week6_code

Created: 18/11/22 13:16 **Updated:** 18/11/22 13:22

partial codes in week_6 code assignment.

> written by *VincentX3*, Nov.22.18

linearRegCostFunction.m

```
mask = ones(size(theta));
mask(1) = 0;
h=X*theta;

J=(h-y)'*(h-y)/(2*m)+(lambda/(2*m))*(theta' * theta - theta(1)^2);
grad=X'*(h-y)/m+lambda/m*(theta.*mask);
```

polyFeatures.m

```
for i=1:p
    X_poly(:,i)=X.^i;
end
```

validationCurve.m

```
for i = 1:length(lambda_vec)
    lambda=lambda_vec(i);
    theta=trainLinearReg(X,y,lambda);
    %error should be computed with non-regularized version J(theta)
    error_train(i)=linearRegCostFunction(X,y,theta,0);
    error_val(i)=linearRegCostFunction(Xval,yval,theta,0);
end
```

learningCurve.m

```
for i=1:m
    theta=trainLinearReg(X(1:i,:),y(1:i),lambda);
    error_train(i)=linearRegCostFunction(X(1:i,:),y(1:i),theta,0);
    error_val(i)=linearRegCostFunction(Xval,yval,theta,0);
end
```

---

self code

prediction_on_test.m

```
function
[prediction]=prediction_on_test(X,y,Xtest,ytest,lambda,p,mu, sigma)

%given the p (indicated the poly) and best lambda
%output the prediction on test set



% Map X onto Polynomial Features and Normalize
X_poly = polyFeatures(X, p);
[X_poly, mu, sigma] = featureNormalize(X_poly);
% Normalize
X_poly = [ones(length(X), 1), X_poly];
% Add Ones

% Map X_poly_test and normalize (using mu and sigma)
X_poly_test = polyFeatures(Xtest, p);
X_poly_test = bsxfun(@minus, X_poly_test, mu);
X_poly_test = bsxfun(@rdivide, X_poly_test, sigma);
X_poly_test = [ones(size(X_poly_test, 1), 1),
X_poly_test];         % Add Ones

theta=trainLinearReg(X_poly,y,lambda);
```

```
prediction=linearRegCostFunction(X_poly_test,ytest,theta,lambda);
```