Model Selection

Logout

Model Selection

In this chapter we will discuss on how to select between alternative models. This question often boils down to the question on how many parameters can be supported by a model. A typical example question would be: How many peaks can we identify in a spectrum?

Before we go into the details, we need to formulate our problem in a Bayesian way: Let us assume that we want to compare two models or hypothesis H_1 and H_2. The first has no parameters and the second one has one parameter λ. This argument can be generalized to comparing any two theories with different number of parameters. To tell which hypothesis is the better one we calculate the posterior ratio:

$$PR = \frac{P(H_1 \mid D)}{P(H_2 \mid D)}$$

if PR is greater than one we would prefer H_1 over H_2, but if PR is on the order unity then the data is not sufficient to make a decision. We will now use Bayes theorem to get a better sense on how to calculate this ratio.

$$\frac{P(H_1 \mid D)}{P(H_2 \mid D)} = \frac{P(D \mid H_1)}{P(D \mid H_2)} \frac{P(H_1)}{P(H_2)}$$

the nice thing about this ratio is that $P(D)$ cancels because it is the same for both hypothesis. The last ratio is the ratio of priors and we should probably assume it to be of the order unity. This is unless there is reason to believe that one hypothesis is more likely to be true then the other one. This leaves the first ratio. $P(D \mid H_1)$ is easy but $P(D \mid H_2)$ involves a parameter which we can get to by expressing it as

$$P(D \mid H_2) = \int P(D, \lambda \mid H_2) d\lambda = \int P(D \mid \lambda, H_2) P(\lambda \mid H_2) d\lambda$$

$P(D \mid \lambda, H_2)$ is just the likelihood function for hypothesis H_2 with parameter λ. $P(\lambda \mid H_2)$ is the prior an reflects the knowledge of the range of λ that is known before the experiment.

Lets now assume that λ must lie between two values λ_{min} and λ_{max} which will assign a uniform prior:

$$P(\lambda \mid H_2) = \frac{1}{\lambda_{max} - \lambda_{min}} \quad \text{for} \quad \lambda_{min} \leq \lambda \leq \lambda_{max}$$

after the measurement we will find that a particular value of $\lambda_0 \pm \delta\lambda$ yields the closest agreement with the data. The corresponding probability $P(D \mid \lambda_0, H_2)$ will be the maximum of H_2's likelihood function. Let us also assume that we can represent the full likelihood as a Gaussian:

$$P(D \mid \lambda, H_2) = P(D \mid \lambda_0, H_2) \exp\left[-\frac{(\lambda - \lambda_0)^2}{2\delta\lambda^2} \right]$$

since the uniform prior does not explicitly depend on λ we can calculate

$$P(D \mid H_2) = \frac{1}{\lambda_{max} - \lambda_{min}} \int_{\lambda_{min}}^{\lambda_{max}} P(D \mid \lambda, H_2)$$

Assuming that the vast majority of the Gaussian is contained in the prior interval this reduces to:

$$P(D \mid H_2) = \frac{P(D \mid \lambda_0, H_2)\delta\lambda\sqrt{2\pi}}{\lambda_{max} - \lambda_{min}}$$

Finally the posterior ratio can be written using three terms:

$$\frac{P(H_1 \mid D)}{P(H_2 \mid D)} = \frac{P(H_1)}{P(H_2)} \frac{P(D \mid H_1)}{P(D \mid \lambda_0, H_2)} \frac{\lambda_{max} - \lambda_{min}}{\delta\lambda\sqrt{2\pi}}$$

the first term is our bias with respect to the two theories (should be unity), the second tells us about how well the two theories fit the data (this should be in favor of H_2 since it has a free parameter). The third term can be seen as penalizing the introduction of the parameter. Since $\lambda_{max} - \lambda_{min}$ should, in general, be larger than $\delta\lambda\sqrt{2\pi}$ this factor is larger than one. This factor is often called a Ockham factor after the thirteen-century Franciscan monk William of Ockham (or Occam, in Latin) who said "it is vain to do with more what can be done with fewer". But instead of blindly applying Ockams razor, we now have a quantitative tool to determine how many paramaters are supported by the data.

In [433]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [434]:
```
# to see how sigma affects PR change it between 0.5 to 3
sigma = 2
x = np.linspace(-5,5,11)
yqua = 2*x + 0.05*x**2 + np.random.normal(loc=0,scale=sigma,size=11)
```

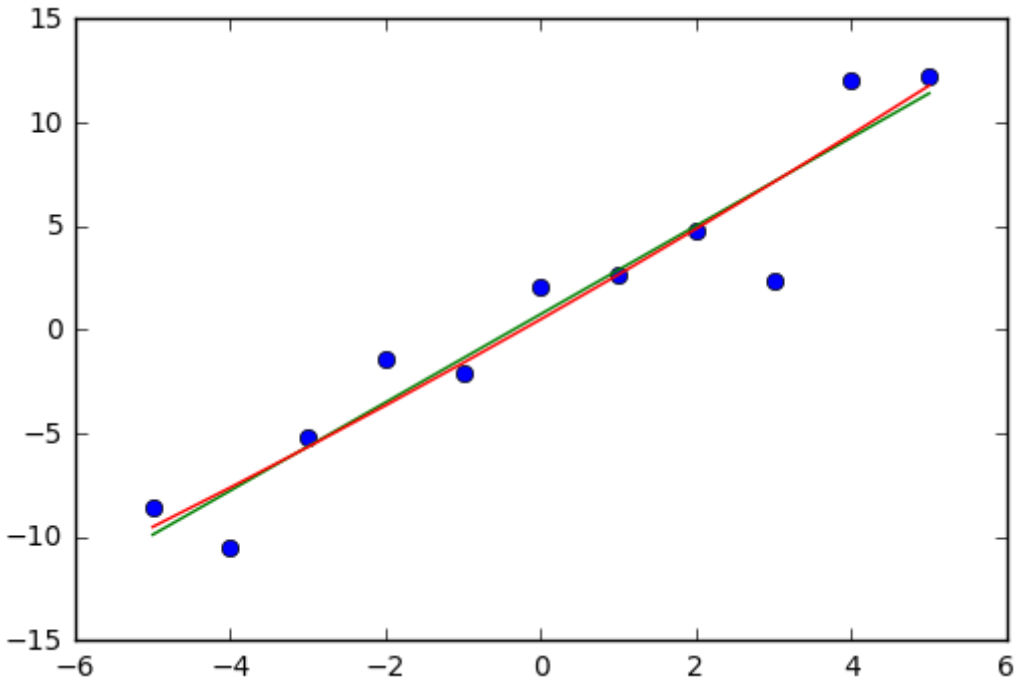
In [435]:

```
p = np.polyfit(x,yqua,1)
print(p)
ylin_fit = np.poly1d(p)(x)
res_lin = ylin_fit - yqua
chi2_lin = np.sum(res_lin**2)/sigma**2
print("Chi^2: ",chi2_lin,"exp(-Chi^2): ",np.exp(-chi2_lin))

p2 = np.polyfit(x,yqua,2,cov=True)
print(p2)
p2_cov = p2[1]
yqua_fit = np.poly1d(p2[0])(x)
res_qua = yqua_fit - yqua
chi2_qua = np.sum(res_qua**2)/sigma**2
print("Chi^2: ",chi2_qua, "P(p0): ",np.exp(-chi2_qua),"delta p0: ",np.sq

print("PR = ",np.exp(-(chi2_lin-chi2_qua))*0.6/2/np.pi/np.sqrt(p2_cov[0]
```

```
[ 2.1297098  0.7631902]
Chi^2:  11.6054050501 exp(-Chi^2):  9.11667822361e-06
(array([ 0.02500416,  2.1297098 ,  0.51314857]), array([[  8.91320738e-03,  -1.169526
45e-18,  -8.91320738e-02],
       [ -1.16952645e-18,   6.95230176e-02,   1.45035233e-17],
       [ -8.91320738e-02,   1.45035233e-17,   1.58655091e+00]]))
Chi^2:  11.4712979024 P(p0):  1.0425061381e-05 delta p0:  0.0944097843626
PR =  0.884529623142
```

In [436]:
```
1 plt.plot(x,yqua,"o")
2 plt.plot(x,ylin_fit)
3 plt.plot(x,yqua_fit)
```

Out[436]:
```
[<matplotlib.lines.Line2D at 0x117a11128>]
```



In [ ]:
```
1
```