ML 2016/17
Exercise 2: Regression

25/10/16

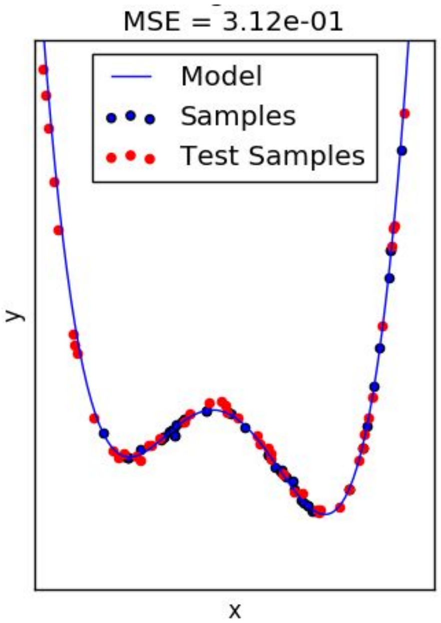
General information

- The assignments are not graded on a scale: it's simply pass/no pass
  - If one homework is not sufficient you can simply redo it
- All assignments must be delivered one month before **you** take the exam
- Submission through email: send to fabiom.carlucci@dis.uniroma1.it with appropriate subject
- Questions can be written to same email address.
- Office hours to meet in person: **Wednesday** at B004 (Via Ariosto, the door in front of library), 10AM-12PM.
- Use any language, Python recommended: https://www.continuum.io/downloads

HW2: Regression

- Familiarize with the basic concepts of linear regression
- How to deal with the non-linear case
- Concept experiment on underfitting, overfitting and model selection

Once you complete the experience send the report to fabiom.carlucci@dis.uniroma1.it with subject “[ML1617] Regression report”



HW2: general overview

1. Load X train, y train, X test and y test and plot them
2. Fit a linear model to X train and y train
3. Test your model on the X, y test and plot both the points and the model

This will not work well, as the given data points cannot be approximated linearly

4. For **n**=1:10
    a. Map the X train data points to a polynomial of degree **n**
    b. Fit a linear model to the mapped points and test it on the (X, y) test
5. Plot the mean squared error for each trained model
6. Choose the model with the lowest error and plot the chosen model

Load the data

Download data from the dropbox folder: *regression.tar* (remember to extract it)

Load X train, y train, X test and y test into memory - ex:

```
numpy.load("regression_Xtrain.npy")
```

For now, let's focus simply on **x train** and **y train**. In general we always learn (or *fit*) our models on the training set and then see how well it performs on the test set. *

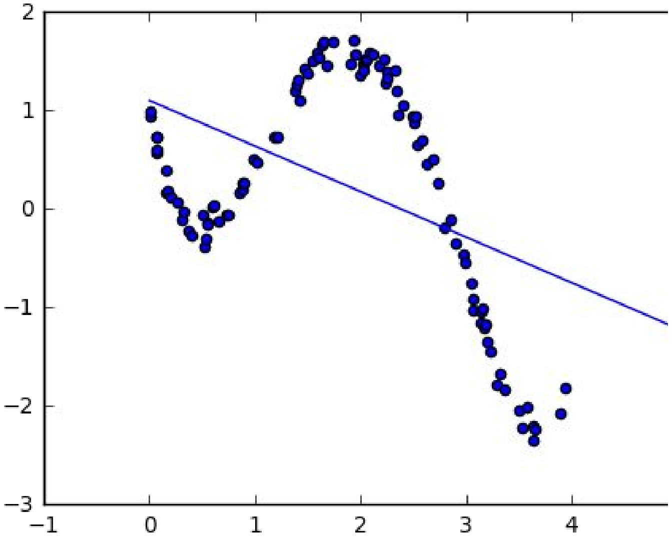
* *in real world problems you always tune the model on a validation set first*

Fit a linear model

Some code to start: (see http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)

```
lr = LinearRegression()
lr.fit( x_train.reshape(-1,1), y)  //LR.fit() wants an array
plt.plot(X_test, lr.predict(X_test.reshape(-1,1), label="Model")
plt.scatter( …. )

predicted = lr.predict(X_test.reshape(-1,1))
mean_square_error = … //you will need x_test and y_test
```

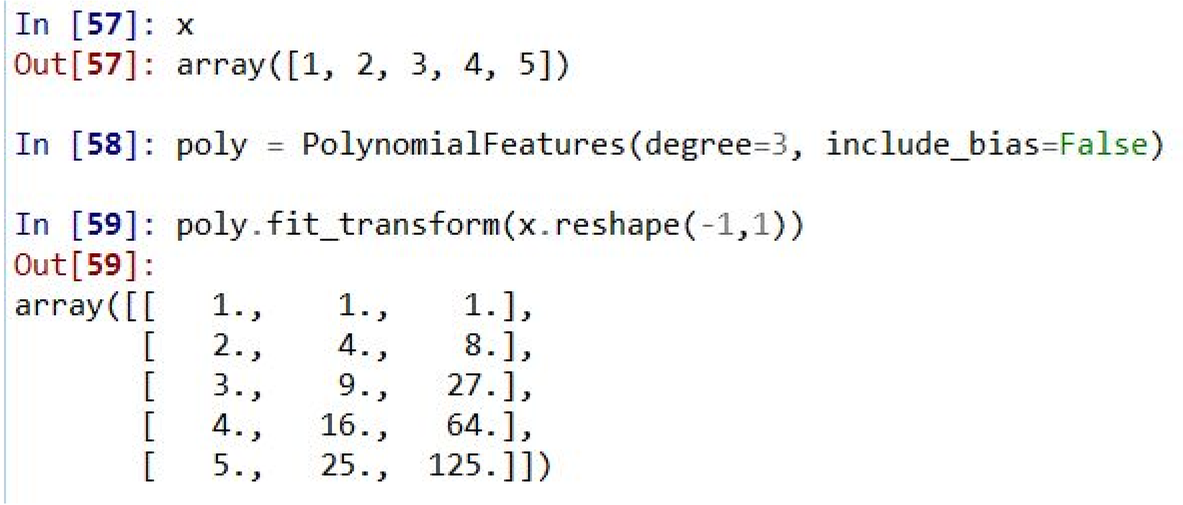


How to model polynomial data?

Linear regression allows you to learn a linear combination of its inputs

But if you map your inputs into a polynomial space you can learn polynomial models! (see http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html)

Example:



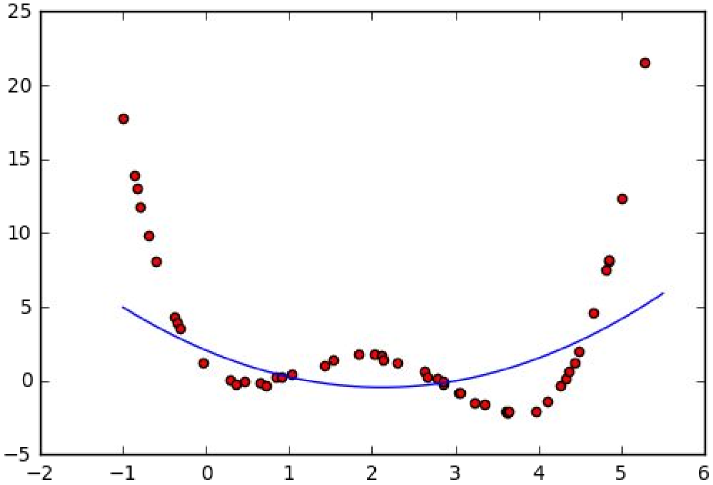
```
In [57]: x
Out[57]: array([1, 2, 3, 4, 5])

In [58]: poly = PolynomialFeatures(degree=3, include_bias=False)

In [59]: poly.fit_transform(x.reshape(-1,1))
Out[59]:
array([[   1.,    1.,    1.],
       [   2.,    4.,    8.],
       [   3.,    9.,   27.],
       [   4.,   16.,   64.],
       [   5.,   25.,  125.]])
```

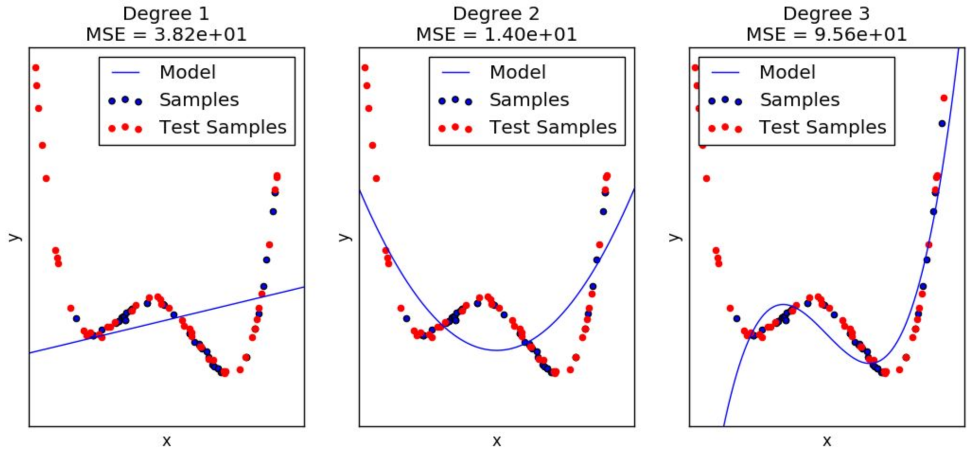
Fitting to a polynomial

```
poly = PolynomialFeatures(degree=2, include_bias=False)
xPoly = poly.fit_transform(Xtrain)
lr = LinearRegression()
lr.fit(xPoly, ytrain)
x_range = np.linspace(-1, 5.5, 100)
predicted = lr.predict(poly.fit_transform(
             x_range.reshape(-1,1)))
plt.plot(x_range.reshape(-1,1), predicted)
plt.scatter(Xtest.reshape(-1,1),
             ytest, c='r')
plt.show()
```



Example output

So, iterate between 1 and 10 and test all polynomial degrees in this range. Plot the MSE while varying the degree and then choose the best fit.
Why do high degree polynomials have a high error?

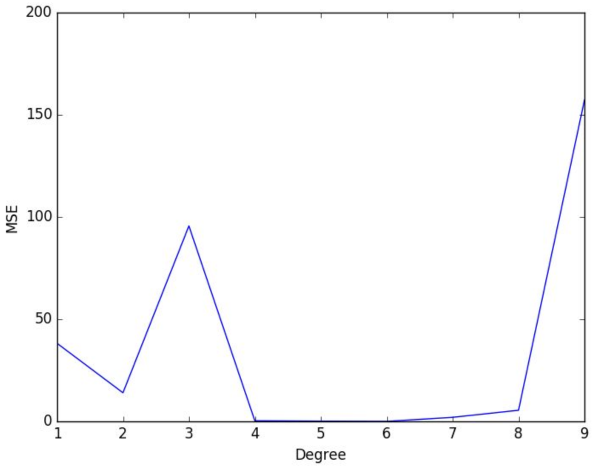


Degree 1
MSE = 3.82e+01

Degree 2
MSE = 1.40e+01

Degree 3
MSE = 9.56e+01

Plotting the regression of various degrees



Plotting the mean square error of various degrees