```
% clear

%% Load the data

load sim.mat
train = Data([1:500 1501:2000 3001:3500 4501:end],:);
test = Data([501:1000 2001:2500 3501:4000],:);
val = Data([1001:1500 2501:3000 4001:4500],:);

x_train = train(:,[1:34 36:end]);
y_train = train(:,35);
x_test = test(:,[1:34 36:end]);
y_test = test(:,35);
x_val = val(:,[1:34 36:end]);
y_val = val(:,35);

%% Scale our data sets
[xs_train, x_mean, x_std] = zscore1(x_train);
[ys_train, y_mean, y_std] = zscore1(y_train);

% Scale the validation and test data to the same scale
xs_test = zscore1(x_test, x_mean, x_std);
ys_test = zscore1(y_test, y_mean, y_std);

xs_val = zscore1(x_val, x_mean, x_std);
ys_val = zscore1(y_val, y_mean, y_std);

%% Bound the alpha values
[loadings latent perExp] = pcacov(cov(xs_train));
scores = xs_train * loadings;
cc = corrcoef([scores ys_train]);
s = svd(xs_train);

% to start, back-calculate from the corellation coefficients
ff_start = abs(cc(1:end-1,end));
alpha0 = sqrt(s.^2 .* (1 - ff_start) ./ ff_start);

alpha_min = min(svd(xs_train));
alpha_max = max(svd(xs_train));
ff = @(alpha) s.^2 ./ (s.^2 + alpha.^2);

%% Initialize the alpha values at 20

%alpha0 = rand(size(xs_train,2), 1) * (alpha_max - alpha_min) + alpha_min
%alpha0 = 20 * ones(size(xs_train, 2), 1);

%% Fitness function and optimization settings
func = @(x) ridge_mse(x, xs_train, ys_train, xs_test, ys_test);
opt = optimset('plotfcn', {@optimplotx, @optimplotfval}, 'display', 'iter');

%% Gradient descent
[alpha_gd, fval, exitflag, output] = fminunc(func, alpha0, opt);
B_gd = (xs_train'*xs_train + diag(alpha_gd.^2))\(xs_train'*ys_train);
ys_v = xs_val * B_gd;
y_gd = ys_v * y_std + y_mean;

%% Genetic algorithm
% Default options
ga_opt1 = optimset('plotfcn', {@optimplotx, @optimplotfval}, 'display', 'iter');
```

```
alpha_ga1 = ga(func, 43)';
B_ga1 = (xs_train'*xs_train + diag(alpha_ga1.^2))\(xs_train'*ys_train);
ys_v = xs_val * B_ga1;
y_ga1 = ys_v * y_std + y_mean;

% Larger population
ga_opt2 = optimset('plotfcn', {@optimplotx, @optimplotfval}, 'display', 'iter');

%% Graphs and stuff!
f = figure;
plot(y_val, '.');
hold on;
plot(y_gd, 'x');
plot(y_ga1, '^');
legend('Validation Output', 'Gradient Descent', 'Genetic Algorithm');

g = figure;
plot(ff(alpha_gd));
hold on;
plot(ff(alpha_ga1));
ylabel('\alpha value');
legend('Gradient descent', 'Genetic algorithm');
title('\alpha values for optimization algorithms');

%% Calculate MSE for each
gd_err = sqrt(mean((y_gd - y_val).^2))
ga_err = sqrt(mean((y_ga - y_val).^2))
```