Duration: **120 minutes**
Aids Allowed: **Non-programmable calculator**

**Student Number:** |_|_|_|_|_|_|_|_|_|_|

**Family Name(s):** SOLUTIONS

**Given Name(s):** _____

**Lecture Section:** ☐ Afternoon Section

☐ Evening Section

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

---

This term test consists of 6 questions on 22 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete, fill in the identification section above, and write your name on the back of the last page.*

Answer each question directly on the test paper, in the space provided, and use the reverse side of the pages for rough work. If you need more space for one of your solutions, use the reverse side of a page and *indicate clearly the part of your work that should be marked.*

You will not lose marks for minor syntax errors. If you cannot complete a helper function that is required to solve a problem, you should clearly indicate what the function that you cannot complete is supposed to do. You will receive part marks as appropriate for knowing what kind of helper function is needed and for using it correctly.

Write up your solutions carefully! Comments and docstrings are *not* required to receive full marks. However, they may help us mark your answers, and part marks *will* be given for showing that you know the general structure of an answer, even if your solution is incomplete.

Marking Guide

# 1: _____/ 15

# 2: _____/ 20

# 3: _____/ 20

# 4: _____/ 20

# 5: _____/ 15

# 6: _____/ 10

TOTAL: _____/100



*Good Luck!*

over...

*Use this page for rough work—clearly indicate any section(s) to be marked.*

## Question 1. [15 MARKS]

### Part (a) [5 MARKS]

Give an example of a dataset to which you could apply linear regression. State what the goal of applying linear regression to that dataset is, and make up a few numerical values for datapoints to make it clear what the data could look like.

**Solution**: Suppose we have a dataset which consists of characteristics of houses and the prices of the houses. The goal of running linear regression on that dataset may be to *predict the prices of houses for which we know the characteristics.* (Alternatively: the goal might be to *understand the relationship between the characteristics of a house and its price.*) The training set might look like this:

| | Number of floors | Surface area | Price |
|---|---|---|---|
| House 1 | 1 | 1,000 | 1,200,000 |
| House 2 | 2 | 800 | 700,000 |
| House 3 | 1 | 500 | 600,000 |

**Marking scheme**: 1 point for a dataset, 2 points for a numerical example that makes sense, 2 points for stating a sensible goal.

### Part (b) [10 MARKS]

Describe how you would obtain the parameters of the linear regression and how you would use those parameters to obtain predictions for new data. You **do** need to give details such as the cost function, but you **do not** have to give the details of Gradient Descent.

**Solution**: we need to obtain the parameters of the linear regression $\theta$, where $y \approx \theta^T x$. In the example above, $y$ is the price and $x$ is the vector that contains the characteristics of a house, for one house. To do that, we need to obtain the $\hat{\theta}$ that minimizes the cost function

$$\sum_i (\hat{\theta}^T x^{(i)} - y^{(i)})^2.$$

Here, $(x^{(i)}, y^{(i)}$ are the characteristics of the i-th input and the output for the i-th input in the training set, respectively. The cost function could be minimized using gradient descent, or using calculus by setting the derivative of the cost function with respect to $\hat{\theta}$ to 0.

Once the $\hat{\theta}$ that minimizes the cost function is obtained, we can use it to obtain predictions for new inputs $x$ using $\hat{y} = \hat{\theta}^T x$.

**Marking scheme**: 4 points for the cost function, 2 points for saying that it needs to be minmized. 4 points for saying how to use the argmin of the cost function to make new predictions.

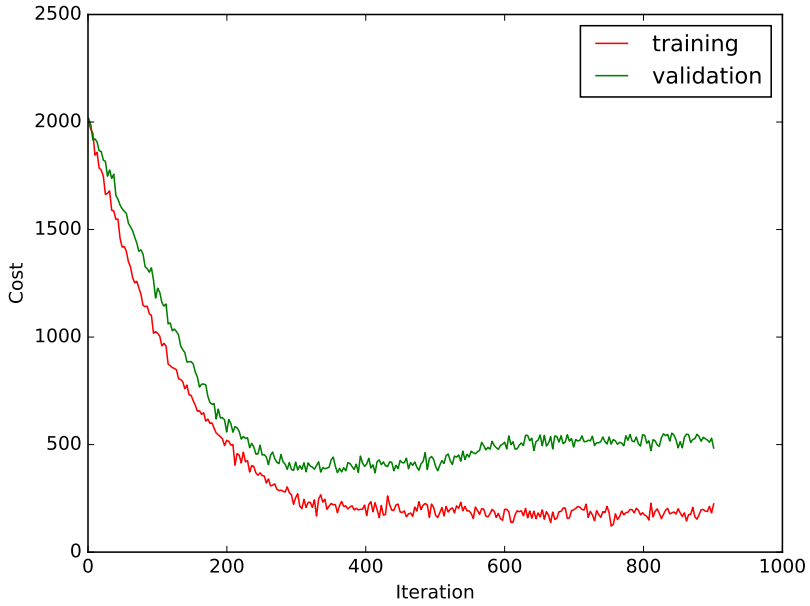
*Use this page for rough work—clearly indicate any section(s) to be marked.*

## Question 2. [20 MARKS]

### Part (a) [10 MARKS]

Draw the typical learning curves for the training and the validation set when training a neural network using stochastic gradient descent. Plot the cost on the y-axis and the iteration number on the x-axis. List the properties of the learning curves that you were trying to demonstrate in your sketch. Make sure to demonstrate all the properties that you would typically observe.

**Solution:**
A sample sketch is shown below.




Here are some features to point out:

- The validation performance is worse than the training performance
- The performance on the training set generally is increasing
- The performance starts out being the same
- The graph is jittery because the gradient descent is stochastic (in fact, the jitter is somewhat correlated in the training and the validation curves because if we do worse on the training set we likely also aren't doing well on the validation data, but this is not shown on this graph (note: people are not expected to say this))
- The performance on the validation set goes up during later iterations, and then stabilizes somewhat above the training performance.

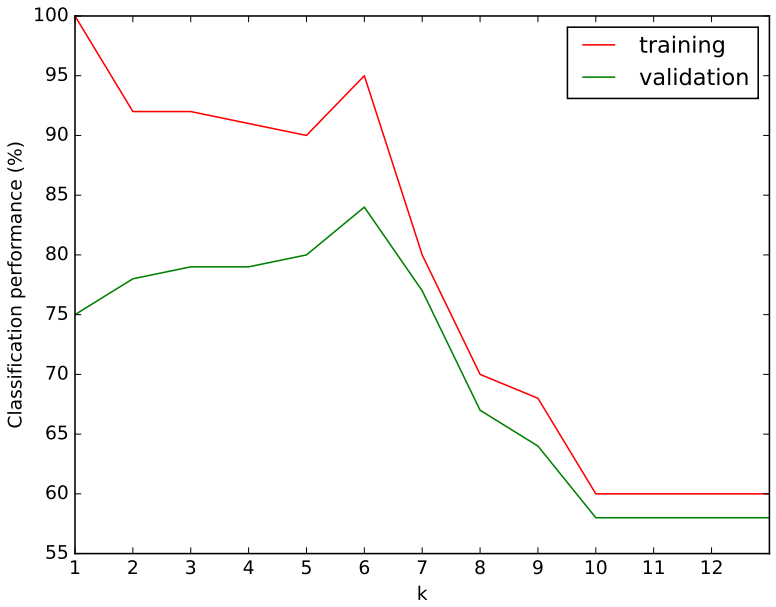
**Marking scheme:** 2 points per property (if a property is obvious in the graph, don't deduct points). 1/2 points for the last property if the validation cost keeps going up for larger iterations. Other reasonable properties could be considered.

*Use this page for rough work—clearly indicate any section(s) to be marked.*

## Part (b) [10 MARKS]

Draw the graphs of the performance when applying k-Nearest Neighbours on the training and validation sets (on the y-axis) versus $k$ (on the x-axis) that you would typically observe. List the properties of the graphs that you were trying to demonstrate in your sketch. Make sure to demonstrate all the properties that you would typically observe.

**Solution:**
A sample sketch is shown below.




Here are some features to point out:

- The validation performance is worse than the training performance
- The performance on the training set starts at 100% for $k = 1$
- There is a maximum of the performance attained by the validation set for some $k$
- For the training set, the performance stabilizes on the proportion of the training set that the plurality element occupies
- The maximum perofrmance for the validation set is near the local maximum of the perofrmance for the training set
- The performance keeps decreasing with $k$ after a certain point

**Marking scheme:** 2.5 points per property (if a property is very obvious in the graph, don't deduct points). Other reasonable properties could be considered. Note that there are 6 properties listed.

*Use this page for rough work—clearly indicate any section(s) to be marked.*

## Question 3. [20 MARKS]

You are given a Python function `def f(X)` which you can call, but whose source code you cannot look at. The function takes in a 42-dimensional NumPy array and returns a `float`. Write Python code that finds and prints a local **maximum** of the function `f` using Gradient Descent.

**Solution:** The idea here is to use finite approximation to compute the gradient, and to then follow the gradient. An implementation (together with a sample run) is below.

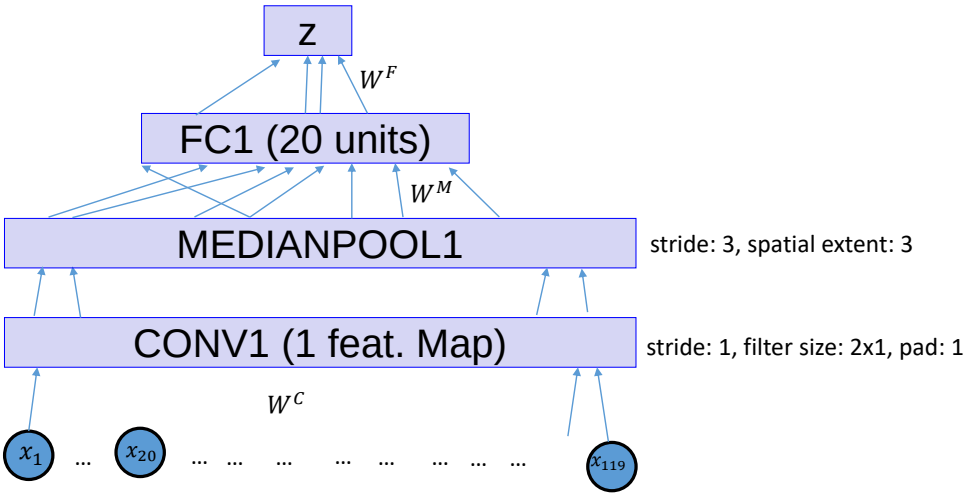
```
def grad(f, x):
    h = 0.0001
    grad = zeros(x.shape)
    for i in range(len(x)):
        xih = x.copy()
        xih[i] += h
        grad[i] = (f(xih)-f(x))/h

    return grad


def grad_ascent(f, init_guess):
    EPS = 0.0001
    alpha = 0.01

    prev_val = 0

    cur_guess = init_guess.copy()

    while linalg.norm(f(cur_guess)-prev_val) > EPS:
        prev_val = f(cur_guess)
        g = grad(f, cur_guess)
        cur_guess += alpha * g
        print cur_guess, f(cur_guess)

    return cur_guess

def f(x):
    return 50*x[2] + 100*x[0] + -x[2]**2 + x[1] - 5 - x[0]**2 - x[1]**2

print grad_ascent(f, array([0.1,0.02 ,0.05]))
```

**Marking scheme**: 10 points for computing the gradient (another option, which involves moving coordinate-by-coordinate, is okay), 10 points for gradient ascent. -3 points for doing descent instead. 2-3 points for minor substantive mistakes (e.g., -2 points for no stopping condition that makes sense). Synax errors are OK.

*Use this page for rough work—clearly indicate any section(s) to be marked.*

## Question 4. [20 MARKS]

Consider the Convolutional Neural Network below.




The network takes in an input of dimension $119 \times 1$, and its output is of dimension $1 \times 1$. The network consists of the input layer X (with a 0-pad of witdth 1), a convolutional layer CONV1 which consists of one feature map with a $2 \times 1$ filter which uses the *ReLU* nonlinearity, a median-pooling layer MEDIANPOOL1, a fully-connected layer FC1 which uses a *ReLU* nonlinearity, and an output layer Z of size $1 \times 1$, which is fully connected to the FC1 layer and uses a *sigmoid* nonlinearity. Recall that $\sigma'(t) = \sigma(t)(1 - \sigma(t))$.

Denote the weight that connects the i-th unit in FC1 to Z by $W_i^F$ and the bias for Z by $b^F$. Denote the weight that connects the j-th unit in MEDIANPOOL1 to the i-th unit in FC1 by $W_{ji}^M$ and the bias of the i-th unit in FC1 by $b_i^M$. Let $W^C = [W_1^C, W_2^C]$ and the bias for the CONV1 layer be $b^C$.

A unit in a median-pooling layer outputs the median value of the neurons in its receptive field (i.e., the neurons connected to the unit).

### Part (a) [4 MARKS]

How many parameters (i.e., values that specify how the network computes its output) are there in this network? Briefly show your work.

**Solution:** 2 weights and 1 bias for $W^C$, plus $(120/3) \times 20 + 20$ weights/biases in $W^M$, plus $20 + 1$ weights/biases in $W^F$, for a total of 844.

**Marking scheme**: 2 points for computing almost the right number for $W^M$, 1 point each for other stuff.

*Use this page for rough work—clearly indicate any section(s) to be marked.*

Let the training set inputs be $X = [X^{(1)}, X^{(2)}, ..., X^{(N)}]$ and the expected outputs be $Y = [Y^{(1)}, Y^{(2)}, ..., Y^{(N)}]$.

Let the outputs of the layers in the network be denoted using $c(X^{(i)})$, $m(X^{(i)})$, $f(X^{(i)})$, and $z(X^{(i)})$ for the CONV1, MEDIANPOOL1, FC1, and Z layers, respectively (you may use notation such as $z_i$, $f_j$, etc.). You may use those without explicitly telling us how to compute them.

The cost function is

$$cost(X, Y) = \sum_n cost(X^{(n)}, Y^{(n)}) = \sum_n (-Y^{(n)} \log(z(X^{(n)})) - (1 - Y^{(n)}) \log(1 - z(X^{(n)})).$$

**Part (b)** [8 MARKS]

Compute $\partial Cost / \partial W_{ji}^M$, for a single training case $(x, y) = (X^{(1)}, Y^{(1)})$. Show the details of the computation. Use Backpropagation to obtain the final answer: show how you would compute the gradients layer-by-layer.

**Solution**:

$$\frac{\partial Cost}{\partial z} = (-y/z + (1 - y)/(1 - z))$$

$$\frac{\partial Cost}{\partial f_i} = \frac{\partial Cost}{\partial z} \frac{\partial z}{\partial f_i} = \frac{\partial Cost}{\partial z} W_i^F z(1 - z)$$

$$\frac{\partial Cost}{\partial W_{ji}^M} = \frac{\partial Cost}{\partial f_i} \frac{\partial f_i}{\partial W_{ji}^M} = \frac{\partial Cost}{\partial f_i} [f_i > 0] m_j$$

Here, $[a > 0]$ is 1 if $a > 0$, and 0 otherwise.

**Marking scheme:** 1 for the cost derivative, 2 for the derivative wrt f, 2 for the derivative wrt $W^M$, 2 for ReLU derivative, 1 for putting it all together layer-by-layer.

*Use this page for rough work—clearly indicate any section(s) to be marked.*

**Part (c)** [8 MARKS]

Compute $\partial Cost/\partial W_1^C$, for a single training case $(x, y) = (X^{(1)}, Y^{(1)})$. Show the details of the computation. Note: the padding is significant. Use Backpropagation to obtain the final answer: show how you would compute the gradients layer-by-layer.

**Solution**: Continuing on from Part (b):

$$\frac{\partial Cost}{\partial m_j} = \frac{\partial Cost}{\partial f_i}\frac{\partial f_i}{\partial m_j} = \frac{\partial Cost}{\partial f_i}[f_i > 0]W_{ji}^M$$

Now we need $\frac{\partial m_j}{\partial c_k}$. To compute it, we have to see that $c_k$ influences $m_j$ iff $c_k$ is the median of the triple of values to which it belongs.

So we have

$$\frac{\partial m_j}{\partial c_k} = \begin{cases} 0, k \notin [3j, 3j+1, 3j+2] \\ 0, k \in [3j, 3j+1, 3j+2] \text{ and } c_k \mathrel{!}= median([c_{3j}, c_{3j+1}, c_{3j+2}]) \\ 1, \text{otherwise} \end{cases}$$

Now, we have, using the multivariate chain rule,

$$\frac{\partial Cost}{\partial c_k} = \sum_j \frac{\partial Cost}{\partial m_j}\frac{\partial m_j}{\partial c_k}$$

(Note that we didn't *have* to have the sum, it was just convenient.)

Now, using the multivariate chain rule again, we have

$$\frac{\partial Cost}{\partial W_1^C} = \sum_k \frac{\partial Cost}{\partial c_k}\frac{\partial c_k}{\partial W_1^C}$$

Finally,

$$\frac{\partial c_k}{\partial W_1^C} = [c_k > 0]x_{k+1}$$

(Assuming that we index $c$ using 0-based indexing.)

vspace.1in

**Marking scheme**: 3 points for differentiating through the median, 3 points for differentiatin through conv, 2 points for putting it all together.

*Use this page for rough work—clearly indicate any section(s) to be marked.*

## Question 5. [15 MARKS]

A coin is flipped 100 times, and the results (1's and 0's) are stored in the array `flips`. The result of a flip is 1 with probability $\theta$, and 0 with probability $1-\theta$. We have the prior belief that $\theta \sim N(0.5, 0.15^2)$. Recall that the Gaussian pdf is

$$f(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

### Part (a) [5 MARKS]

Write Python code to obtain the Maximum Likelihood estimate of $\theta$ given the training set `flips`.

**Solution:** From lecture, the ML estimate is just the average of `flips`. (People who said do gradient descent on the likelihood and specified the Binomial likelihood are correct too)

```
np.mean(flips)
```

**Marking scheme:** Hard to get part marks here. If the solution is partly right, part marks should be awarded.

### Part (b) [10 MARKS]

Write Python code to obtain the Maximum A-Posteriori estimate of $\theta$ given the training set `flips`.

**Solution:** Several possibilities here, including optimizing the posterior using gradient descent. We'll do what we did in lecture.

```
def g(x, mu, sigma):
    return exp( (-(x-mu)**2)/(2*sigma**2))

def lik(flips, theta):
    return (theta**sum(flips))*((1-theta)**(len(flips)-sum(flips)))

flips = [0, 0, 1, 1, 0, 1, 0, 1, 1, 1]

mu = 0.5
sigma = 0.15

theta = arange(0, 1, 0.01)
posterior = g(theta, mu, sigma)*lik(flips, theta)

print theta[argmax(posterior)]
```

**Marking scheme:** 2 points for either trying to find the best posterior using optimization or using grid search. 2 points for computing the likelihood of the data. 6 points for putting it all together.

*Use this page for rough work—clearly indicate any section(s) to be marked.*

## Question 6. [10 MARKS]

For what kind of dataset does the expected performance **on the test set** of k-Nearest Neighbours always grow as k grows? Describe a way to randomly generate that kind of dataset (use pseudocode or Python). The dataset should have two possible labels (i.e., $y = 0$ or $y = 1$).

**Solution**: The assumption behind k-means is that if two points are close together, they are likely to have the same labels. If that assumption is incorrect, k-means would not work, and the best we can hope for is to just guess according to which element is the majority element. Doing anything else will just overfit on the training set, producing worse performance on the test set in expectation, assuming that there is a majority label which we could always select.

The easiest way to generate the dataset is

```
for i = 1, 2, 3, 4, 5, ..., N
  Generate random 2D coordinates X[i],
  Assign a random label Y[i] to them with probability 60\%
```

**Marking scheme**: Full marks for solutions that evince understanding of the assumptions behind k-NN. 5 points for generating sets that work without explaining that in general what you need is a dataset that violates the assumptions of k-NN.

*Use this page for rough work—clearly indicate any section(s) to be marked.*

*Additional page for answers*

**On this page, please write nothing except your name.**

**Family Name(s):** _____

**Given Name(s):** _____