```
NON_SEIZURE_SET = dlmread('../data/training_set_n.txt')'; % Label = 0
SEIZURE_SET = dlmread('../data/training_set_s.txt')'; % Label = 1
TEST_SET = dlmread('../data/test_set.txt')';
TEST_LABELS = dlmread('../data/test_labels.txt')';

% Run all classifiers with 5-fold stratified crossvalidation
[folds0, folds1] = make_cv(NON_SEIZURE_SET, SEIZURE_SET, TEST_SET, TEST_LABELS);

results_per_fold = zeros(3, 5, 2, 2);

results_per_fold(1,:,:,:) = run_cv(folds0, folds1, @simple_train, @simple_classify);
results_per_fold(2,:,:,:)  = run_cv(folds0, folds1, @linearsvm, @linear_classify);
results_per_fold(3,:,:,:)  = run_cv(folds0, folds1, @nonlinearsvm,↙
@nonlinear_classify);

cv_accuracies = zeros(3,5);
cv_tp_rate = zeros(3,5);
cv_fp_rate = zeros(3,5);

avg_accuracies = zeros(1,3);
avg_tp_rate = zeros(1,3);
avg_fp_rate = zeros(1,3);

for i=1:3
    for j=1:5
        tp = results_per_fold(i,j,2,2);
        tn = results_per_fold(i,j,1,1);
        fp = results_per_fold(i,j,2,1);
        fn = results_per_fold(i,j,1,2);

        cv_accuracies(i,j) = (tp+tn)/(tp+tn+fp+fn);
        cv_tp_rate(i,j) = (tp)/(tp+fn);
        cv_fp_rate(i,j) = (fp)/(fp+tn);

        avg_accuracies = mean(cv_accuracies');
        avg_tp_rate = mean(cv_tp_rate');
        avg_fp_rate = mean(cv_fp_rate');
    end


end

% Run all classifiers on the given training set and test set
results = zeros(3, 2, 2);

[avg_0, avg_1] = simple_train(NON_SEIZURE_SET, SEIZURE_SET);
simple_predicted_labels = simple_classify(TEST_SET, avg_0, avg_1);
results(1,:,:) = confusionmat(TEST_LABELS, simple_predicted_labels');

[linear_a, linear_b] = linearsvm(NON_SEIZURE_SET, SEIZURE_SET);
```

```
linear_predicted_labels = linear_classify(TEST_SET, linear_a, linear_b);
results(2,:,:)= confusionmat(TEST_LABELS, linear_predicted_labels');

[nonlinear_a, nonlinear_b] = nonlinearsvm(NON_SEIZURE_SET, SEIZURE_SET);
nonlinear_predicted_labels = nonlinear_classify(TEST_SET, nonlinear_a, nonlinear_b);
results(3,:,:)= confusionmat(TEST_LABELS, nonlinear_predicted_labels');

accuracies = zeros(1,3);
tp_rate = zeros(1,3);
fp_rate = zeros(1,3);

for i=1:3
        tp = results(i,2,2);
        tn = results(i,1,1);
        fp = results(i,2,1);
        fn = results(i,1,2);

        accuracies(i) = (tp+tn)/(tp+tn+fp+fn);
        tp_rate(i) = (tp)/(tp+fn);
        fp_rate(i) = (fp)/(fp+tn);
end

function [folds0, folds1] = make_cv(tr0, tr1, ts, ts_labels)
    numfolds = int32(5);

    for i=1:length(ts)
        if ts_labels(i) == 0
            tr0 = [tr0; ts(i,:)];
        else
            tr1 = [tr1; ts(i,:)];
        end
    end

    foldsize0 = idivide(length(tr0),numfolds,'ceil');
    foldsize1 = idivide(length(tr1),numfolds,'ceil');

    folds0 = zeros(foldsize0, length(tr0(1,:)), numfolds);
    folds1 = zeros(foldsize1, length(tr1(1,:)), numfolds);

    for i=1:numfolds
        folds0(:,:,i) = tr0((i-1)*foldsize0+1:i*foldsize0,:);
        folds1(:,:,i) = tr1((i-1)*foldsize1+1:i*foldsize0,:);
    end
end

function results = run_cv(folds0, folds1, trainingfun, testfun)
    numfolds = int32(5);
    results = zeros(numfolds, 2, 2);

    for i=1:numfolds
```

```
        training0 = [];
        training1 = [];
        testing0 = [];
        testing1 =[];
        for j=1:numfolds
            if i==j
                testing0 = [testing0; folds0(:,:,i)];
                testing1 = [testing1; folds1(:,:,i)];
            else
                training0 = [training0; folds0(:,:,i)];
                training1 = [training1; folds1(:,:,i)];
            end
        end

        [a,b] = trainingfun(training0, training1);
        predicted_labels = testfun([testing0; testing1], a, b);
        true_labels = [zeros(1,length(testing0)), ones(1,length(testing1))];

        results(i,:,:)= confusionmat(true_labels, predicted_labels);
    end


end

function [avg_0, avg_1] = simple_train(tr0, tr1)
    avg_0 = mean(tr0);
    avg_1 = mean(tr1);
end

function predicted_labels = simple_classify(data, avg_0, avg_1)
    predicted_labels=zeros(1, length(data));
    for i=1:size(data, 1)
        sample = data(i, :);
        if norm(sample - avg_0) > norm(sample - avg_1)
            % the test point is closer to the average point in class 1
            predicted_labels(i) = 1;
        end
    end
end

function [m, b]=linearsvm(tr0, tr1)
    X = tr1;
    Y = tr0;

    nf = size(X, 2);
    nx = size(X, 1);
    ny = size(Y, 1);
    n = nf + 1 + nx + ny;
    Q = zeros(n, n);
    Q(1:nf, 1:nf) = eye(nf);
```

```
    Tau = 1000;
    q = [zeros(nf + 1, 1); Tau * ones(nx + ny, 1)];
    A = [-X   ones(nx, 1)      -eye(nx)        zeros(nx, ny);
         Y    -ones(ny, 1)     zeros(ny, nx)   -eye(ny)];
    b = -ones(nx + ny, 1);
    L = [-inf * ones(nf + 1, 1); zeros(nx + ny, 1)];

    [z, ~] = quadprog(Q, q, A, b, [], [], L, [], []);

    m = z(1:nf);
    b = z(nf + 1);
end

function predicted_labels=linear_classify(data, m, b)
    predicted_labels = zeros(1, size(data, 1));
    for i = 1:size(data, 1)
        predicted_labels(i) = (sign(m' * data(i, :)' - b) + 1) / 2;
    end
end

function [m, b]=nonlinearsvm(tr0, tr1)
    X = tr1;
    Y = tr0;

    nf = size(X, 2);
    nx = size(X, 1);
    ny = size(Y, 1);
    n = nf + 1 + nx + ny;
    Q = zeros(n, n);
    Q(1:nf, 1:nf) = eye(nf);
    Tau = 1000;
    q = [zeros(nf + 1, 1); Tau * ones(nx + ny, 1)];
    A = [-X   ones(nx, 1)      -eye(nx)        zeros(nx, ny);
         Y    -ones(ny, 1)     zeros(ny, nx)   -eye(ny)];
    b = -ones(nx + ny, 1);
    lowerbound = [-inf * ones(nf + 1, 1); zeros(nx + ny, 1)];

    objFun = @(x) ((1/2)* x' * Q * x + q' * x)^2;

    [z, ~] = fmincon(objFun, zeros(n, 1), A, b, [], [], lowerbound, []);
    m = z(1:nf);
    b = z(nf + 1);
end

function predicted_labels=nonlinear_classify(data, m, b)
    predicted_labels = zeros(1, size(data, 1));
    for i = 1:size(data, 1)
        predicted_labels(i) = (sign((m' * data(i, :)')^2 - b) + 1) / 2;
    end
end
```