Duration: **120 minutes**
Aids Allowed: **None**

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 test consists of 6 questions on 11 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.*

Write up your solutions carefully! If you are giving only one part of an answer, indicate clearly what you are doing. Part marks might be given for incomplete solutions where it is clearly indicated what parts are missing.

You **must** write the test in pen if you would like to potentially request for the test to be regraded.

MARKING GUIDE

1: _____/15

2: _____/20

3: _____/10

4: _____/15

5: _____/10

6: _____/20

TOTAL: _____/90

Good Luck!

OVER...

Question 1. [15 MARKS]

Part (a) [3 MARKS]

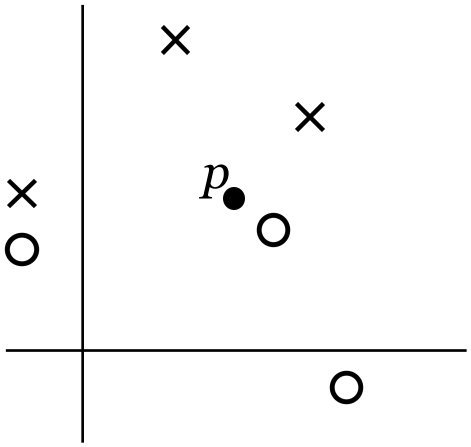
We would like to use 1-Nearest Neighbour to classify the point p as either X or O using the training data shown below. What is the prediction if cosine distance (i.e, negative cosine similarity) is used as the distance measure? What is the prediction if Euclidean distance is used?

(A) Cosine distance: O, Euclidean distance: X

(B) Cosine distance: O, Euclidean distance: O

(C) Cosine distance: X, Euclidean distance: X

(D) **Cosine distance: X, Euclidean distance: O**

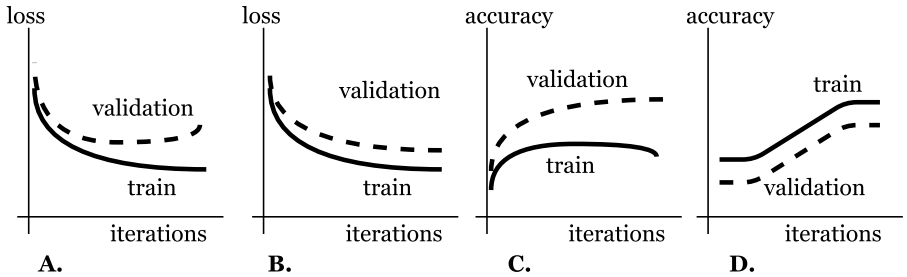


Explanation: the angle between vectors extending from the origin to p and a vector to an X is close to zero. The closest point to p is a circle.

Part (b) [3 MARKS]

Which of the following learning curves demonstrates overfitting? **Circle one of choices 1-5**.

1. **A**
2. B
3. C
4. D
5. A and C



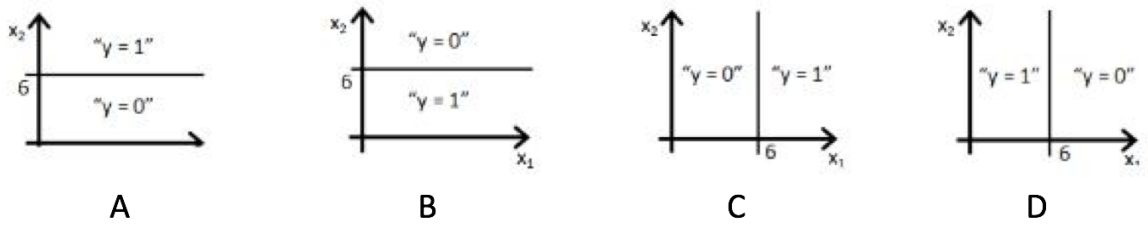
Explanation: the validation loss increased and the train loss decreased, indicating overfitting. The graph C is not a valid answer because training accuracy should be greater than validation accuracy during overfitting (in fact C is not a realistic training curve that you should encounter).

Part (c) [3 MARKS]

Suppose you train a logistic regression classifier and the learned hypothesis function is

$$h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2),$$

where $\theta_0 = 6, \theta_1 = 0, \theta_2 = -1$. Which of the following represents the decision boundary for $h_\theta(x)$?



A B C D

Explanation: the answer is **(B)**. We can rule out C and D because the decision boundary is independent of x_1. Additionally, $h_\theta(x)$ is smaller than 0.5 for $x_2 > 6$ (so the output is 0), and larger than 0.5 for $x_2 < 6$ (so the output is 1).

Part (d) [3 MARKS]

Alice and Bob, two students from the CSC411/2515 night section, walk home after lecture, and see what appears to be an alien spaceship floating in the sky. Alice concludes that aliens are real. How can we use Bayes' rule to explain why Bob might not reach the same conclusion?

(A) Bob's reasoning is more like gradient descent with momentum, which can give different results than gradient descent without momentum.

(B) Alice is using MAP inference, whereas Bob is using Maximum Likelihood inference.

(C) **Bob's prior beliefs about aliens' existence are different from Alice's.**

(D) None of the above.

Explanation: Bob's prior about aliens $P(aliens)$ is very small, and Alice's prior is larger, making it so that Alice's posterior $P(aliens|spaceship) = \frac{P(spaceship|aliens)P(aliens)}{P(spaceship)}$ is larger than Bob's. **(B)** is wrong because ML inference means comparing $P(spaceship|aliens)$ and $P(spaceship|\neg aliens)$. If Bob is doing that, he would more readily conclude that aliens exist. Other options don't really make sense.

Part (e) [3 MARKS]

Which of the following is (are) true about optimizers?

(A) **We can speed up training by using an optimizer that uses a different learning rate for each weight.**

(B) Dropout should not be used alongside momentum.

(C) Reducing the batch size when using Stochastic Gradient Descent always improves training.

(D) It does not make sense to use Stochastic Gradient Descent to train a linear regression model because linear regression is convex.

(E) All of the above.

Explanation: When we discussed optimizers, we discussed ones (like AdaGrad, Adam, etc) that use different learning rates for each weight. (B) does not make sense. (C) is wrong since just using one example per batch can cause gradient estimates to be too noisy. (D) is wrong because if your dataset is large, it makes sense to subsample to compute gradients faster.

.

Question 2. [20 MARKS]

Your training set is $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(m)}, y^{(m)})\}$. Assume that your model for the data is

$$y^{(i)} \sim \text{Laplace}(\theta^T x^{(i)}, 1).$$

The probability density function of the Laplace distribution with mean μ and scale parameter b is

$$f(x|\mu, b) = \frac{1}{2b} \exp\left(\frac{-|x-\mu|}{b}\right).$$

Part (a) [5 MARKS]

Write down the formula for the likelihood of the training set D.

Solution:

The likelihood of the training set is the product of the probabilities of the $y^{(i)}$s given the $x^{(i)}$s:

$$L_\theta(y|x) = \prod_{i=1}^{m} f(y|\theta^T x^{(i)}, 1) = \prod_{i=1}^{m} \left(\frac{1}{2} \exp(-|\theta^T x^{(i)} - y^{(i)}|)\right).$$

Marking scheme: 4 points for likelihood, 1 point for the product.

Part (b) [15 MARKS]

Suppose you would like to learn the model with penalized maximum likelihood, with an L1 regularization/penalty term. Derive the Gradient Descent update for this setting. Briefly justify every step. **Draw a rectangle around the formula that you derived**.

Solution:

The cost function is the negative log-likelihood plus a penalty term:

$$\begin{aligned} cost_{L1} &= -\log L_\theta(y|x) + \lambda||\theta||_1 && (1)\\ &= -\log \prod_{i=1}^{m} \left(\frac{1}{2} \exp(-|\theta^T x^{(i)} - y^{(i)}|)\right) + \lambda||\theta||_1 && (2)\\ &= -\sum_{i=1}^{m} (-|\theta^T x^{(i)} - y^{(i)}|) + \lambda \sum_j |\theta_j| && (3)\\ &= \sum_{i=1}^{m} |\theta^T x^{(i)} - y^{(i)}| + \lambda \sum_j |\theta_j| && (4) \end{aligned}$$

Now,

$$\boxed{\frac{\partial cost_{L1}}{\partial \theta_j} = \sum_{i=1}^{m} sign(\theta^T x^{(i)} - y^{(i)}) x_j^{(i)} + \lambda sign(\theta_j), sign(t) = \begin{cases} -1, & \text{if } t < 0 \\ 1, & \text{otherwise.} \end{cases}}$$

And the update rule is $\boxed{\theta \leftarrow \theta - \alpha \begin{bmatrix} \frac{\partial cost_{L1}}{\partial \theta_1} \\ \frac{\partial cost_{L1}}{\partial \theta_2} \\ \dots \\ \frac{\partial cost_{L1}}{\partial \theta_k} \end{bmatrix}}$.

Marking: 3 pts for Log-Likelihood, 2 pts for -LL+penalty, 3 pts for L1 penalty, 3 pts for gradient of cost + 2 pts for gradient of absolute value, 2 pts for update.

Question 3. [10 MARKS]

Suppose we are interested in a neuron in the 4-th layer of a convolutional neural network. What are two different methods of visualizing what the job of that neuron is? Do not just give names of algorithms or methods; give enough detail for the reader to be able to implement the methods you are describing.

Solution:

We discussed multiple ways of figuring out what a neuron is doing.

Find the parts of the image that activate the neuron the most Pass a the training set (or another large image set) through the network, and find the images (say 10 of them) for which the activation of the neuron is the highest. Display the parts of the images that are relevant to the neuron.

One quick-and-dirty way to obtain the parts of the image that are relevant to the input is to set all the weights in the network to 1, set all the biases to 0, and compute

$$\partial neuron/\partial x_i$$

for every i. The i's for which $\partial neuron/\partial x_i \; != 0$ will represent the indices of the pixels that are relevant to the neuron (this will be a square in the image).

Compute the gradient with respect to the input Find an input image x on which the output of the neuron is high (otherwise there is nothing to explain), and compute

$$\partial neuron/\partial x_i$$

for each dimension i for x. Then display the result as if it were an image: $\partial neuron/\partial x$ can be reshaped to have the same dimensions as x.

Use Guided Backprop to produce a cleaner visualization than is possible with gradient with respect to the input Same idea as above, but modify the gradient computation procedure. When the "gradient" with respect to the input is compute using backprop, at every iteration, zero out the negative components of $\partial layer_i/\partial layer_{i-1}$ before multiplying that by "$\partial neuron/\partial layer_i$" to obtain "$\partial neuron/\partial layer_{i-1}$". (Other ways to present this algorithm are possible)

Use gradient ascent Start with a random input. Keeping the weights fixed, use gradient ascent to maximize neuron activity. That is, use the following update rule

$$x \leftarrow x + \alpha \frac{\partial neuron}{\partial x}$$

until x converges. The converged x is an input image that will (locally) maximize *neuron* activation. This is a solution that we accepted, but in practise this method can be unpredictable.

Display the incoming weights to the neuron This is a neuron in the fourth layer of the ConvNet, so most solutions like that only got part marks. **If you assume** that the neuron is in the first fully-connected layer, you can treat the incoming weights are as the weights coming in from k "images." Those can be displayed somehow, if some thought is put into it.

Marking: 5 points for each of the two methods. 1 points for naming the method, 4 points for details that allow someone to implement the method.

.

Question 4. [15 MARKS]

Bob would like to classify emails as spam or non-spam. He would like to estimate the probability that a new email e containing the keywords $(w_1, w_2, ..., w_n)$ is spam by taking all the emails in the training set with those keywords, and then computing the proportion of those emails that are spam. Specifically, he estimates the probability using

$$P(\text{spam}|\text{new email } e) = \frac{\text{num. of spam emails with keywords } w_1, w_2, ..., w_n \text{ in the training set}}{\text{num. of total emails with keywords } w_1, w_2, ..., w_n \text{ in the training set}}.$$

Part (a) [3 MARKS]

Explain why Bob's plan will generally **not** work.

Solution: The main issue here is the *curse of dimensionality*. For reasonably-sized training sets, num. of spam emails with keywords $w_1, w_2, ..., w_n$ in the training set will just be 0, because it is unlikely for an email to contain n keywords for a large n. That means that we'll estimate 0 probability (or the probability will be undefined if the denominator is also 0) for most new emails.

Marking: 2 points for a good reason, 1 point for an explanation.

Part (b) [5 MARKS]

Describe the datasets for which Bob's plan might work. Be specific: state which properties are required of the datasets.

Solution: Basically, we need a large enough dataset. Here is one way to estimate how large: we want at least, say, 10 emails with each possible set of keywords. We'd need on the order of 2^n emails in the training set, where n is the number of keywords in the vocabulary.

Marking 4 points for explanation, 1 point for some kind of attempt to estimate the size of the training set.

Part (c) [5 MARKS]

Describe how to estimate the probability that an email is spam using the Naive Bayes assumption. Your answer should include formulas.

Solution: This question is just asking you to say how to use Naive Bayes for text classification. The important things to understand here are: using Bayes' rule to compute $P(spam|w)$ from $P(w|spam)$, factoring $P(w|spam)$ into $\prod_i P(w_i|spam)$ using the Naive Bayes assumption, and estimating the probabilities using counts.

Note that below, we'll definte the notation a little differently. First, assume there are n unique words in the training set. For a new email, let $w_i = 0$ if the email contains word with index i, and $w_i = 1$ otherwise.

Training:

$$\hat{P}(w_i = 1|spam) = \frac{\text{num. of spam emails containing } w_i}{\text{num. of spam emails}} \tag{5}$$

$$\hat{P}(w_i = 0|spam) = 1 - \hat{P}(w_i = 0|spam) \tag{6}$$

$$\hat{P}(w_i = 1|\neg spam) = \frac{\text{num. of non-spam emails containing } w_i}{\text{num. of non-spam emails}} \tag{7}$$

$$\hat{P}(w_i = 0|\neg spam) = 1 - \hat{P}(w_i = 0|\neg spam) \tag{8}$$

$$\hat{P}(spam) = \frac{\text{num. of spam emails}}{\text{total num. of emails}} \tag{9}$$

$$\hat{P}(\neg spam) = 1 - \hat{P}(spam) \tag{10}$$

Estimate using the Naive Bayes assumption:

$$\hat{P}(spam|w_1, w_2, ..., w_n) \propto \hat{P}(spam) \prod_{i=1}^{n} \hat{P}(w_i|spam)$$

$$\hat{P}(\neg spam|w_1, w_2, ..., w_n) \propto \hat{P}(\neg spam) \prod_{i=1}^{n} \hat{P}(w_i|\neg spam)$$

$$P(spam|w_1, w_2, ..., w_n) = \frac{\hat{P}(spam) \prod_{i=1}^{n} \hat{P}(w_i|spam)}{\hat{P}(spam) \prod_{i=1}^{n} \hat{P}(w_i|spam) + \hat{P}(\neg spam) \prod_{i=1}^{n} \hat{P}(w_i|\neg spam)}$$

Marking: 2 points for computing $\hat{P}(w_i = 1|spam)$, 2 points for computing $P(spam|w_1, w_2, ..., w_n)$, 1 point for putting it all together.

Part (d) [2 MARKS]

Why does the Naive Bayes assumption address a problem with Bob's plan?

We are now estimating $\hat{P}(w_i|spam)$ using $\frac{\text{num. of spam emails containing } w_i}{\text{num. of spam emails}}$, which will be a lot less sparse: the counts will be zero less frequently because we just need one keyword to appear in the training email instead of a combination of many keywords. We need a much more reasonably-sized training set.

.

Question 5. [10 MARKS]

Explain why a one-hidden-layer neural network is less likely to overfit if it has fewer units in the hidden layer. You should assume the reader is familiar with neural networks, but you should not assume that the reader is familiar with other concepts from CSC411/2515.

Solution:

The big idea here is that more neurons means that the network is able to represent a larger set of functions, so that the network can do very well on the training set, but possibly poorly on new data.

If the hidden layer is large enough, each neuron in the network can specialize in detecting a particular input in the training set (by making the weights incoming to it to look like that particular input, making the activation large on that particular input). That way, the network could get perfect performance on the training set, by connecting with a large positive weight the neuron in the hidden layer that is specializing in detecting a specific input $x^{(i)}$ to the output neuron that corresponds to the appropriate label $y^{(i)}$.

However, this would tend to make it so that the network is modelling the peculiarities of the training set – "templates" for inputs that look like examples in the training set are unlikely to work very well on new input.

A smaller network cannot do the same thing: the best thing to do for a smaller network is to make the hidden units activate on many possible inputs. This means that the neurons will be more likely to work appropriately on new inputs.

Note: One way to prove that a one-hidden-layer neural network (A) with more hidden units will have greater capacity than the same network with fewer hidden units (B), is that any function in the hypothesis space of A is also in the hypothesis space of B. That is, for any function in the hypothesis space of A, the same function can be recreated in B by using setting the weight going in and out of the additional neurons to zero. We did not require this proof.

Marking: 10 points for good answers, down to 5 for worse answers that contain correct statements.

Question 6. [20 MARKS]

The Poisson distribution is used to model data that consists of non-negative integers. Its probability mass function is $p(k) = \frac{\lambda^k e^{-\lambda}}{k!}$. For the dataset $\{k_1, k_2, k_3, ..., k_n\}$, the maximum likelihood estimate for λ is $\frac{1}{n}\sum_{j=1}^{n} k_j$.

Suppose you observe m integers in your training set. Your model assumption is that each integer is sampled from one of two different Poisson distributions. You would like to learn this model using the EM algorithm.

Part (a) [2 MARKS]

List all the parameters of the model.

Solution:

π_1, π_2 – the probabilities that a sample would come from the first or the second probability distribution. (N.B.: we could also just set $\pi_2 = \pi$ and have $\pi_1 = 1 - \pi$).

λ_1, λ_2 – the parameters of the two Poisson distributions.

Marking: 1 point for each of π and λ.

Part (b) [4 MARKS]

What is the likelihood of the training set under the model?

Solution: We accepted various answers. If we don't know the assignments $\{z_i\}$ (with $z_i = 1$ if k_i came from the first Poisson distribution), we can write:

$$P(k_1, k_2, ..., k_m|\pi, \lambda) = \prod_{i=1}^{m} \sum_{z_i \in \{1,2\}} P(k_i, z_i|\pi, \lambda) = \prod_{i=1}^{m} \sum_{z_i \in \{1,2\}} P(k_i|z_i, \lambda) P(z_i|\pi) = \prod_{i=1}^{m} \sum_{z_i \in \{1,2\}} p_{\lambda_{z_i}}(k) \pi_{z_i}.$$

If we do know the z's, we can write

$$P(k_1, k_2, ..., k_m|\pi, \lambda, z) = P(k_1, k_2, ..., k_m|\lambda, z) = \prod_{i=1}^{m} p_{\lambda_{z_i}}(k_i)$$

Here, $p_\lambda(k) = \frac{\lambda^k e^{-\lambda}}{k!}$.

Marking: 1 point for decomposition, 1 point for mixing proportions, 1 point for correct likelihood terms, 1 point for combining correctly

Part (c) [7 MARKS]

Derive the E-step for this model. Your answer should include a mathematical justification. **Draw a rectangle around the formula that you derived**.

Solution:

We would like to calculate the *soft assignments* for each one of the ks. The intuition here is that the close the k is to λ_1, the more likely it is that it was genrated by the first distribution. We do the same thing we did for the Mixture of Gaussians model.

$$P(z_i = 1|k_i, \pi, \lambda) \propto P(k_i|z_i = 1, \pi, \lambda) P(z_i = 1) = p_{\lambda_1}(k_i)\pi_1$$
$$P(z_i = 2|k_i, \pi, \lambda) \propto P(k_i|z_i = 2, \pi, \lambda) P(z_i = 2) = p_{\lambda_2}(k_i)\pi_2$$

So the E-Step is:

$$\boxed{w_{i,1} \leftarrow P(z_i = 1|k_i, \pi, \lambda) = \frac{p_{\lambda_1}(k_i)\pi_1}{p_{\lambda_1}(k_i)\pi_1 + p_{\lambda_2}(k_i)\pi_2}, w_{i,2} \leftarrow P(z_i = 2|k_i, \pi, \lambda) = 1 - (z_i = 1|k_i, \pi, \lambda)}$$

Marking: 1 point for getting the general idea, 3 points for some progress with the math, 5 points for significant progress with the math.

.

Part (d) [7 MARKS]

Derive the M-step for this model. Your answer should include a mathematical justification. **Draw a rectangle around the formula that you derived**.

Solution:

We would now like to find the π and λ using the soft assignments.

We can compute the expectation of the number of samples from each distribution (note that this was not required for the solution – an answer involving the w's is enough).

$$N_1 = E[\sum_i I[z_i = 1]] = \sum_i P(z_i = 1) = \sum_i w_{i,1}$$

$$N_2 = E[\sum_i I[z_i = 1]] = \sum_i P(z_i = 2) = \sum_i w_{i,2}$$

Now the average of the integers for each distriubtion, using the soft assignments, is:

$$\mu_1 = \frac{\sum_i w_{i,1} k_i}{\sum_i w_{i,1}}$$

$$\mu_2 = \frac{\sum_i w_{i,2} k_i}{\sum_i w_{i,2}}$$

To estimate the π, using the expected counts: compute $N_1/(N_1 + N_2)$ and $N_1/(N_1 + N_2)$.

The M-step is then:

$$\boxed{\lambda_1 \leftarrow \mu_1 = \frac{\sum_i w_{i,1} k_i}{\sum_i w_{i,1}}, \lambda_2 \leftarrow \mu_2 = \frac{\sum_i w_{i,2} k_i}{\sum_i w_{i,2}}, \pi_1 \leftarrow \frac{\sum_i w_{i,1}}{\sum_i w_{i,1} + \sum_i w_{i,2}} \pi_2 \leftarrow \frac{\sum_i w_{i,2}}{\sum_i w_{i,1} + \sum_i w_{i,2}}}$$

Marking: 3 for correct update of the πs. 2 points for each of λ_1, λ_2.

Family Name(s): _____

Given Name(s): _____