Self-test answers

Make sure to have a proper attempt at the questions before looking at these answers. If necessary, work through tutorials on probability, linear algebra, and calculus first. If you look at the answers before you've had an honest attempt, it will be too easy to trick yourself into thinking you have no problems.

I expect some of the class will have no trouble with any of these questions. If you couldn't do everything, that's also ok. However, to do well in this course, this material needs to become second nature. So if you're not nearly there, there isn't time to develop the mathematical experience required while taking the class.

This class will require you to apply mathematical reasoning, and *explain your answers*, not just mechanically push symbols around.

Certainly if you don't like the style of these answers, this isn't the course for you, as the notes are written by the same person!

1. The point of this question is to check that you know some of the basic rules of probability: the product rule, the sum rule, independence. You should know the formulae, and how to apply them to special cases.

   i) The conditional probability distribution is

   $$P(x \,|\, y\!=\!1) = P(x, y\!=\!1)/P(y\!=\!1) \propto P(x, y\!=\!1). \tag{1}$$

   You can find the marginal by the sum rule, $P(y = 1) = \sum_x P(x, y = 1)$. Or write down what the conditional distribution is proportional to, and spot the normalization (which is equivalent).

   $$P(x \,|\, y\!=\!1) \propto P(x, y\!=\!1) \propto \begin{cases} 1 & x = 1 \\ 3 & x = 2 \end{cases} = \begin{cases} \frac{1}{4} & x = 1 \\ \frac{3}{4} & x = 2. \end{cases} \tag{2}$$

   Your conditional distribution, like any distribution, should be normalized (add up to one). You should have noticed if it did not!

   ii) Two variables are independent if and only if $P(x, y) = P(x)P(y)$ for all x and y. Equivalently you can check whether it's always the case that $P(y \,|\, x) = P(y)$ or $P(x \,|\, y) = P(x)$.

   Here $P(x\!=\!1) = 0.3 \neq P(x\!=\!1 \,|\, y\!=\!1)$, so the variables are not independent. Computing other values would be a waste of time once you have already disproved independence. However, if the values had matched, you would have had to keep on checking.

2. Means and variances (not standard deviations) add for independent variables. (Means add for dependent variables too, but variances do not.) Multiplying an outcome by a constant increases the variance by the square of the constant. Review the note on expectations for more detail.

   Mean of Z: $\mu_x + 3\mu_y$

   Standard deviation of Z: $\sqrt{\sigma_x^2 + 9\sigma_y^2}$

3. This question was meant to test whether you are able to use mathematical knowledge to answer questions that aren't just mechanical evaluation of equations. Learning some rules isn't enough: you'll need the mathematical experience to be able to select relevant results, without being told which ones to use.

i) The product rule states:

$$P(X\!=\!x, Y\!=\!y) = P(X\!=\!x \,|\, Y\!=\!y)\, P(Y\!=\!y). \tag{3}$$

As $P(X\!=\!x \,|\, Y\!=\!y)$ is a probability, it is between 0 and 1. Therefore, removing it would increase the value of the right-hand side (unless it is one):

$$P(X\!=\!x, Y\!=\!y) \le P(Y\!=\!y), \tag{4}$$

with equality when $P(X\!=\!x \,|\, Y\!=\!y) = 1$.

To be thorough: The conditional probability is not defined when $P(Y\!=\!y) = 0$. In that case the inequality is also true, and with equality. (I forgot this case on first writing these answers.)

Alternatively you could write down the sum rule, and argue that the left-hand side is just one term in a positive sum that gives the right-hand side. Therefore the joint probability can't possibly be more than the marginal.

ii) In many situations, probability density functions (PDFs) can be treated as if they were probabilities. However, PDFs are not probabilities, and this question demonstrates that sometimes care is required.

As before, the product rule is:

$$p(x, y) = p(x \,|\, y)\, p(y). \tag{5}$$

However, $p(x \,|\, y)$ is not bounded above. For example, if x is forced to be equal to some function f of y, then $p(x \,|\, y) = \delta(x - f(y))$, where δ is a Dirac delta function. Then $p(x \,|\, y)$ is infinite at $x\!=\!f(y)$. Conditional densities can also be zero, therefore $p(x, y)$ can be more or less than $p(y)$.

If you didn't spot the difference from i), please make sure you understand how PDFs relate to probabilities.

These notes use $P()$ for probabilities and $p()$ for probabilities densities. This choice is not universal, but not uncommon either (example, example). Probability notation varies a lot. And there's a trade-off in being super-rigorous and unambiguous, and keeping the notation fairly clean. You will find that machine learning textbooks and papers are often even sloppier than these notes. While Murphy's book calls a density f when first reviewing probability, $p()$ is used throughout the book for discrete probability masses and probability densities, without regard for the difference. Bishop's book says that $p()$ is a density and $P()$ is a cumulative probability, but then also uses $p()$ for both discrete probabilities and probability densities throughout the book.

4. You can easily check this question for yourself on a computer.

Matlab/Octave:

```
A = [1 3 5; 4 6 0];
B = [0 5 2; 7 9 1; 8 2 3];
C = A*B;
size(C)
C(2,3)
```

Python:

```
import numpy as np

A = np.array([[1, 3, 5], [4, 6, 0]])
B = np.array([[0, 5, 2], [7, 9, 1], [8, 2, 3]])
C = np.dot(A, B)
print(C.shape)
print(C[2-1,3-1]) # -1's because Python uses 0-based indexes
```

5. The previous question tested a mechanical knowledge of how matrix multiplication works. You are also expected to have (or rapidly develop) some understanding of how matrices can correspond to transformations.

   The question said A is a 2×3 matrix, X is a $N\times3$ matrix.

   i) A new data matrix $Y = XA^\top A$ is created.

   $Z = XA^\top$ is a $N\times2$ matrix, so each row is a point that's now only in 2-dimensions.

   $Y = ZA$ transforms the 2-dimensional data Z back into 3-dimensions, however, all the points will still lie within a 2-dimensional plane contained within the three-dimensional space. It's not possible to recover the original positions X, as different values for a row of X would have been transformed to the same 2-dimensional point in $XA^\top$, and then the same 3-dimensional point in $XA^\top A$. Information was lost.

   Constructing a special case on your computer may help check your understanding. For example in Matlab/Octave:

```
N = 1000;
X = randn(N, 3);
A = randn(2,3);
Z = X*A'; % Nx2
Y = Z*A;  % Nx3
plot3(Y(:,1), Y(:,2), Y(:,3), '.');
```

   Click the rotation icon on the toolbar (Octave, older Matlab) or choose "Rotate 3D" from the tools menu (newer Matlab), and spin the points with the mouse. You'll see a 2-dimensional disc of points, embedded in the plotting cube. I think of a "pancake" of points, suspended in mid-air.

   Alternatively from a Python command-line prompt, you can run the following, and then spin the points using the mouse:

```
import numpy as np
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

N = 1000
X = np.random.randn(N, 3)
A = np.random.randn(2,3)
Z = np.dot(X, A.T) # Nx2
Y = np.dot(Z, A)   # Nx3
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
plt.plot(Y[:,0], Y[:,1], Y[:,2], '.')
plt.show()
```

   The Z points are a two-dimensional spray of points. Rotating, shearing and reflecting this disc of points, keeps it as a disc. That's all that's possible with a linear transformation A. There is no way to fill out the 3D space.

   ii) Because applying the matrix $A^\top A$ is an irreversible transformation, this matrix cannot be invertible.

   More technical answer: A has at most rank 2, so the product $A^\top A$ has at most rank 2, and has at most 2 non-zero eigenvalues. As the matrix has size 3×3, it is low-rank and is not invertible.

   iii) The matrix $AA^\top$ could be invertible. It transforms two-dimensional data, via a temporary transformation into 3D, which doesn't necessarily lose any information. For most 2×3 matrices A, the 2×2 matrix $AA^\top$ will be full rank.

Pathological case: if the columns of A are all multiples of each other, then $AA^\top$ will be low rank and not invertible.

For square matrices B and C, $(BC)^{-1} = C^{-1}B^{-1}$. However, we can't write $A^{-\top}A^{-1}$ here, because only square matrices have inverses, and A is not square.

6. Given the function $f(x,y) = (2x^2 + 3xy)^2$, the vector of partial derivatives is:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2(2x^2 + 3xy)(4x + 3y) \\ 6x(2x^2 + 3xy) \end{bmatrix}. \tag{6}$$

You might have written your answer in a different form, for example by multiplying out the brackets.

i) For a small vector $\delta = [\delta_x \; \delta_y]^\top$, the inner-product with the partial derivatives or gradient vector is

$$(\nabla f)^\top \delta = \frac{\partial f}{\partial x}\delta_x + \frac{\partial f}{\partial y}\delta_y. \tag{7}$$

For infinitesimal changes $\delta_x = \mathrm{d}x$ and $\delta_y = \mathrm{d}y$ this expression is the chain rule of partial derivatives, giving the infinitesimal change $\mathrm{d}f$ in the function. For small finite changes, it will still give the approximate change in the function value:

$$f(x+\delta_x,\, y+\delta_y) - f(x,y) \;\approx\; \frac{\partial f}{\partial x}\delta_x + \frac{\partial f}{\partial y}\delta_y. \tag{8}$$

This approximation can be used with small δ to numerically check the analytically-derived derivatives. I did such a check to reduce the chance of making an embarrassing mistake in these answers. In research or a job, where you won't have a lecturer providing answers, you'll need ways to check things too.

ii) We just saw that for moves from some position (x,y) to $(x+\delta_x, y+\delta_y)$, the inner product $(\nabla f)^\top \delta$ gives the change in the function value (in the limit of small δ).

$(\nabla f)^\top \delta = |\nabla f||\delta|\cos\theta$, where θ is the angle between direction of the change and the gradient vector. Considering moves of the same length $|\delta|$, in different directions θ, the change in function value is maximized by $\cos\theta = 1$ or $\theta = 0$. The function is increased the most by moving in the direction of the gradient vector.

7. The Uniform$[0,1]$ distribution has PDF

$$p(x) = \begin{cases} 1 & 0 \le x \le 1 \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The variance of a distribution is

$$\mathbb{E}[x^2] - \mathbb{E}[x]^2 = \int_0^1 x^2 \,\mathrm{d}x - \left(\tfrac{1}{2}\right)^2 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}. \tag{10}$$

You might have computed $\mathbb{E}[(x-\mu)^2] = \mathbb{E}[(x-\frac{1}{2})^2]$ instead, which should have led to the same answer. (An exercise on the expectations sheet is to prove that these two expressions for variance are equivalent.)

This variance of standard deviations can be looked up online: `https://en.wikipedia.org/wiki/Uniform_distribution_(continuous)`

It's also easy to check numerically: `var(rand(1e7,1))` in Matlab/Octave or in Python: `np.var(np.random.rand(int(1e7)))`. These should give a number close to 1/12.

8. You might not have programmed in Matlab or Python before. You should have used a language that looks sufficiently similar that you can follow these answers.

   A Python function to find the minimum value in a list could look like this:

```
def min_of_list(my_list):
    """minimum value in sequence my_list"""
    cur_min = my_list[0]
    for item in my_list[1:]:
        if item < cur_min:
            cur_min = item
    return cur_min
```

   We would actually use Python's `min()` function, or `np.min()` on a NumPy array. Matlab/Octave also has a built in `min` function, which you should normally use. However, a Matlab/Octave implementation could look like this:

```
function the_min = min_of_list(my_list)
%MIN_OF_LIST smallest number in input array

cur_min = my_list(1);
for ii = 2:numel(my_list)
    item = my_list(ii);
    if item < cur_min
        cur_min = item;
    end
end
the_min = cur_min;
```

   Both functions above will work when the list has only one item. However, if I had tried to access `my_list[1]` in Python or `my_list(2)` in Matlab, the code would crash given a list of only one item. Would your function have been ok?

   Both of my functions will throw errors if the list contains no items. Crashing may be desirable, as arguably the minimum item isn't defined. Alternatively, you may wish to define some non-error behaviour for empty lists, such as (in Python):

```
if len(my_list) == 0:
    return None
```

   That code will still throw an error if my_list isn't a sequence that `len()` understands. One way to prevent those errors is by brute force:

```
try:
    cur_min = my_list[0]
except:
    return None
```

   However, blanket catching of all errors (including ones you haven't thought of) is usually bad practice. It will make the program harder to debug later.

   `cur_min` has to be defined before we can start comparing to it. I have previously seen answers that declare `cur_min = 0` at the start, which would give the wrong answer if every item in the list was positive. You could initialize with ∞ if you assume the list contains numbers, and your programming language knows how to deal with ∞. However, I thought using one of the data items was easier and more general.

   Even a trivial function can require a little bit of thought, and a little bit of experience to get right! Those of you with some programming experience should have no trouble. However, if you haven't done any programming before, expect to do a serious amount of work to catch up.

   Maths background is more important than programming background. The assessed assignment is only worth 20% of the course. Also, the programs you write for this course will be fairly simple but mathematically dense.