Univariate Gaussians: notes on answers

The Gaussian note asked some questions. Here are some of the answers. I advise having a good attempt before looking at these answers. I'm also giving some guidance on how you could check answers yourself, as outside of a class you have to become sure of your own answers.

1 Check your understanding further

- What is the integral $Z = \int e^{-\frac{1}{2\sigma^2}(u-\mu)^2} \, \mathrm{d}u$?

  As PDFs are normalized, $\int \mathcal{N}(u; \mu, \sigma^2) \, \mathrm{d}u = 1$. Comparing this integral to the one in the question will give you Z.

  You could check your answer numerically for a particular μ and σ. For example, in Matlab/Octave:

  ```
  mu = 2;
  sigma = 3;
  delta = sigma / 100;
  xx = (mu - 10*sigma):delta:(mu + 10*sigma);
  integrand = exp(-0.5*(xx - mu).^2/sigma^2);
  Zapprox = sum(integrand*delta)
  ```

  Or you could use a more clever numerical integration scheme, perhaps the `quad` function. If you wish to use Python:

  ```
  mu = 2.0
  sigma = 3.0
  delta = sigma / 100.0
  xx = np.arange(mu - 10*sigma, mu + 10*sigma, delta)
  integrand = np.exp(-0.5*(xx - mu)**2/sigma**2)
  Zapprox = np.sum(integrand*delta)
  print("Zapprox = %g" % Zapprox)
  ```

  Or you could use a routine from `http://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html`

- The integral $C = \int \sum_{k=1}^{K} w_k \mathcal{N}(x; \mu_k, \sigma_k^2) \, \mathrm{d}x$, is $\sum_{k=1}^{K} w_k$. You do not need to substitute in the PDF of a Gaussian, or do any maths. You simply need to use the fact that each Gaussian is normalized. Again, you could check the answer yourself numerically.

2 Matlab/Octave code

The full code to generate a million outcomes from a standard normal, and plot a histogram, along with its theoretical prediction:

```
N = 1e6;
xx = randn(N,1);
% 100 bars shows shape better than default of 10
hist(xx, 100);
[cc, bin_centres] = hist(xx, 100);
pdf = exp(-0.5*bin_centres.^2) / sqrt(2*pi);
bin_width = bin_centres(2) - bin_centres(1);
predicted_bin_heights = pdf * N * bin_width;
% Finally, plot the theoretical prediction over the histogram:
hold on;
plot(bin_centres, predicted_bin_heights, '-r', 'linewidth', 3);
```

3 Python code

The full code to generate a million outcomes from a standard normal, and plot a histogram, along with its theoretical prediction:

```
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)
bin_centres = 0.5*(hist_stuff[1][1:] + hist_stuff[1][:-1])
pdf = np.exp(-0.5*bin_centres**2) / np.sqrt(2*np.pi)
bin_width = bin_centres[1] - bin_centres[0]
predicted_bin_heights = pdf * N * bin_width
plt.plot(bin_centres, predicted_bin_heights, '-r', linewidth=3)
plt.show()
```