Univariate Gaussians

The Gaussian distribution, also called the normal distribution, is widely used in probabilistic machine learning. This week we'll see Gaussians in the context of doing some basic statistics of experimental results. Later in the course we'll use Gaussians as building blocks of probabilistic models, and to represent beliefs about unknown quantities in inference algorithms.

We know that many of you will have seen all of this material before (some of you several times). However, not everyone has, and large parts of the MLPR course depend on thoroughly understanding Gaussians. This note is more detailed (slow) than many of the machine learning text books, and provides some exercises. A later note on multivariate Gaussians will also be important.

1 The standard normal distribution, the zero-mean unit-variance Gaussian

I think of a probability distribution as an object, like a black box, that outputs numbers. A number-producing box for the standard normal distribution has a label, $\mathcal{N}(0,1)$, and a button. If we press the button, the box displays a number. You can simulate this process at a python prompt by typing `np.random.randn()` (after importing numpy as usual). If you 'press the button' multiple times, or run `randn()` multiple times, you'll get different numbers. To attach a label to an outcome from the generator we write in mathematics $x \sim \mathcal{N}(0,1)$, or in code `x = np.random.randn()`.

Each call to `randn()` gives a different number, and many calls reveal the distribution encoded in the generator. A histogram of a million draws from the distribution, reveals a classic 'bell-curve' shape. You should be able to sketch this bell-shaped curve, with points of inflection at ± 1. Try plotting a histogram of a million standard normal samples on a computer now. If you know how to do it, opening a terminal and creating the plot can be done in about 10 seconds. If you don't know how to do it, learning to sample test data and to create plots is an important skill to develop. Example code is at the end of this document.

The observed or empirical mean of the samples will be about 0, and the empirical variance will be about 1. Check the mean and variance of the samples you've just simulated! In the limit of infinitely many samples, these values become exact, hence the numbers in "$\mathcal{N}(0,1)$", the description of the distribution. Formally: $\mu = E[x] = \int x\, p(x)\, \mathrm{d}x = 0$ and $\mathrm{var}[x] = E[(x-\mu)^2] = \int x^2\, p(x)\, \mathrm{d}x = 1$. If you don't know these definitions, please work through the review sheet on expectations in the background material section.

The *probability density function* (PDF) for the standard normal distribution describes the bell-shaped curve:

$$p(x) = \mathcal{N}(x; 0, 1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}. \qquad (1)$$

The probability of landing in a narrow range of width δ between $x - \delta/2$ and $x + \delta/2$ is about $p(x)\delta$. This statement becomes accurate in the limit $\delta \to 0$. So with $N = 10^6$ samples, we expect about $p(x)N\delta$ samples to land in a narrow histogram bin of width δ at position x. Using this approximation, you should be able to plot a theoretical prediction through the histogram you produced earlier.

2 Shifting and scaling: general univariate Gaussians

[The separate notes on expectations contain more material on shifting and scaling random variables.]

We can define a process to generate a quantity y that draws a value from a standard normal, $x \sim \mathcal{N}(0,1)$, and then adds a constant $y = x + \mu$. The mean of this distribution, is μ.

If we drew many values from $\mathcal{N}(0,1)$, and added some constant μ to each one, the histogram of the values keeps the same shape, and simply shifts to be centred at μ. The PDF will shift in the same way. This distribution has the same shape, just with a different location, and so is still called a Gaussian, just with mean μ instead of mean 0. To get the PDF of the new variable, we replace every occurrence of x with $(y-\mu)$:

$$p(y) = \mathcal{N}(y; \mu, 1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(y-\mu)^2}. \tag{2}$$

The choice of variable names, such as x or y, is arbitrary in this note. You will of course also see x used for Gaussians with non-zero mean.

Similarly, we can define a random variable z that multiplies or 'scales' a standard normal outcome by a constant σ, before adding a constant μ:

$$z = \sigma x + \mu. \tag{3}$$

If we scale and shift many draws from a standard normal, the histogram of values will be stretched horizontally, and then shifted. Scaling by σ multiplies the variance by σ^2 (see the notes on expectations), and leaves the mean at zero. Adding μ doesn't change the width of the distribution, or its variance, but adds μ to the mean.

The distribution of z maintains the same bell-curve shape, with the points of inflection now at $\mu \pm \sigma$ (note, *not* $\pm\sigma^2$). We still say the variable is Gaussian distributed, but with different parameters: $z \sim \mathcal{N}(\mu, \sigma^2)$. By convention, the second parameter of the normal distribution is usually its variance σ^2, not its width or standard deviation σ. However, if you are reading a paper, or using a new library routine, it is worth checking the parameterization being used, just in case. Sometimes people choose to define a Gaussian by its precision, $1/\sigma^2$, instead of the variance.

For a general univariate Gaussian variable $z \sim \mathcal{N}(\mu, \sigma^2)$, we can identify a standard normal, by undoing the shift and scale above:

$$x = \frac{z-\mu}{\sigma}. \tag{4}$$

We now work out the probability density for z, by transforming the density for this x. [See the further reading if you can't follow this section, or want more detail.]

Substituting the above expression into the PDF for the standard normal, suggests the shape of the shifted and scaled distribution that we imagined above is described by the density

$$p(z) = \mathcal{N}(z; \mu, \sigma^2) \propto e^{-\frac{1}{2\sigma^2}(z-\mu)^2}. \tag{5}$$

However, we have to be careful transforming real-valued random variables. Here, stretching out the bell-curve to be σ times wider would make the area underneath it σ times bigger. However, PDFs must be normalized: the area under the curve must be one. To conserve probability mass, if we stretch a region of outcomes, we must also decrease the PDF there by the same factor. Hence, the PDF of a general (univariate) Gaussian is that for a standard normal, scaled down by a factor of σ:

$$p(z) = \mathcal{N}(z; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(z-\mu)^2}, \tag{6}$$

which can also be written by replacing the σ in the denominator with σ^2 inside the square-root.

3 Check your understanding

3.1 Notation

[The website version of this note has a question here.]

3.2 Maths background

[The website version of this note has a question here.]

3.3 Code demonstration

[The website version of this note has a question here.]

4 Further reading

`https://en.wikipedia.org/wiki/Normal_distribution`

If you would like to work through the change of variables more rigorously, or in more detail, we are using a specific form of the following result:

If an outcome x has probability density $p_X(x)$, and an invertible, differentiable function g is used to create a new quantity $z = g(x)$, then the probability density of the new quantity is $p_Z(z) = p_X(g^{-1}(z))|\frac{\mathrm{d}x}{\mathrm{d}z}|$. In our case the derivative is simple: $1/\sigma$.

This method for transforming densities for a change of variables is in Bishop Section 1.2.1, with more detail and the multivariate case in Murphy Section 2.6.

5 Python code

To generate a million outcomes from a standard normal and plot a histogram:

```
import numpy as np
from matplotlib import pyplot as plt

N = int(1e6) # 1e6 is a float, numpy wants int arguments
xx = np.random.randn(N)
hist_stuff = plt.hist(xx, bins=100)
plt.show()
```

We increase the default number of bins from 10, which is not really enough to see the shape properly. We also want narrow bins to make an approximation suggested in the notes work well.

To check the mean and variance:

```
print('empirical_mean = %g' % np.mean(xx)) # or xx.mean()
print('empirical_var = %g' % np.var(xx))   # or xx.var()
```

You should understand how to plot a theoretical prediction of this histogram shape. Fill in the parts marked TODO below. If you don't know the answers, read the note again: it does contain both the PDF and how many samples we (approximately) expect to land in a narrow bin.

```
bin_centres = 0.5*(hist_stuff[1][1:] + hist_stuff[1][:-1])
# Fill in an expression to evaluate the PDF at the bin_centres.
# To square every element of an array, use **2
pdf = TODO
bin_width = bin_centres[1] - bin_centres[0]
predicted_bin_heights = TODO # pdf needs scaling correctly
# Finally, plot the theoretical prediction over the histogram:
plt.plot(bin_centres, predicted_bin_heights, '-r')
plt.show()
```