxed{324}$
3. Number of neurons $= \dfrac{108-18}{6} + 1 = \boxed{16}$ and $\dfrac{162-6}{6} + 1 = \boxed{27}$ in x and y directions respectively. So the total number of neurons is $16 \times 27 = \boxed{432}$.
4. $$\text{output filter } 1 = \boxed{\begin{bmatrix} 8 & 9 & 10 \\ 11 & -5 & 13 \\ 2 & 3 & 11 \end{bmatrix}} \qquad \text{output filter } 2 = \boxed{\begin{bmatrix} 2 & 17 & 0 \\ 26 & -13 & 16 \\ 9 & 16 & -5 \end{bmatrix}}$$

2 Optimization and Lagrangian Duality

1. $\mathcal{L}(x_1, x_2, \nu) = \boxed{\frac{1}{2}(x_1^2 + x_2^2) + \nu(3x_1 + 2x_2 - 1)}$

2. $\frac{\partial \mathcal{L}}{\partial x_1} = x_1 + 3\nu = 0 \Rightarrow x_1 = -3\nu$

   $\frac{\partial \mathcal{L}}{\partial x_2} = x_2 + 2\nu = 0 \Rightarrow x_1 = -2\nu$

   $\therefore \phi(\nu) = \boxed{-\frac{13}{2}\nu^2 - \nu}$

3. $\frac{\mathcal{L}}{dv} = -13\nu - 1 = 0$

   $\nu^* = -\frac{1}{13}$

   $\therefore x_1^* = \boxed{\frac{3}{13}}, x_2^* = \boxed{\frac{2}{13}}$

3 Kernel Functions

1. $\phi(x) = \boxed{\sqrt{c} \cdot K_2(x, x')}$
2. Such a mapping cannot exist (not a valid kernel) because for any given c, we cannot guarantee $K > 0$.
3. $\phi(x) = \boxed{\begin{pmatrix} \phi_1(x) \\ \phi_2(x) \end{pmatrix}}$
4. $\phi(x) = \boxed{\begin{bmatrix} \phi_1(x) \\ \phi_1(x) \\ \vdots \\ \phi_1(x) \end{bmatrix} \circ \begin{bmatrix} \phi_{21}(x) \\ \vdots \\ \phi_{21}(x) \\ \vdots \\ \phi_{2d_2}(x) \\ \vdots \\ \phi_{2d_2}(x) \end{bmatrix}}$

   The left vector consists of d_2 rows of ϕ_1. The right vector consists of d_1 rows of ϕ_2. $\circ$ is the element-wise multiplication between the two vectors.
5. $\phi(x) = \boxed{\phi_1(f(x))}$

4 SVM and Neural Nets: Programming Exercise

4.1 SVM on synthetic data

1. (a) Cross Validation Error = $\boxed{0.0840}$

   (b) Training Error = $\boxed{0.0860}$

   (c) Test Error = $\boxed{0.1090}$

   For the plot, please see Figure 1.

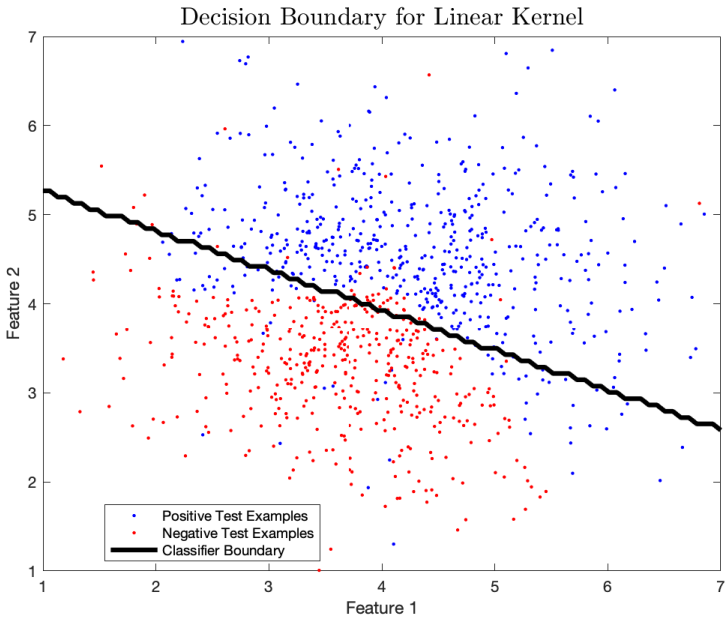

Figure 1: Decision Boundary for Linear Kernel.

2. (a) When $\boxed{\sigma = 0.1, C = 0.1}$ For the plot, please see Figure 2.

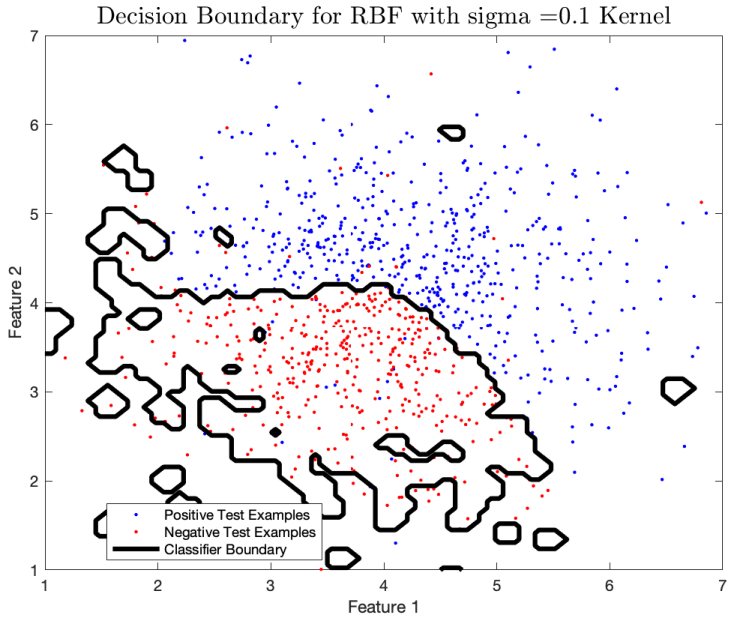

Figure 2: Decision Boundary for RBF Kernel with $\sigma = 0.1$.

(b) When $\boxed{\sigma = 1, C = 100}$ For the plot, please see Figure 3.

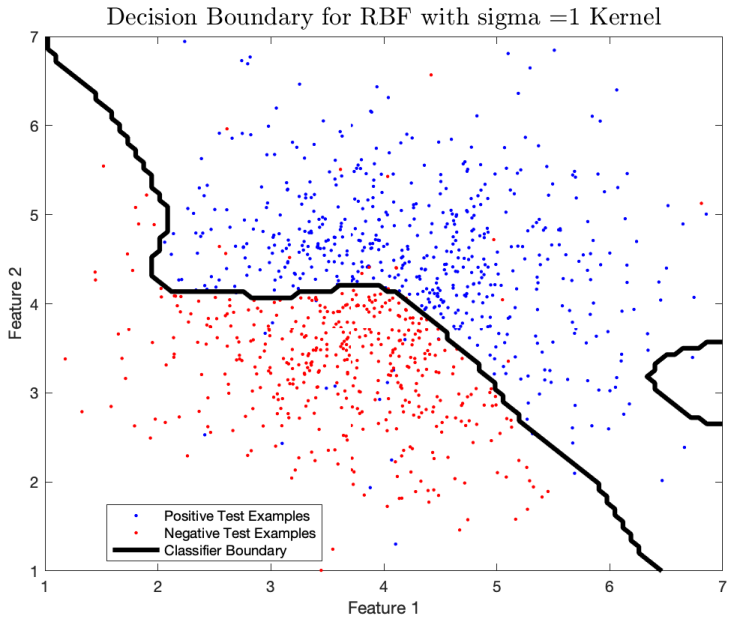

Figure 3: Decision Boundary for RBF Kernel with $\sigma = 1$.

(c) When $\boxed{\sigma = 10, C = 100000}$ For the plot, please see Figure 4.

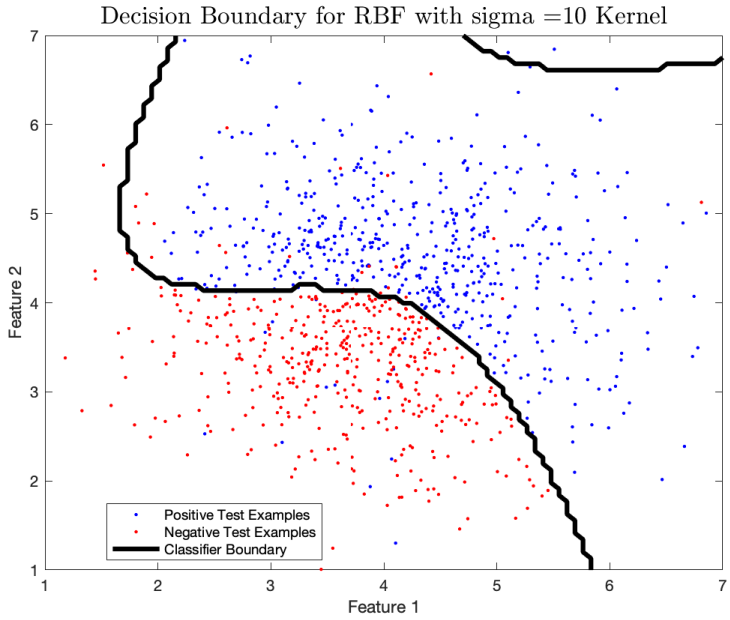

Figure 4: Decision Boundary for RBF Kernel with $\sigma = 10$.

(d) When $\boxed{\sigma = 100, C = 100000}$ For the plot, please see Figure 5.

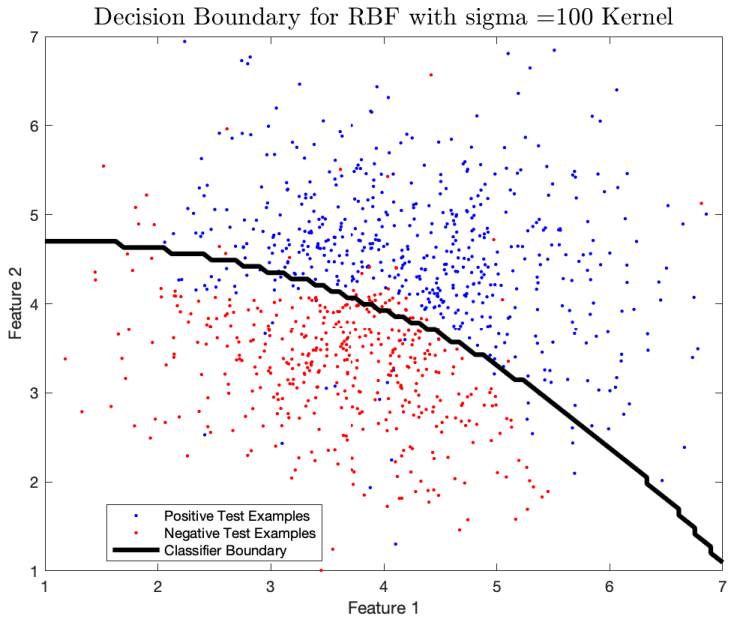

Figure 5: Decision Boundary for RBF Kernel with $\sigma = 100$.

(e) When $\boxed{\sigma = 1000, C = 100000}$ For the plot, please see Figure 6.

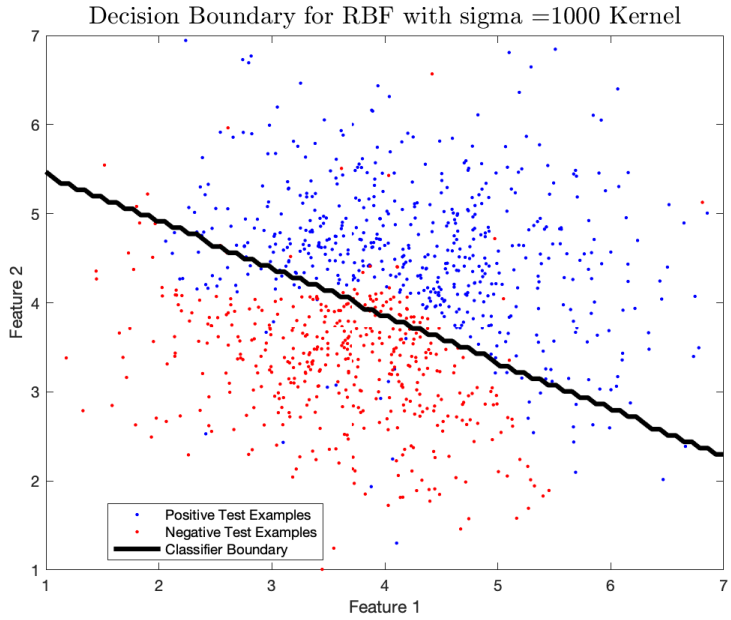

Figure 6: Decision Boundary for RBF Kernel with $\sigma = 1000$.

(f) insert line plot of errors For the plot, please see Figure 7.

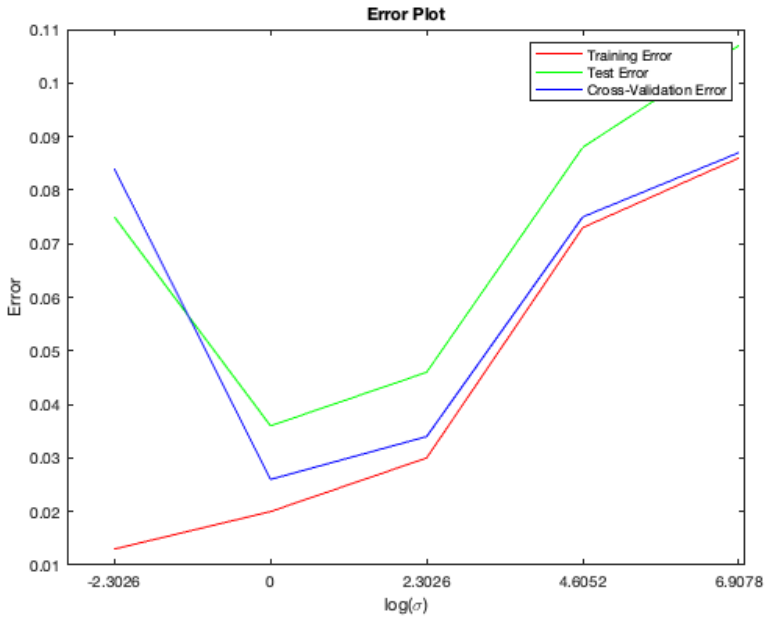

Figure 7: Error Plot.

(g) σ that achieves lowest test error is $\boxed{1}$.

(h) σ that achieves lowest cross-validation error is $\boxed{1}$.

3. (a) Absolute difference between test errors is $\boxed{0.01}$.

   (b) This difference in test errors is large. Selecting these parameters by cross-validation is a good approach because cross-validation error is very nearly unbiased for test error. The slight bias is due to that the training set in cross-validation is slightly smaller than the actual data set. In nearly all situations, the effect of this bias is conservative, i.e., the estimated fit is slightly biased in the direction suggesting a poorer fit. In practice, this bias is rarely a concern.

4.2 SVM and Neural Nets on Breast Cancer Dataset

1. Summary of SVM results

σ	Best C	Train Error	Cross-Validation Error
0.1	1	0	0.315
1	10	0	0.11667
10	1	0.025	0.03
100	10	0.026667	0.026667
1000	1000	0.026667	0.026667

A table of σ, the best value of C, train error and cross-validation error are shown in the chart above. When $\sigma = 0.1$ and $\sigma = 1$, their train errors are equal to 0, and their cross-validation errors are high as well, which means over-fitting. When $\sigma = 100$ and $\sigma = 1000$, their cross-validation errors reach minimum and the SVM provides the best result.

2. Summary of NN results

	Best C	Train Error	Cross-Validation Error
ReLU + MSE	0.01	0.0250	0.0283
ReLU + Cross Entropy	0	0.0433	0.0933
Sigmoid + MSE	0	0.0617	0.0433
Sigmoid + Cross Entropy	0.1	0.0417	0.0467

A table of different combination of networks, best value of C, the corresponding train error and cross-validation error are shown in the chart above. However, each value is different after every round. However, generally speaking, MSE provides a better result than cross entropy due to its low cross-validation error. Based on the chart, ReLU + MSE gives the best result.

4.3 Appendix

Code for 4.2.1

```
clc;
clear;

load('Breast-Cancer/trainingdata.mat');
load('Breast-Cancer/testdata.mat');

sigmas = [0.1,1,10,100,1000];
Boxconstrains = [1,10,100,1000,10000,100000];
Error = zeros(6,1);
train_err = zeros(5,1);
val_err = zeros(5,1);
Best_C = [];

for index = 1:size(sigmas, 2)
    sigma = sigmas(index)
    Error = zeros(6,1);
    for idx = 1:numel(Boxconstrains)
        err = 0;
        Boxconstrain = Boxconstrains(idx);
        for i = 1:5
            train_data = ['Breast-Cancer/CrossValidation/Fold',mat2str(i),'/cv-train.mat'];
            load(train_data);

            test_data = ['Breast-Cancer/CrossValidation/Fold',mat2str(i),'/cv-test.mat'];
            load(test_data);

            SVMModel = fitcsvm(cv_train(:,1:9),cv_train(:,10),'BoxConstraint',...
                Boxconstrain,'KernelFunction','rbf','KernelScale',sigma);

            labels= predict(SVMModel, cv_test(:,1:9));
            err = err + classification_error(labels, cv_test(:,10));
        end
        Error(idx,1) = err/5;
    end

    [Boxconstrains_min,column]=find(Error==min(min(Error)));
    if length(Boxconstrains_min) > 1
        Boxconstrains_min = Boxconstrains_min(1);
        column = column(1);
    end

    C = Boxconstrains(Boxconstrains_min)
    Best_C(end+1) = C;

    SVMModel = fitcsvm(train_inputs,train_labels,'BoxConstraint',...
        Boxconstrains(Boxconstrains_min),'KernelFunction','RBF','KernelScale',sigma);

    train_err(index,1) = classification_error(predict(SVMModel, train_inputs), train_labels)
    val_err(index,1) = Error(Boxconstrains_min,column)

end
```

```
sigmas = sigmas';
Best_C = Best_C';
table(sigmas, Best_C, train_err, val_err)

SVMModel = fitcsvm(train_inputs,train_labels,'BoxConstraint',10,'KernelFunction','RBF',...
    'KernelScale',100);
labels = predict(SVMModel, test_inputs);
labels = sign(labels);
save('SVMlabels.mat','labels');
```

Code for 4.2.2

```
clc;
clear;

load('Breast-Cancer/trainingdata.mat');
load('Breast-Cancer/testdata.mat');

Cs = [0,0.0001,0.001,0.01,0.1,1];
Cs_Err = zeros(6,1);

for index = 1:size(Cs, 2)
    C = Cs(index);
    net = feedforwardnet([10]);
    net.layers{1}.transferFcn = 'poslin';
    %net.layers{1}.transferFcn = 'logsig';
    net.performFcn = 'crossentropy';
    %net.performFcn = 'mse';
    net.performParam.regularization = C;
    net.trainFcn = 'trainrp';

    err = 0;
    for i = 1:5
        train_data = ['Breast-Cancer/CrossValidation/Fold',mat2str(i),'/cv-train.mat'];
        load(train_data);

        test_data = ['Breast-Cancer/CrossValidation/Fold',mat2str(i),'/cv-test.mat'];
        load(test_data);

        net = train(net,cv_train(:,1:9)',cv_train(:,10)');
        labels= net(cv_test(:,1:9)');

        err = err + classification_error(sign(labels), cv_test(:,10)');
    end
    Cs_Err(index,1) = err/5;

end

[Cs_min,column]=find(Cs_Err==min(min(Cs_Err)));

if length(Cs_min) > 1
    Cs_min = Cs_min(1);
    column = column(1);
end
```

```
Best_c = Cs(Cs_min)
val_error = Cs_Err(Cs_min,column)

net = feedforwardnet(10);
net.layers{1}.transferFcn = 'poslin';
%net.layers{1}.transferFcn = 'logsig';
net.performFcn = 'crossentropy';
%net.performFcn = 'mse';
net.performParam.regularization = Cs(Cs_min);
net.trainFcn = 'trainrp';

net = train(net,train_inputs',train_labels');
labels= net(test_inputs')';
labels = sign(labels);
save('NNlabels.mat','labels');

net = train(net,train_inputs',train_labels');
labels= net(train_inputs')';
labels = sign(labels);
train_error = classification_error(sign(labels), train_labels)
```