```
Part 3:
t = 10;
te = 1000;
repeats = 100;

%define parameters: u, sigma, lambda, featurefun
n = 2;
u = [0; ones(n-1,1)]; % target weights
sigma = 0.1; % noise level
lambda = 0.005; % regularization parameter
featurefun = @(X)quadfeatures(X);
%generate test data: Xtest and ytest
Xtest = [ones(te,1) rand(te, n-1)]; % training patterns

ytest = (Xtest*u).^2 + randn(te,1)*sigma; % target values
for r = 1:repeats

  %generate training data: X and y
        X = [ones(t,1) rand(t, n-1)]; % training patterns
        y = (X*u).^2 + randn(t,1)*sigma; % target values


  w1 = fitlin(y,X)
  w2 = fitlinreg(y,X,lambda)
  w3 = fitgenreg(y,X,lambda,featurefun)

  yhat_train1 = X*w1
  yhat_train2 = X*w2
  yhat_train3 = predictgen(w3,X,featurefun)

  err_train1(r) = (y - yhat_train1)'*(y - yhat_train1)/t
  err_train2(r) = (y - yhat_train2)'*(y - yhat_train2)/t
  err_train3(r) = (y - yhat_train3)'*(y - yhat_train3)/t

  yhat_test1 = Xtest*w1
  yhat_test2 = Xtest*w2
  yhat_test3 = predictgen(w3,Xtest,featurefun)

  err_test1(r) = (ytest - yhat_test1)'*(ytest - yhat_test1)/te;
  err_test2(r) = (ytest - yhat_test2)'*(ytest - yhat_test2)/te;
  err_test3(r) = (ytest - yhat_test3)'*(ytest - yhat_test3)/te;

end
```

```
average_train_error1 = mean(err_train1)
average_train_error2 = mean(err_train2)
average_train_error3 = mean(err_train3)
std_train_error1 = std(err_train1)/sqrt(repeats)
std_train_error2 = std(err_train2)/sqrt(repeats)
std_train_error3 = std(err_train3)/sqrt(repeats)

average_test_error1 = mean(err_test1)
average_test_error2 = mean(err_test2)
average_test_error3 = mean(err_test3)
std_test_error1 = std(err_test1)/sqrt(repeats)
std_test_error2 = std(err_test2)/sqrt(repeats)
std_test_error3 = std(err_test3)/sqrt(repeats)

average_train_error1 =0.0119
average_train_error2 =0.0119
average_train_error3 =0.0072
std_train_error1 =  6.0272e-04
std_train_error2 =  6.0266e-04
std_train_error3 = 3.8441e-04
average_test_error1 =0.0199
average_test_error2 =0.0197
average_test_error3 =0.0147
std_test_error1 = 7.2153e-04
std_test_error2 =  6.9403e-04
std_test_error3 = 7.1646e-04
```

Part 6:

```
%load data1.mat;
featurefun = @(X)quadfeatures(X);
kernelfun1 = @(X1,X2)quadkernel(X1,X2);
kernelfun2 = @(X1,X2)gausskernel(X1,X2,50);

ww1 = fitlin(y,X);
ww2 = fitlinreg(y,X,25);
ww3 = fitgenreg(y,X,1e5,featurefun);
aa1 = fitdualgenreg(y,X,1e5,kernelfun1);
aa2 = fitdualgenreg(y,X,5e-3,kernelfun2);

   yhat_train1 = X*ww1;
```

```
  yhat_train2 = X*ww2;
  yhat_train3 = predictgen(ww3,X,featurefun);
  yhat_train4 = predictdualgen(aa1,X,X,kernelfun1);
  yhat_train5 = predictdualgen(aa2,X,X,kernelfun2);

  err_train1 = (y - yhat_train1)'*(y - yhat_train1)/67
  err_train2 = (y - yhat_train2)'*(y - yhat_train2)/67
  err_train3 = (y - yhat_train3)'*(y - yhat_train3)/67
  err_train4 = (y - yhat_train4)'*(y - yhat_train4)/67
  err_train5 = (y - yhat_train5)'*(y - yhat_train5)/67

  yhat_test1 = Xtest*ww1;
  yhat_test2 = Xtest*ww2;
  yhat_test3 = predictgen(ww3,Xtest,featurefun);
  yhat_test4 = predictdualgen(aa1,Xtest,X,kernelfun1);
  yhat_test5 = predictdualgen(aa2,Xtest,X,kernelfun2);

  err_test1 = (ytest - yhat_test1)'*(ytest - yhat_test1)/30
  err_test2 = (ytest - yhat_test2)'*(ytest - yhat_test2)/30
  err_test3 = (ytest - yhat_test3)'*(ytest - yhat_test3)/30
  err_test4 = (ytest - yhat_test4)'*(ytest - yhat_test4)/30
  err_test5 = (ytest - yhat_test5)'*(ytest - yhat_test5)/30
```

```
err_train1 =0.6832
err_train2 =0.7304
err_train3 =0.6947
err_train4 =0.6947
err_train5 =0.6033
err_test1 =0.8104
err_test2 =0.7344
err_test3 =0.7411
err_test4 =0.7411
err_test5 = 0.7981
```

--5th function best for training data
--2nd function best for testing data
--The overall best function is the 5th one (fitdualgenreg(y,X,5e-3,kernelfun2);) because the overall error between the average of the training and testing data is the smallest among the other functions and is also more accurate with a larger set of data than the others.