Mary Karroqe: mk5721

Model Selection In-Class Exercise

In this example, you will a linear model to data and select the model order by model order selection. First load the standard packages.

In [10]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pickle
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import numpy.polynomial.polynomial as poly
```

Load the Data

The data in this exercise is completely synthetic. You can load it with the following commmand.

In [4]:

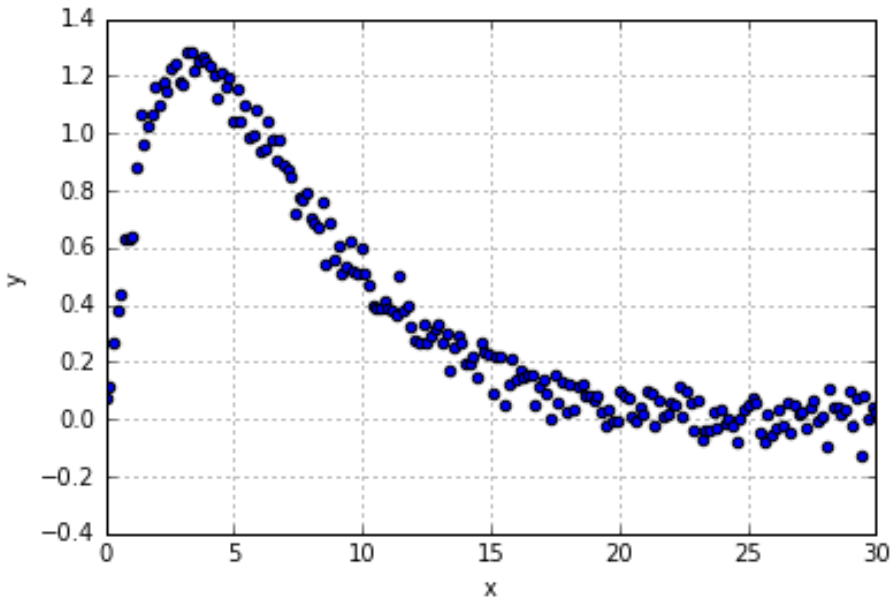
```
fn_src = 'https://raw.githubusercontent.com/sdrangan/introml/master/unit04_model_sel/synth_data.p'
fn_dst = 'synth_data.p'

import os
from six.moves import urllib

if os.path.isfile(fn_dst):
    print('File %s is already downloaded' % fn_dst)
else:
    urllib.request.urlretrieve(fn_src, fn_dst)
    print('File %s downloaded' % fn_dst)

with open(fn_dst,'rb') as fp:
    x,y = pickle.load(fp)
```

```
File synth_data.p is already downloaded
```

Plot the data `y` vs. `x` using a scatter plot.

In [7]:

```
plt.scatter(x,y)
plt.xlim([0,30])
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()
```



Fit the Data

We will now try to fit the data. First, split the data into training and test. You can use the `train_test_split()` method in the `sklearn` package.

In [8]:

```
xtr, xts, ytr, yts = train_test_split(x, y, test_size=0.5)
```

Now try to fit a linear model to the data. You can pick any linear model with a variable number `d` of basis functions. For example, you can use the polynomials up to degree `d-1`.

For each model order `d`:

- Fit the model on the training data
- Test the model on the test data

Plot the test error vs. `d`. Select the model order `dopt` with the lowest test error.

In [106]:

```
dtest = np.array(range(1,11))
RSStr = []
RSSts = []

for d in dtest:
    beta_hat = poly.polyfit(xtr, ytr, d)

    # rss: training data
    y_hat = poly.polyval(xtr, beta_hat)
    RSSd = np.mean((y_hat - ytr)**2)
    RSStr.append(RSSd)

    # rss test data
    y_hat = poly.polyval(xts, beta_hat)
    RSSd = np.mean((y_hat - yts)**2)
    RSSts.append(RSSd)

plt.plot(dtest, RSStr, 'o-', linewidth=2, c='b')
plt.plot(dtest, RSSts, 's-', linewidth=2, c='r')
plt.xlabel("Model Order")
plt.ylabel("RSS")
plt.title("Training v. Test Data")
plt.legend(['Training',  'Test'], loc='upper right')
plt.grid()
```

In [107]:

```
dopt = np.argmin(RSSts)
print("Estimated model order= {0:d}".format(dtest[dopt]))
```

```
Estimated model order= 7
```

Select the optimal model order `dopt`. Re-train the model for that model order. On one plot:

- Plot the predicted value `yhat` vs. `x` for your model for `x in [0,35]`
- Plot a scatter plot of the test data `xts` vs. `yts`

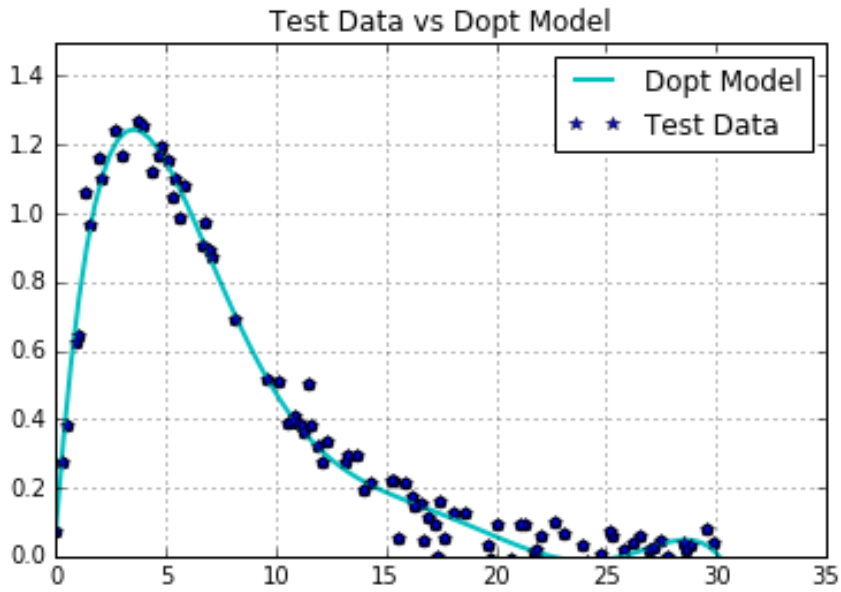
Does your model fit the test data well? Does it extrapolate reasonably in the range `x >= 30`?

In [108]:

```
beta_hat = poly.polyfit(xtr, ytr, dopt)

# Plot true function
xp = np.linspace(0,35,100)
yp_hat = poly.polyval(xp, beta_hat)
plt.xlim(0, 35)
plt.ylim(0, 1.5)
plt.plot(xp, yp_hat,'c-',linewidth=2)

# Plot data
plt.scatter(xts,yts)
plt.plot(xts,yts, '*')
plt.grid()
plt.title("Test Data vs Dopt Model")
plt.legend(['Dopt Model','Test Data'],loc='upper right')
```

Out[108]:

```
<matplotlib.legend.Legend at 0x129deefd0>
```



The model of degree 7 seems to fit the data fairly well up to 30. Beyond that.. since the curve goes downward, I expect it may not perform as well.

In [ ]: