Homework 03

ECE 469/568 – Machine Learning

Date: 11/12/2024
Due data: 12/06/2024 by 11.59 PM
Section: SVM, K-means, PCA, and ANN
Instructions: Solutions must include codes, plots, and results, and discussions

1. [Training a SVM classifiers using a Gaussian Kernel based on MATLAB]

   In MATLAB, the function "fitcsvm" implementations dual soft-margin SVM problems, which are quadratic programming problems. For binary classification, the default solver is sequential minimal optimization (SMO).

   (a) Generate 200 points uniformly distributed in the unit disk by using the following Matlab script.

```
rng(1); % For reproducibility
r = sqrt(rand(200,1)); % Radius
theta = 2*pi*rand(200,1);  % Angle
data1 = [r.*cos(theta), r.*sin(theta)]; % Points
```

   (b) Use the following Matlab script to generate another data set consisting of 200 points uniformly distributed in the annulus.

```
r2 = sqrt(3*rand(200,1)+1); % Radius
theta2 = 2*pi*rand(200,1);       % Angle
data2 = [r2.*cos(theta2), r2.*sin(theta2)]; % points
```

   (c) Plot the points generated in "data1" and "data2". Also plot circles of radii 1 and 2 for comparison purposes. Use the following script to generate this plot.

```
figure;
plot(data1(:,1),data1(:,2),'r.','MarkerSize',15)
hold on
plot(data2(:,1),data2(:,2),'b.','MarkerSize',15)
ezpolar(@(x)1);ezpolar(@(x)2);
axis equal
hold off
```

   (d) Use the following script to combine the two data sets in one matrix to generate a vector for classifications.

```
data3 = [data1;data2];
theclass = ones(400,1);
theclass(1:200) = -1;
```

(e) Use the following script to train an SVM classifier with “KernelFunction” set to ’rbf’ and BoxConstraint set to Inf. Plot the decision boundary and flag the support vectors.

```
    %Train the SVM Classifier
    cl = fitcsvm(data3,theclass,'KernelFunction','rbf',...
    'BoxConstraint',Inf,'ClassNames',[-1,1]);

    % Predict scores over the grid
    d = 0.02;
    [x1Grid,x2Grid] = ...
       meshgrid(min(data3(:,1)):d:max(data3(:,1)),...
    min(data3(:,2)):d:max(data3(:,2)));
    xGrid = [x1Grid(:),x2Grid(:)];
    [¬,scores] = predict(cl,xGrid);

    % Plot the data and the decision boundary
    figure;
    h(1:2) = gscatter(data3(:,1),data3(:,2),theclass,'rb','.');
    hold on
    ezpolar(@(x)1);
    h(3) = plot(data3(cl.IsSupportVector,1), ...
    data3(cl.IsSupportVector,2),'ko');
    contour(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid)),[0 ...
       0],'k');
    legend(h,{'-1','+1','Support Vectors'});
    axis equal
    hold off
```

(f) Set the value of “BoxConstraint” to 1 and re-run the above script. Discuss your observations with respect to miss-classifications of points by your SVM.

2. [Training a SVM classifiers using a custom Kernel (Sigmoid kernel) based on MATLAB]

(a) Use the following script to generate a random set of points within the unit circle. This script also labels points in the first and third quadrants as belonging to the positive class, and those in the second and fourth quadrants in the negative class.

```
rng(1);  % For reproducibility
n = 100; % Number of points per quadrant

r1 = sqrt(rand(2*n,1));                      % Random radii
t1 = [pi/2*rand(n,1); (pi/2*rand(n,1)+pi)]; % Random angles ...
    for Q1 and Q3
X1 = [r1.*cos(t1) r1.*sin(t1)];              % ...
    Polar-to-Cartesian conversion
```

```
r2 = sqrt(rand(2*n,1));
t2 = [pi/2*rand(n,1)+pi/2; (pi/2*rand(n,1)-pi/2)]; % Random ...
    angles for Q2 and Q4
X2 = [r2.*cos(t2) r2.*sin(t2)];

X = [X1; X2];         % Predictors
Y = ones(4*n,1);
Y(2*n + 1:end) = -1; % Labels
```

(b) Plot the data points by using the following script.

```
    figure;
    gscatter(X(:,1),X(:,2),Y);
    title('Scatter Diagram of Simulated Data')
```

(c) Now, you are supposed to use a sigmoid kernel. Use the following script, which provides a Matlab function that accepts two matrices in the feature space as inputs, and transforms them into a Gram matrix using the sigmoid kernel.

```
        function G = mysigmoid(U,V)
        % Sigmoid kernel function with slope gamma and ...
            intercept c
        gamma = 1;
        c = -1;
        G = tanh(gamma*U*V' + c);
        end
```

(d) Next, train an SVM classifier using the above sigmoid kernel function also standardize the data before training. Use the following script to perform two above tasks.

```
                Mdl1 = ...
                    fitcsvm(X,Y,'KernelFunction','mysigmoid',...
                'Standardize',true);
```

(e) Use the following script to plot the data, and identify the support vectors and the decision boundary.

```
  % Compute the scores over a grid
d = 0.02; % Step size of the grid
[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...
min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];        % The grid
[¬,scores1] = predict(Mdl1,xGrid); % The scores

figure;
h(1:2) = gscatter(X(:,1),X(:,2),Y);
```

```
hold on
h(3) = plot(X(Mdl1.IsSupportVector,1),...
X(Mdl1.IsSupportVector,2),'ko','MarkerSize',10);
% Support vectors
contour(x1Grid,x2Grid,reshape(scores1(:,2),size(x1Grid)),[0 ...
    0],'k');
% Decision boundary
title('Scatter Diagram with the Decision Boundary')
legend({'-1','1','Support Vectors'},'Location','Best');
hold off
```

(f) Determine the out-of-sample misclassification rate by using 10-fold cross validation via the following script.

```
CVMdl1 = crossval(Mdl1);
misclass1 = kfoldLoss(CVMdl1);
misclass1
```

(g) We can adjust the kernel parameters in an attempt to improve the shape of the decision boundary. Consequently, this may also decrease the within-sample misclassification rate.

- Write "mysigmoid2" by modifying the parameter from "gamma = 1" to "gamma = 0.5" in the function which defines the Sigmoid kernel.

```
    function G = mysigmoid2(U,V)
    % Sigmoid kernel function with slope gamma and ...
        intercept c
    gamma = 0.5;
    c = -1;
    G = tanh(gamma*U*V' + c);
    end
```

- Modify the program in Part (e) to train another SVM classifier using the adjusted sigmoid kernel "mysigmoid2". Plot the data and the decision region, and determine the out-of-sample misclassification rate. Discuss your results and observations.

3. [Design of a support vector machine for an object recognition system]

   You are asked to build a support vector machine (SVM) to classify data and use cross-validation to find the best SVM kernel and regularization value.

   - Extract the support vector machine Matlab codes in svm.zip and refer to "Contents.m" for more information.

   - Clearly denote the various components and the function calls or scripts that execute your Matlab functions.
   - Save all figures in Matlab.

You are asked to test your SVM by using the following steps.

(a) To test your SVM, you will build a simple object recognition system. To this end, you need to load dataset2.mat, which contains a matrix $\mathbf{X}$ of size 100×4 and a vector $\mathbf{y}$ of size 100×1. The $\mathbf{X}$ matrix consists of 100 data points of dimensionality 4 that belong to one of two classes. Their actual labels are given in the $\mathbf{y}$ vector.

(b) Train your SVM by using four-fold cross-validation.

(c) Show performance of the SVM for linear, polynomial and Gaussian kernels with different settings of the polynomial order (d), σ and C values.

(d) Try to hunt for a good setting of these parameters to obtain high recognition accuracy.

(e) Plot training error and test error to depict the bias versus variance trade-offs. Discuss your results as much as possible by referring to the theory that you learned during the lectures.

[Hint]: Use linear scale for d values and log scale for σ and C values. When plotting the results convert the σ and C values into log scale and try to plot a 3D graphs using 'surf' and 'meshgrid' commands. For polynomial kernel, you can take d values as the x-axis, $\log_{10}(C)$ values as the y-axis, and the error as the z-axis. For Gaussian kernel, you can take $\log_{10}(\sigma)$ values as the x-axis, $\log_{10}(C)$ values as the y-axis, and the error as the z-axis.

Example:

```
d = 1:10;
C_array = logspace(-10,10,50);
sigma = logspace(-8,8,50);

% for plotting purposes
C_log = log10(C array);
[plot_d, plot_C] = meshgrid(C_log,d);
sigma_log = log10(sigma);
[plot_sigma, plot_C] = meshgrid(C_log,sigma_log);

figure(1)
surf(plot_d, plot_C, poly_train_error);
figure(2)
surf(plot_sigma, plot_C, Gauss_train_err);
```

4. [K-means clustering]

The Fisher's Iris data set is a multivariate data set used and made famous by the British statistician and biologist Ronald Fisher in his 1936. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

This data set embedded within Matlab and can be loaded, and petal lengths and widths can be extracted by using the following command.

```
load fisheriris
X = meas(:,3:4);
```

(a) Since the data set loaded into X is 2D, we can visualize it by plotting data points. Use the following script to plot the corresponding data points in 2D space.

```
figure;
plot(X(:,1),X(:,2),'k*','MarkerSize',5);
title 'Fisher''s Iris Data';
xlabel 'Petal Lengths (cm)';
ylabel 'Petal Widths (cm)';
```

(b) In your plot, the larger cluster seems to be split into a lower variance region and a higher variance region. This might indicate that the larger cluster is two, overlapping clusters. Thus, we need to set the number of clusters to be three ($K = 3$) in K-means algorithm.

Run K-means algorithm via the in-built Matlab function using the following script. The output "idx" will provide you a vector of predicted cluster indices corresponding to the observations in data set X. Here, C is a 3-by-2 matrix containing the final centroid locations.

```
    rng(1); % For reproducibility
[idx,C] = kmeans(X,3);
```

(c) Use the following script to compute the distance from each centroid to points on a grid. To do this, pass the centroids (C) and points on a grid to kmeans, and implement one iteration of the algorithm. [Hint: kmeans may display a warning stating that the algorithm did not converge, which you should expect since thi script only implements one iteration.]

```
x1 = min(X(:,1)):0.01:max(X(:,1));
x2 = min(X(:,2)):0.01:max(X(:,2));
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:),x2G(:)]; % Defines a fine grid on the plot
idx2Region = kmeans(XGrid,3,'MaxIter',1,'Start',C);
```

(d) Use the following script to plot the cluster regions for visualization purposes.

```
figure;
gscatter(XGrid(:,1),XGrid(:,2),idx2Region,...
[0,0.75,0.75;0.75,0,0.75;0.75,0.75,0],'..');
```

```
hold on;
plot(X(:,1),X(:,2),'k*','MarkerSize',5);
title 'Fisher''s Iris Data';
xlabel 'Petal Lengths (cm)';
ylabel 'Petal Widths (cm)';
legend('Region 1','Region 2','Region ...
    3','Data','Location','SouthEast');
hold off;
```

(e) Next, extract the length and the width of the sepals from the data set "fisheriris". Follow the above procedure to run K-means algorithm and plot the cluster regions.

5. [Principle Component Analysis (PCA)]

Assume that you have been given a data matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, in which there are m examples each having n features. The variation of data projected into a vector $\mathbf{u}$ is captured by $\mathbb{V}ar[\mathbf{uX}] = \mathbf{u}^T\mathbf{C}_X\mathbf{u}$, where $\mathbf{C}_x = \mathbb{E}[\mathbf{X}\mathbf{X}^T]$ is the statistical covariance matrix of the data matrix $\mathbf{X}$.

The first principle component of the data matrix can be derived by solving the following constrained optimization problem:

$$\begin{array}{cc} \max_{\mathbf{u}} & \mathbf{u}^T\mathbf{C}_X\mathbf{u} \\ \text{subject to} & ||\mathbf{u}|| = 1 \end{array}$$

(a) By using the generalized Lagrangian function and solving the dual optimization problem, show that the first principle component of $\mathbf{X}$ is the eigenvector of $\mathbf{C}_X$ that corresponds to its largest eigenvalue.

(b) Thereby, verify that a reduced dimensional data set $\mathbf{Z} \in \mathbb{R}^{k\times m}$ can be obtained by using

$$\mathbf{Z} = \mathbf{U}_{\text{red}}^T\mathbf{X},$$

where $\mathbf{U}_{\text{red}} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k]$ and $\mathbf{u}_i$ for $i \in \{1, \cdots, k\}$ is the eigenvector that corresponds to the ith eigenvalue.

(c) If $\mathbf{A} = \mathbf{U}_{\text{red}}\mathbf{Z}$, then show that $\mathbf{A} \neq \mathbf{X}$.

(d) Assume that a matrix $\mathbf{B}$ is obtained as follows:

$$\mathbf{B} = \mathbf{U}^T\mathbf{X},$$

where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n]$ and $\mathbf{u}_i$ is the eigenvector corresponding to the ith eigenvalue of $\mathbf{C}_X$ for $i \in \{1, \cdots, n\}$,

i. Show that the covariance matrix of $\mathbf{B}$ is a diagonal matrix.

ii. If $\hat{\mathbf{X}} = \mathbf{UB}$, then show that $\hat{\mathbf{X}} = \mathbf{X}$.

(e) The Matlab data set "hald" consists of data related to "Heat of cement versus mix of ingredients". Load this data set using "load hald" command and answer the following questions:

i. In Matlab let $\mathbf{X}$ = ingredients. Find all principle components of $\mathbf{X}$ by first making sure that X is zero mean by taking out its sample mean, constructing the sample covariance matrix, and taking the eigenvectors of this sample covariance matrix.

ii. Then use the following in-built Matlab command to find the principle components of $\mathbf{X}$. The columns of Mat_pc correspond to four principal components.

```
      Mat_pc = pca(ingredients)
```

(f) Recall that during the lectures, we studied PCA as a means to reduce dimensionality, by minimizing the squared error of predictions in lower dimension or projections onto a range space, as compared to the actual values.

In this question, you are asked to prove that this problem is equivalent to maximizing the squared length of these projections or maximizing variance of projected data.

Consider n feature vectors of the form $\mathbf{x} \in \mathbb{R}^d$. Using only k basis vectors, we want to project $\mathbf{x}$ in a new space of lower dimensionality, say from $\mathbb{R}^d$ to $\mathbb{R}^k$:

$$\mathbf{z} = \mathbf{W}^T\mathbf{x},$$

where $\mathbf{W}$ is the orthogonal mapping matrix of size $d \times k$.

To obtain an approximation of $\mathbf{x}$, you can the following reconstruction:

$$\hat{\mathbf{x}} = \mathbf{W}\mathbf{z}.$$

Note that $\mathbf{W}\mathbf{W}^T = \mathbf{P}$ is a projection matrix, which is Idempotent: $\mathbf{P} = \mathbf{P}^T = \mathbf{P}^2$.

Our objective is to minimize the sum of squared error during reconstruction, which can be written as

$$\min ||\mathbf{x} - \hat{\mathbf{x}}||^2$$

i. Show that the sum of squared error, $||\mathbf{x} - \hat{\mathbf{x}}||^2$, is minimized when $\mathbf{P} = \mathbf{W}\mathbf{W}^T$.

ii. Run PCA on the following input matrix $\mathbf{X}$ to reduce the number of dimensions to $k = 1$, where $n = 6$ and $d = 2$, such that $\mathbf{x} \in \mathbb{R}^d$and $\hat{\mathbf{x}} \in \mathbb{R}^k$.

$$\mathbf{X} = \begin{bmatrix} 2 & 4 & 5 & 5 & 3 & 2 \\ 2 & 3 & 4 & 5 & 4 & 3 \end{bmatrix}$$

6. [Image compression using PCA]

In this question, you will work with a standard test image ("peppers.png") that is often used in image processing and image compression.

Use the following script to implement image compression using PCA.

```
[RGB map] = imread('peppers.png');
imshow(RGB)
peppers_gray = rgb2gray(RGB);
save peppers_gray
figure
imshow(peppers_gray)
peppers_gray=double(peppers_gray); % convert to double precision
axis off, axis equal
X = peppers_gray; %raw data matrix
[m, n] = size(X);
mean_X = mean(X,2); % compute row mean
reformat_mean = repmat(mean_X,1,n);
tilde_X = X - reformat_mean; % subtract row mean to obtain X
covX = tilde_X*tilde_X'/(n-1); %Sample covariance matrix of X
[U,S] = eig(covX);
[Ordered_eigValue,ind] = sort(diag(S),'descend');
U_od = U(:,ind);
variances = (Ordered_eigValue); % compute variances
figure
bar(variances(1:30)) %   plot of variances
%% Extract first 40 principal components
PCs = 300; % Number of principle components used for compression
U_red = U(:,1:PCs);
Z = U_red'*tilde_X; % project data onto PCs
X_hat = U_red*Z; % convert back to original basis
figure
imshow(X_hat), axis off; % display results
```

(a) Use different values for the number of principle components and observe the quality of the compressed image. Comment on the image quality provided via image compression with PCA.

7. [Design of a feed-forward artificial neural network for classification]

You are asked to design a feed-forward neural network for binary classification in Matlab using gradient descent. Your code should accept a data-set $\{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(i)}, y^{(i)}), \cdots (\mathbf{x}^{(N)}, y^{(N)}))\}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^3$ (with $x_0^{(i)} = 1$) and $y^{(i)} \in \{0, 1\}$, and train weights $\mathbf{w}_{j,i}^{(l)}$ for the classification problem. All hidden units and the output unit are activated by using logistic sigmoid functions of the following form

$$\sigma(z) = \frac{1}{1 + \exp(-z)}.$$

Your algorithm must minimize the following cross-entropy cost function at the output unit together with the regularization term [Hint: refer to the cost function that you were taught at the class].

(a) Plot the data points in dataset1.mat and identify the feature set and the output variables. [Hint: Use "load dataset1," and you will have the variables $\mathbf{x}$ (input vectors) and y (binary labels) in your Matlab environment which contain the data-set. For Python-based implementations, transform the file format of dataset1.mat to a suitable format.]

(b) Design the architecture of your feed-forward neural network that is suitable for training examples in dataset1.mat. You should have at least 2 hidden layers, and each hidden layer must have at least 20 units.

(c) You are asked to develop a Matlab/Python script to implement

- Feed-forward activations of hidden units and output units
- Compute the cost function
- Backpropagation to find all partial derivatives
- Gradient descent algorithm to minimize the cost function

(d) Use your code to learn a classification function for the data-set in "dataset1.mat".

(e) Plot the training and test errors against the number of iterations to show the convergence of your algorithm for suitable learning rates.

(f) Interpret your plot in terms of machine learning terminology, including generalization gap, learning rate, etc.