```
load('ion.mat');
rng('default')
y_aux = swith(y);
y = [y_aux, y];
[~, n] = size(X);
% [test_err_min, training_err_min, best_m] = cv(X, y, 5, 100)
[test_err_min, training_err_min, best_m] = cv(X, y, n, 60)
% [test_err_min, training_err_min, best_m] = loocv(X, y, 100)

function y_aux = swith(y)
    [n, ~] = size(y);
    y_aux = zeros(n, 1);
    for i = 1:n
        if y(i, :) == 0
            y_aux(i,:) = 1;
        end
    end
end

function y_hat = classify_y(y)
    [n, ~] = size(y);
    y_hat = zeros(n, 1);
    for i = 1:n
        if y(i, 1) < y(i, 2)
            y_hat(i,:) = 1;
        end
    end
end

function [y_hat, H] = loocv_y_hat(X, y, m)
    [~, n] = size(X);
    [idx,~,sumd,D] = kmeans(X', m);
    Phi = zeros(m, n);
    [GC, ~] = groupcounts(idx);
    avg = sumd ./ GC;
    avg(avg == 0) = 1e-10;
    for i = 1:n
        Phi(:,i) = exp(- D(i,:)' ./ avg);
%         Phi(:,i) = exp(- D(i,:)');
    end
    H = (Phi * Phi') \ Phi;
    H = Phi' * H;
```

```
    y_hat = H * y;
    y_hat = classify_y(y_hat);
end

function [test_err_min, training_err_min, best_m] = loocv(X, y, max_node)
    training_err = zeros(max_node, 1);
    test_err = zeros(max_node, 1);
    [~, n] = size(X);
    for m = 1:max_node
        test_err_m = 0;
        [y_hat, H] = loocv_y_hat(X, y, m);
%         H(H == 1) = 1 - 1e-10;
        for i = 1:n
            if H(i, i) == 1
                continue
            end
            test_err_m = test_err_m + ((y_hat(i, :) - y(i, 2)) / (1 - H(i, i))) .^2;
%             test_err_m = test_err_m + (sum(y_hat(i, :) - y(i, :)) / (1 - H(i, i))) .^2
        end
%         y_hat = classify_y(y_hat);
        training_err(m, :) = sum((y_hat - y(:, 2)).^2) / n;
        test_err(m, :) = test_err_m / n;
    end
    [test_err_min, best_m] = min(test_err);
    training_err_min = training_err(best_m,:);
    plot(test_err);
    hold on;
    plot(training_err);
    legend({'test error','training error'},'Location','northeast');
end

function [y_test_hat, y_train_hat] = get_y_hat(X_train, y_train, X_test, m)
    [~, n_train] = size(X_train);
    [idx,C,sumd,D] = kmeans(X_train', m);
    Phi = zeros(m, n_train);
    [GC, ~] = groupcounts(idx);
    avg = sumd ./ GC;
    avg(avg == 0) = 1e-10;
    for i = 1:n_train
        Phi(:,i) = exp(- D(i,:)' ./ avg);
%         Phi(:,i) = exp(- D(i,:)');
    end
```

```
    W = (Phi * Phi') \ Phi * y_train;
    y_train_hat = Phi' * W;
    y_train_hat = classify_y(y_train_hat);

    [~, n_test] = size(X_test);
    dists = zeros(m, n_test);
    for i = 1:n_test
        for j = 1:m
            dist = (X_test(:,i) - C(j,:)')' * (X_test(:,i) - C(j,:)');
            dists(j,i) = exp(- dist ./ avg(j,:));
%             dists(j,i) = exp(- dist);
        end
    end
    y_test_hat = dists' * W;
    y_test_hat = classify_y(y_test_hat);
end

function [test_err_min, training_err_min, best_m] = cv(X, y, k, max_node)
    divide = 0:k;
    if k == 5
        divide = [0, 70, 140, 210, 280, 351];
    end
    training_err = zeros(max_node, 1);
    test_err = zeros(max_node, 1);

    for m = 1:max_node
        training_err_m = 0;
        test_err_m = 0;
        for i = 1:k
            from = divide(:, i) + 1;
            to = divide(:, i + 1);
            train = [1:from - 1, (to + 1):351];
            test = from:to;
            [y_test_hat, y_train_hat] = get_y_hat(X(:, train), y(train,:), X(:, test), m

            [len_test, ~] = size(y_test_hat);
            [len_train, ~] = size(y_train_hat);
            test_err_m = test_err_m + sum((y_test_hat - y(test,2)).^2) / len_test;
            training_err_m = training_err_m + sum((y_train_hat - y(train,2)).^2) / len_t
        end
        training_err(m, :) = training_err_m / k;
        test_err(m, :) = test_err_m / k;
```

```
    end
    [test_err_min,best_m] = min(test_err);
    training_err_min = training_err(best_m,:);
    plot(test_err);
    hold on;
    plot(training_err);
    legend({'test error','training error'},'Location','northeast');
end
```

```
load('linear_new.mat');
get_ans(X, y, Xtest, ytest, 0, 1)
% load('noisylinear_new_1.mat');
% get_ans(X, y, Xtest, ytest, -3, 3)
% load('quadratic_new.mat');
% get_ans(X, y, Xtest, ytest, 0, 1)

function [b, b_0] = HardMarg(X, y)
    H = X .* y';
    H = H' * H;
    [~, n] = size(X);
    alphas = quadprog(H, -ones(n ,1), [], [], y', 0, zeros(n, 1), []);
    b = sum(alphas .* y .* X', 1)';
    idx = find(alphas > 0.0001);
    i = idx(1,:);
    b_0 = (1 - y(i,:) * b' * X(:, i)) / y(i,:);
end

function [b, b_0] = SoftMarg(X, y, gamma)
    H = X .* y';
    H = H' * H;
    [~, n] = size(X);
    b = zeros(n , 1);
    gamma_vec = zeros(n , 1) + gamma;
    alphas = quadprog(H, -ones(n ,1), [], [], y', 0, b, gamma_vec);
    b = sum(alphas .* y .* X', 1)';
    idx = find(alphas > 0.0001);
    i = idx(1,:);
    b_0 = (1 - y(i,:) * b' * X(:, i)) / y(i,:);
end

function [yhat] = classify(Xtest, b, b_0)
    y = Xtest' * b + b_0;
    yhat(y <= 0) = -1;
    yhat(y > 0) = 1;
end

function mme = cal_mme(y_test, y_hat)
    [n, ~] = size(y_test);
    count = 0;
    for i = 1:n
        if y_test(i,:) ~= y_hat(i, :)
```

```
            count = count + 1;
        end
    end
    mme = count / n;
end

function get_ans(X, y, Xtest, ytest, from, to)
    [bh, b_0h] = HardMarg(X, y);
    [bs, b_0s] = SoftMarg(X, y, 0.5);
    cls_1 = X(:, y == -1);
    cls_2 = X(:, y == 1);
    plot(cls_1(1,:), cls_1(2,:), '.')
    hold on
    plot(cls_2(1,:), cls_2(2,:), '.')
    hold on
    f = @(x) - bh(1,:) / bh(2,:) * x - b_0h / bh(2,:);
    line1 = fplot(f, [from , to]);
    hold on
    f = @(x) - bs(1,:) / bs(2,:) * x - b_0s / bs(2,:);
    line2 = fplot(f, [from , to]);
    legend([line1 line2],'hard margin SVM','soft margin SVM');
    yhat = classify(Xtest, bh, b_0h);
    mme_hard = cal_mme(ytest, yhat')
    yhat = classify(Xtest, bs, b_0s);
    mme_soft = cal_mme(ytest, yhat')
end
```