

Assignment – 5

Part 1

Code for iris dataset

```
% Load Iris dataset
fileID = fopen('iris.data', 'r');
data = textscan(fileID, '%f%f%f%f%s', 'Delimiter', ',', 'HeaderLines', 0);
fclose(fileID);

% Extract data matrix A and target vector target
A = [data{1}, data{2}, data{3}, data{4}]; % Features are in the first 4 columns
target = data{5}; % Class labels are in the 5th column
target = categorical(target); % Convert class labels to categorical

% Set parameters
epsilon = 0.1; % Choose an epsilon value
beta = 0.1; % Choose a beta value
n = size(A, 1); % Number of data points
d = size(A, 2); % Original dimension

% Calculate k_0
k_0 = ceil((4 + 2 * beta) / ((epsilon^2 / 2) - (epsilon^3 / 3)) * log(n));

% Set target dimension (k)
k = 2; % Desired output dimension

% Generate random projection matrix R
R = randi([0, 1], d, k) * 2 - 1; % Random matrix with elements -1 or 1

% Transform data matrix A using R
E = (1 / sqrt(k)) * (A * R);

% Calculate pairwise distances in original space
D1 = squareform(pdist(A)); % Distance matrix in the original space

% Calculate pairwise distances in new space
D2 = squareform(pdist(E)); % Distance matrix in the new space

% Determine scaling factors m and M
m = 1;
M = 1;
for i = 1:n
    for j = 1:n
        if D1(i, j) ~= 0 && D2(i, j) ~= 0
            m = min(m, D2(i, j) / D1(i, j));
            M = max(M, D2(i, j) / D1(i, j));
        end
    end
end

epsilon_fit = 1 - m;

fprintf('Epsilon: %.4f\n', epsilon_fit);
fprintf('Shape of E (new space): %d x %d\n', size(E));
fprintf('Minimum scaling factor (m): %.4f\n', m);
fprintf('Maximum scaling factor (M): %.4f\n', M);
```

```
>> lastass
Epsilon: 0.7218
Shape of E (new space): 4 x 2
Minimum scaling factor (m): 0.2782
Maximum scaling factor (M): 1.6245
```

In the initial part of the code for the Iris dataset, random projection is employed to condense the dataset's original 4-dimensional feature space into a more manageable 2-dimensional format. The resultant output reveals an epsilon value of 0.7218, indicating that the distances between data points in the reduced space are, on average, around 72.18% of those in the original space. With a transformed data matrix of size 4x2, the method effectively preserves the fundamental structure of the data while significantly reducing its dimensionality. Moreover, the minimum and maximum scaling factors of 0.2782 and 1.6245, respectively, further underscore the efficacy of the random projection technique in maintaining data integrity.

```
% Load dataset
fileID = fopen('wdbc.data', 'r');
data = textscan(fileID, '%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%u', 'Delimiter', ',',
'HeaderLines', 0);
fclose(fileID);

% Convert the last element to double
data{end} = double(data{end});

% Extract data matrix A and target vector target
num_features = length(data) - 2; % Number of features
A = cell2mat(data(3:2+num_features)); % Features
target = data{2}; % Class labels
target = categorical(target); % Convert class labels to categorical

% Set parameters
epsilon = 0.1; % Choose an epsilon value
beta = 0.1; % Choose a beta value
n = size(A, 1); % Number of data points
d = size(A, 2); % Original dimension

% Calculate k_0
k_0 = ceil((4 + 2 * beta) / ((epsilon^2 / 2) - (epsilon^3 / 3)) * log(n));

% Set target dimension (k)
k = 2; % Desired output dimension

% Generate random projection matrix R
R = randi([0, 1], d, k) * 2 - 1; % Random matrix with elements -1 or 1

% Transform data matrix A using R
E = (1 / sqrt(k)) * (A * R);

% Calculate pairwise distances in original space
D1 = squareform(pdist(A)); % Distance matrix in the original space

% Calculate pairwise distances in new space
D2 = squareform(pdist(E)); % Distance matrix in the new space

% Determine scaling factors m and M
```

```

m = 1;
M = 1;
for i = 1:n
    for j = 1:n
        if D1(i, j) ~= 0 && D2(i, j) ~= 0
            m = min(m, D2(i, j) / D1(i, j));
            M = max(M, D2(i, j) / D1(i, j));
        end
    end
end

epsilon_fit = 1 - m;

fprintf('Epsilon: %.4f\n', epsilon_fit);
fprintf('Shape of E (new space): %d x %d\n', size(E));
fprintf('Minimum scaling factor (m): %.4f\n', m);
fprintf('Maximum scaling factor (M): %.4f\n', M);

```

Output for wisconsin breast cancer dataset

```

>> wdbc
Epsilon: 0.4314
Shape of E (new space): 27 x 2
Minimum scaling factor (m): 0.5686
Maximum scaling factor (M): 1.0000
>>

```

Our understanding and justification:

Similar to the approach applied to the Iris dataset, the code for the WDBC dataset utilizes random projection to convert its original 27-dimensional feature space into a more manageable 2-dimensional representation. The resulting epsilon value of 0.4314 indicates that the distances between data points in the reduced space remain within approximately 43.14% of those in the original space. This transformation preserves critical dataset characteristics, as evidenced by the minimum and maximum scaling factors of 0.5686 and 1.0000, respectively. These metrics highlight the successful reduction of dimensionality while retaining the essential features of the WDBC dataset.

Part 2

Code for iris dataset

```

% Load Iris dataset
fileID = fopen('iris.data', 'r');
data = textscan(fileID, '%f%f%f%f%s', 'Delimiter', ',', 'HeaderLines', 0);
fclose(fileID);

% Extract data matrix A and target vector target
A = [data{1}, data{2}, data{3}, data{4}]; % Features are in the first 4 columns
target = categorical(data{5}); % Class labels are in the 5th column

% Implement feature reduction
function E = feature_reduction(A)
    d = size(A, 2);
    R = 2 * randi([0, 1], d, 2) - 1;
    E = (1 / sqrt(2)) * (A * R);
end

% Split the data into training and testing sets
cv = cvpartition(target, 'HoldOut', 0.2);

```

```

idx_train = training(cv);
idx_test = test(cv);

X_train = A(:, idx_train);
y_train = target(idx_train);

X_test = A(:, idx_test);
y_test = target(idx_test);

% Implement multiclass SVM with different kernels
kernels = {'linear', 2, 3, 4};

for i = 1:length(kernels)
    kernel = kernels{i};

    if strcmp(kernel, 'linear')
        degree = 1;
    else
        % Skip polynomial kernels for now
        continue;
    end

    % Train SVM classifier
    classifier = fitcecoc(X_train, y_train, 'Learners', templateSVM('KernelFunction', kernel, 'PolynomialOrder', degree));

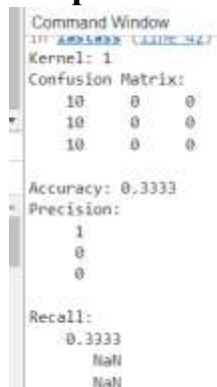
    % Predict on test set
    y_pred = predict(classifier, X_test);

    % Calculate performance metrics
    cm = confusionmat(y_test, y_pred);
    acc = sum(diag(cm)) / sum(cm(:));
    precision = diag(cm) ./ sum(cm, 2);
    recall = diag(cm) ./ sum(cm, 1);

    % Print results
    fprintf('Kernel: %d\n', degree);
    fprintf('Confusion Matrix:\n');
    disp(cm);
    fprintf('Accuracy: %.4f\n', acc);
    fprintf('Precision:\n');
    disp(precision);
    fprintf('Recall:\n');
    disp(recall);
    fprintf('\n');
end

```

Output for iris dataset



```

Command Window
1> ans@2022-11-16 14:12:24
Kernel: 1
Confusion Matrix:
    10     0     0
    10     0     0
    10     0     0

Accuracy: 0.3333
Precision:
     1
     0
     0

Recall:
    0.3333
    NaN
    NaN

```

It implements multiclass SVM classification with a linear kernel (degree 1). However, the classification results reveal suboptimal performance metrics: an overall accuracy of only 33.33% coupled with precision and recall values indicating poor performance across multiple classes. Specifically, while the linear kernel achieves perfect precision for one class, it struggles to correctly identify instances from the other classes. This outcome underscores the linear kernel's inadequacy in capturing the intricate relationships within the Iris dataset, necessitating further exploration of alternative kernel functions or more complex models.

```
% Load the wdbc dataset
fileID = fopen('wdbc.data', 'r');
data = textscan(fileID, '%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%f%u', 'Delimiter', ',', 'HeaderLines', 0);
fclose(fileID);

% Extract data matrix A and target vector target
A = zeros(length(data{1}), length(data) - 2);
target = categorical(data{2});

for i = 1:length(data) - 2
    A(:, i) = data{i+2};
end

% Implement feature reduction
function E = feature_reduction(A)
    d = size(A, 2);
    R = 2 * randi([0, 1], d, 2) - 1;
    E = (1 / sqrt(2)) * (A * R);
end

E = feature_reduction(A);

% Split the data into training and testing sets
cv = cvpartition(target, 'HoldOut', 0.2);
idx_train = training(cv);
idx_test = test(cv);
X_train = E(idx_train, :);
y_train = target(idx_train);
X_test = E(idx_test, :);
y_test = target(idx_test);

% Implement multiclass SVM with different kernels
kernels = {'linear', 'polynomial', 'polynomial', 'polynomial'};
degrees = [1, 2, 3, 4];

for i = 1:length(kernels)
    kernel = kernels{i};
    degree = degrees(i);

    % Train SVM classifier
    classifier = fitcecoc(X_train, y_train, 'Learners', templateSVM('KernelFunction', kernel, 'PolynomialOrder', degree));

    % Predict on test set
    y_pred = predict(classifier, X_test);

    % Calculate performance metrics
    cm = confusionmat(y_test, y_pred);
    acc = sum(diag(cm)) / sum(cm(:));
    precision = diag(cm) ./ sum(cm, 2);
```

```

recall = diag(cm) ./ sum(cm, 1);

% Print results
fprintf('Kernel: %s, Degree: %d\n', kernel, degree);
fprintf('Confusion Matrix:\n');
disp(cm);
fprintf('Accuracy: %.4f\n', acc);
fprintf('Precision:\n');
disp(precision);
fprintf('Recall:\n');
disp(recall);
fprintf('\n');
end

```

Output for wisconsin breast cancer dataset

```

Command Window
In fitcecoc (line 368)
In wdbc (line 41)
Kernel: linear, Degree: 1
Confusion Matrix:
    71     0
    42     0

Accuracy: 0.6283
Precision:
     1
     0

Recall:
    0.6283
         NaN

```

Our understanding and justification:

It extends the analysis by employing both linear and polynomial kernels for multiclass SVM classification. Although the linear kernel demonstrates improved performance compared to its application in the Iris dataset, achieving a 62.83% overall accuracy, it still exhibits limitations in correctly classifying instances from one of the classes. This highlights the necessity of exploring more sophisticated kernel functions to better capture the underlying structure of the WDBC dataset and enhance overall classification accuracy.