Building Models

Bernoulli trials

Carl Henrik Ek

October 11, 2017

Abstract

In this document we will look at how we can learn from data. We have put this together to help you with the assignment. The key thing is for you to be able to abstract what you have here to what you have in the assignment. Here we are working with binary data, in the assignment with continous, **but** importantly the ideas are still the same, the computations are the same.

1 Task

We have been given the task to observe a system which has a binary outcome, you can think of the example of modelling a coin toss. So now we put on our machine learning hat. We will refer to a single outcome of the system as x and if we run the system for N iterations we will call all the data $\mathbf{x}$.

1.1 Likelihood

The first thing we need to think of is what is the likelihood function. For a binary system it makes sense to use a `Bernoulli Distribution` as likelihood function,

$$\text{Bern}(x|\mu) = \mu^x(1-\mu)^{1-x}.$$

This distribution takes a single parameter μ which completely parametrises our likelihood. This means, we have a conditional distribution and to understand the system we need to find this parameter μ. If we know μ we can generate output that is similar to what the actual system does, this means we can predict how the system behaves. Now we want to use examples of the systems predictions `training data` find μ. So far we have only set the likelihood for one data-point, what about when we see lots of them. Now we will make our first assumption, we will assume that each output of the system is independent. This means that we can factorise the distribution over several trials in a simple manner,

$$p(\mathbf{x}|\mu) = \prod_{i=1}^{N} \text{Bern}(x|\mu) = \prod_{i=1}^{N} \mu^x(1-\mu)^{1-x}.$$

Now we have the likelihood for the whole data-set $\mathbf{x}$.

1.2 Prior

In order to say something about the system we need to have a prior belief about what we think the parameter μ is. Now our prior knowledge comes into play, what do I know about the system? If our system is the outcome of a coin toss then we have a lot of prior knowledge, most coins that I toss are unbiased so I have quite a good idea of what I think it should be. Another way of seeing this is that I would need to see a lot of coin tosses saying something else for me to believe that a coin is not biased. If it is not a coin toss but something that I have no experience in my prior might be different. Once we have specified the prior we can just do Baye's rule and get the posterior,

$$p(\mu|\mathbf{x}) = \frac{p(\mathbf{x}|\mu)p(\mu)}{p(\mathbf{x})}.$$

Now comes the first tricky part, what should the distribution be for the prior? If you choose your prior or likelihood wrong (you have to believe me on this right now but we will see it later) then this computation might not even be analytically possible to perform. So this is when we will use *conjugacy*. First lets note that the posterior distribution is proportional to the likelihood times the prior (or the joint distribution),

$$p(\mu|\mathbf{x}) \propto p(\mathbf{x}|\mu)p(\mu).$$

The second thing we will think of is that it does make sense that if I have a prior of a specific form (say a Gaussian) then I would like the posterior to be of the same form (i.e. Gaussian). So this means that we should *choose* a prior such that when multiplied with the likelihood it leads to a distribution that is of the same form of the prior. So why is this useful? This is very useful because it means we do **not** have to compute the denominator in Baye's rule, the only thing we need to do is to multiply the prior with the likelihood and then **identify** the parameters that of the distribution, we can do this as we know its form.

So how do we know which distributions are conjugate to what? Well this is something that we leave to the mathematicians most of the time, we simply exploit their results. There is a list on Wikipedia of conjugate priors here URL. Its like a match making list for distributions. For the Bernoulli distribution the conjugate prior to its only parameter μ is the `Beta` distribution,

$$\text{Beta}(\mu|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1},$$

where $\Gamma(\cdot)$ is the gamma function. The role of the Gamma function is to normalise this to make sure it becomes a distribution not just a normal function. Now we have choosen our prior we are ready to derive the posterior.

1.3 Posterior

To get to the posterior we are going to multiply the likelihood and the prior together,

$$
\begin{aligned}
p(\mu|\mathbf{x}) &\propto p(\mathbf{x}|\mu)p(\mu) && (1)\\
&= \prod_{i=1}^{N} \text{Bern}(x|\mu)\text{Beta}(\mu|a,b) && (2)\\
&= \prod_{i=1}^{N} \mu^{x}(1-\mu)^{1-x}\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1} && (3)\\
&= \mu^{\sum_i x_i}(1-\mu)^{\sum_i (1-x_i)}\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1} && (4)\\
&= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{\sum_i x_i}(1-\mu)^{\sum_i (1-x_i)}\mu^{a-1}(1-\mu)^{b-1} && (5)\\
&= \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{\sum_i x_i + a}(1-\mu)^{\sum_i (1-x_i)+b-1}. && (6)
\end{aligned}
$$

Now comes the trick with conjugacy, *we know the form of the posterior*. This means we can just identify the parameters of the posterior and in this case it is trivial,

$$
\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{\underbrace{\sum_i x_i + a}_{a_n}}(1-\mu)^{\underbrace{\sum_i (1-x_i) + b}_{b_n}-1}
$$

This mean that my posterior is,

$$
\text{Beta}(\mu|a_n,b_n) = \frac{\Gamma\left(\sum_i x_i + a + \sum_i (1-x_i) + b\right)}{\Gamma\left(\sum_i x_i + a\right)\Gamma\left(\sum_i (1-x_i) + b\right)}\mu^{\sum_i x_i + a}(1-\mu)^{\sum_i (1-x_i)+b-1}
$$

So thats it, we have the posterior and now we can fix our parameters for the prior a and b and then compute the posterior and get a_n and b_n after seeing n data-points. So lets write code that simulates one of these experiments.

2 Code

Common practice if you want to test something is to generate data from a distribution with a parameter that you know and then see if you can recover the parameter. In `numpy` there isn't a way of sampling from a Bernoulli distribution, but as this is the same as sampling from a `Binomial` we can do this instead. So we start of with setting μ to 0.2 and then generate 200 values from this distribution, i.e. running the system 200 iterations or tossing a coin 200 times. Then we define our prior by setting the parameters a and b. Now we can compute our posterior, we know its form, we both derived it above, but in most cases you just write it down, that is what you will do for linear regression. Now

we can plot the posterior when we see more and more examples and see what will happen. Below is the image that it generated for me,

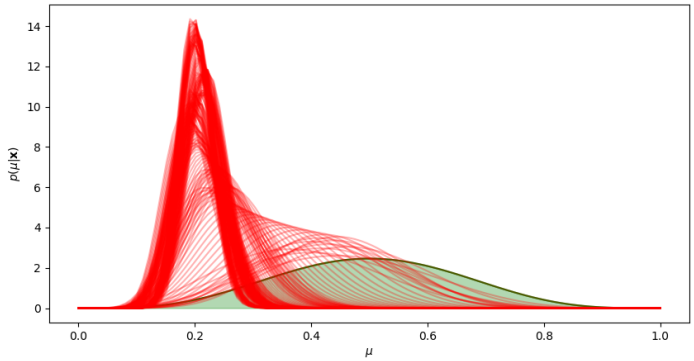

Figure 1: *The green distribution is the prior distribution over μ and the red distributions are the updated belief, our posterior when I see more and more data-points.*

```
import numpy as np
from scipy.stats import beta
import matplotlib.pyplot as plt

def posterior(a,b,X):
    a_n = a + X.sum()
    b_n = b + (X.shape[0]-X.sum())

    return beta.pdf(mu_test,a_n,b_n)

# parameters to generate data
mu = 0.2
N = 200

# generate some data
X = np.random.binomial(1,mu,N)
mu_test = np.linspace(0,1,100)

# now lets define our prior
a = 5.0
b = 5.0

# p(mu) = Beta(alpha,beta)
prior_mu = beta.pdf(mu_test,a,b)
```

```
# we have derived the posterior
a_n = a + X.sum()
b_n = b + (N-X.sum())
posterior_mu = beta.pdf(mu_test,a_n,b_n)

# create figure
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(111)

# plot prior
ax.plot(mu_test,prior_mu,'g')
ax.fill_between(mu_test,prior_mu,color='green',alpha=0.3)

ax.set_xlabel('$\mu$')
ax.set_ylabel('$p(\mu|\mathbf{x})$')

# lets pick a random (uniform) point from the data
# and update our assumption with this
index = np.random.permutation(X.shape[0])
for i in range(0,X.shape[0]):
    y = posterior(a,b,X[:index[i]])
    plt.plot(mu_test,y,'r',alpha=0.3)

plt.show()
```

3 Summary

So how does this help me, we are not doing Bernoulli data? Well the idea is that it is all the same, you need to do exactly the same proceedure, specify likelihood, choose the prior, get the posterior (or rather pick it from the lecture slides/book/wikipedia) and then generate something similar. The only difference is that you will get the posterior and then *sample* from this to get lots of $\mathbf{w}$ and then plot these lines. Here we plotted just the distribution but generating binary data isn't particularly interesting. Now you can play around with this data and see what is happening, test to generate data and then set the prior to be very similar to the value that used to generate, i.e. if you generate the data with μ_0 make sure that $p(\mu_0)$ is large, what happens now? Try the reverse, set a prior really far from μ_0 what happens now?