COMS30007 - Machine Learning

Distributions

Carl Henrik Ek

Week 2

Abstract

In this part we will look at how we can learn from data. The aim with this lab is to understand the concepts of *conjugacy* and the interplay between *likelihood*, *prior* and *posterior*. In this task the data is binary and next week we will work on continuous data. Importantly the ideas are the same, the computations are the same, your challenge is therefore to abstract this document beyond the details and understand the underlying concepts.

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. We will parametrise the system using a single parameter that describes how often we get each outcome, with the coin analogy how biased the coin is. We will refer to a single outcome of the system as x and if we run the system for N iterations we will refer to all the data $\mathbf{x}$. We will first create a function to generate the data by sampling from a distribution with known parameters, the machine learning task is then to *recover* the parameters of the distribution from only the data-points.

In order to phrase this as a learning problem we need to formulate the probabilistic objects that constitutes the model of how the data have been generated. We will formulate the *likelihood* function that will express our belief in specific types of data if we know the parametrisation of the system. Say that if I think that I have an unbiased coin then seeing 100 heads in a row would be a very unlikely outcome while seeing 50 heads and 50 tails would be highly likely. We will then parametrise the *prior* which will encode our belief in the parameter value of the system, i.e. how likely do we believe the coin to be biased before we see data. Having these two objects we have completely specified our model and can generate data. Now the *inference* procedure starts where we try to reach the *posterior* distribution. The posterior distribution is the distribution that encapsulate both our prior belief about the system with the evidence in the data.

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,

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

This distribution takes a single parameter μ which completely parametrises our likelihood. This means, we have a conditional distribution and the system is completely specified by 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 $\mathbf{x}$. Think a bit about what this assumption implies. It means that,

1. each data-point is independent as our likelihood function is invariant to any permutation of the data.

2. it assumes that each of the data-points are generated by the same distribution. For the coin example this means that by tossing the coin you do not change the actual coin by the act of tossing it[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[2] 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 any function. Now we have choosen our prior we are ready to derive the posterior.

[1] this is probably an OK assumption for metal coins while for chocolate coins this assumption is too simplistic

[2] its a bit more complicated than this but you have to believe me on this right now but we will see more of this later in the unit

3 Posterior

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

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

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,

$$
p(\mu|\mathbf{x}) \propto \mu^{\underbrace{\sum_i x_i + a - 1}_{a_n - 1}}(1-\mu)^{\underbrace{\sum_i (1-x_i) + b - 1}_{b_n - 1}}.
$$

Now what is left is to make sure that the expression is actually a probability distribution such that it integrates to one. This means that we need to solve the following,

$$
1 = Z\int p(\mu|\mathbf{x})\mathrm{d}\mu = Z\int \mu^{a_n-1}(1-\mu)^{b_n-1}.
$$

In this case this is trivial as we know the normaliser of the beta distribution which means that,

$$
Z = \frac{\Gamma(a_n+b_n)}{\Gamma(a_n)\Gamma(b_n)}
$$

This mean that my posterior is,

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

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.

4 Implementation

Common practice if you want to test something is to generate data from your model with known parameters, throw away the parameters and then see if you can recover the parameter. What we first then want to do is to sample a large set of binary outcomes. We can do this by using the `Binomial`[3] in `numpy`. 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

[3] `https://en.wikipedia.org/wiki/Binomial_distribution`

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,

Code

```
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 = 100

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

# now lets define our prior
a = 10
b = 10

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

# 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)

y = posterior(a,b,X)
plt.plot(mu_test,y,'b',linewidth=4.0)

# ignore this
plt.tight_layout()
plt.savefig(path, transparent=True)
return path
```

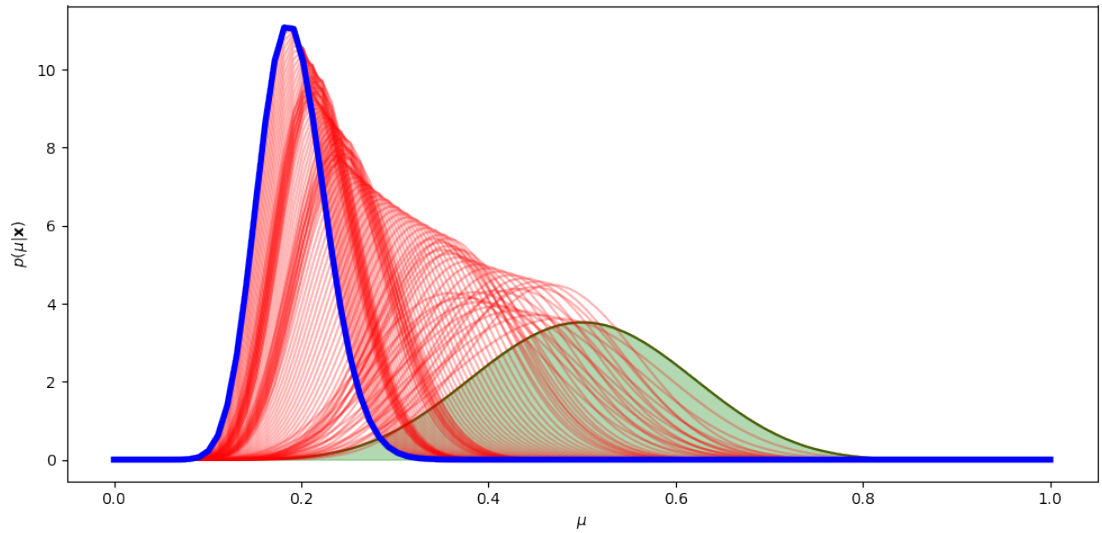

Figure 1: *The green distribution is the prior distribution over μ and the red distributions are the updated belief, our posterior when I see more and more data-points and the blue is the final posterior when we have seen all the points.*

As you can see from the plot above the posterior distribution places significant mass over the value of μ that was used to generate the data. Furthermore, when there is very little evidence i.e. we have seen very little data, the posterior is very close to the prior.

Now when we have built our model we can try out lots of different things, implement the tasks below and evaluate what they mean

1. what happens if you choose a prior that is very confident at a place far from the true value?
2. what happens if you choose a prior that is very confident at the true value?
3. in the plot above we cannot see the order of the red lines, create a plot where the `x-axis` is the number of data-points that you have used to compute the posterior and the `y-axis` is the distance of the posterior mean to the prior mean.
4. How much does the order of the data-points matter? Redo the plot above but with many different random permutations of the data. Now plot the `mean` and the `variance` of each iteration. What can you see?
5. Even disregarding computational benefits why does it make sense too choose the conjugate prior?

5 Summary

The intention of this lab was to show the mechanism that allows us to learn from data, how we can take our prior beliefs and update them by learning from data. Even though a coin-toss might be a silly example[4] it importantly contains all the elements of what we will do over the next few weeks. So your task is to really try and abstract this simple example to what it really "means" and then you will find the more mathematically complicated tasks later a lot simpler.

[4]not if you are playing cricket where if you win the toss you **always** choose to bat