University College Dublin
An Coláiste Ollscoile, Baile Átha Cliath

SEMESTER II EXAMINATIONS

ACADEMIC YEAR 2017/2018

COMP 47590

Advanced Machine Learning

Prof. J. Pitt

Prof. P. Cunningham

Dr. B. Mac Namee *

Time Allowed: 2 Hours

Instructions for Candidates

Answer any **four** out of five questions. All questions carry equal marks. Total marks available **100**. The value of each part of each question is shown in brackets next to it.

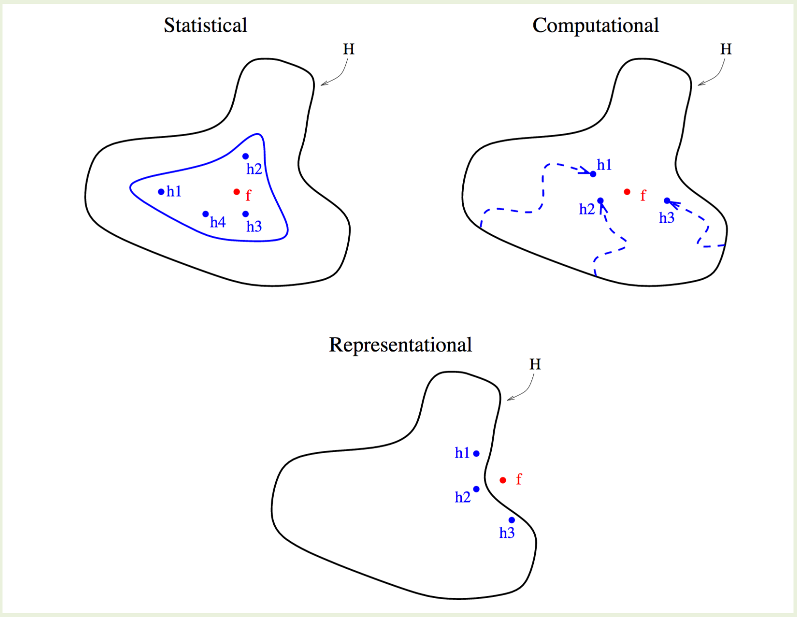
Instructions for invigilators

This is a Closed Book/Notes exam.
Students are **not** permitted to bring materials to the Exam Hall.
Non-programmable calculators allowed.

1.	**(a)**	The **Bayes Optimal Classifier** is defined as follows: $t = \underset{l \in levels(t)}{\operatorname{argmax}} \sum_{\mathbb{M}_i \in \mathbb{M}} P(l\|\mathbb{M}_i) P(\mathcal{D}\|\mathbb{M}_i) P(\mathbb{M}_i)$	
		(i)	Describe what the Bayes Optimal Classifier computes. In your answer ensure to describe each term in the summation.
			[4]
			<u>Sample Answer</u> Students should describe that the Bayes optimal classifier sums across every possible model in a hypothesis space computing these three components: $P(l\|\mathbb{M}_i)$: The probability of a target label given a model from the hypothesis space. $P(\mathcal{D}\|\mathbb{M}_i)$:The probability of the training dataset given a model from the hypothesis space. $P(\mathbb{M}_i)$: The prior probability of a specific model. from the hypothesis space.
		(ii)	Explain why the **Bayes Optimal Classifier** is never actually implemented in practice.
			[3]
			<u>Sample Answer</u> The Bayes optimal classifier cannot be used in practice for the following reasons: 1. Most interesting hypothesis spaces are too large to iterate over. 2. Computing an unbiased estimate of the probability of the training set given a hypothesis is non-trivial. 3. Estimating the prior probability for each hypothesis is rarely feasible.
	(b)	Thomas Deittrich describes three motivations for using ensemble models: **statistical**, **computational**, and **representational**. Describe each of these motivations and how they explain the performance of ensemble models.	
			[8]
		<u>Sample Answer</u>	

Students should treat these motivations one by one. A reproduction of the diagram below from (Deittrich, 2000) would be useful.

The statistical motivation is arises from the fact that we always have a sample of the full data space associated with a machine learning problem and so the likelihood of arriving at a hypothesis matching the true function we are trying to model is low. In fact in all likelihood we will arrive at multiple hypotheses that are all equally accurate in relation to the training dataset sample that we have available. By averaging the outputs of this set of models we are likely to arrive at an overall model that is more close to the true underlying function.



The computational motivation arises from the fact that most machine learning algorithms perform some form of local search through a hypothesis space and can stop at a local minimum rather than the global minimum. By averaging across many runs of this local search process (even if the individual runs result in local minima) we are likely to arrive a much better overall model.

The representational motivation arises from the fact that in many cases it is not possible to actually represent the true underlying function that we are trying to model using a particular modelling algorithm. In the diagram above we show that the true function, f, lies outside the hypothesis space. However, it is possible that the aggregate of an ensemble of models that can be represented will be closer to this true underlying model than any single model that can be represented - an ensemble allows us to jump outside of what can be represented.

(c) Benchmark experiments have found repeatedly that ensemble models based on **bagging** are more robust to noise in the target features of a training dataset than ensemble models trained using **boosting**. Explain why this is the case. In your answer provide a short explanation of the bagging and boosting techniques.

[5]

<u>Sample Answer</u>

		Bagging builds an ensemble by repeatedly performing bootstrap sampling with replacement on a training dataset and using these samples to train a set of base models. The outputs of these models are then aggregated using majority voting (for categorical targets) or averaging (for continuous targets). Boosting builds an ensemble by iteratively training models and adjusting a distribution across the training set based on the performance of the last model trained. In this way the next model trained will focus on the parts of the training set that the previous model struggled with as it takes this distribution into account during training. The reason that boosting is more sensitive to noise in the target features than bagging is that it is prone to over fitting to these noisy instances as models will typically struggle to correctly predict them which means subsequent models will focus too much on them.
	(d)	**Gradient boosting** has recently been shown to offer significant performance improvements over other boosted ensemble approaches. Explain what the gradient boosting algorithm trains its base models to predict.
		[5]

Sample Answer

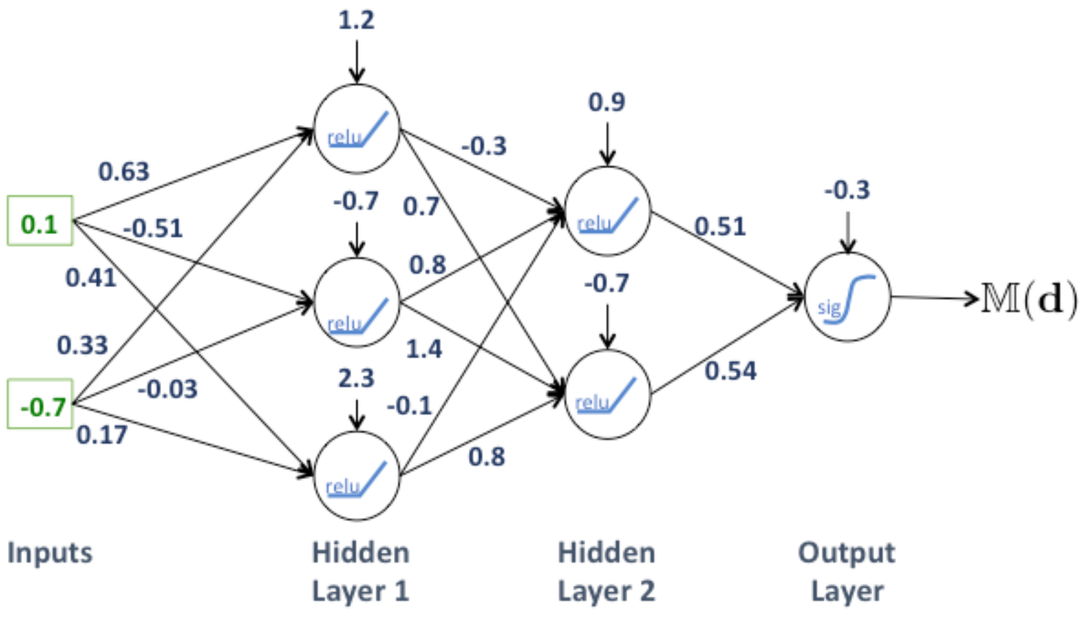
Students should explain that the gradient boosting approaching trains models sequentially like other boosting algorithms. However, rather than training each model to predict the target feature in the original training dataset, the algorithm trains each model to predict the errors made by the previous model:

$$\mathbb{M}_{iter\,N}\left(\mathbf{d}_i\right) = t_i - \mathbb{M}_{n-1}\left(\mathbf{d}_i\right)$$

So, a gradient boosting ensemble becomes:

Iteration	Ensemble Model
1	$\mathbb{M}_1 = \frac{\sum_{i=1}^{n} t_i}{n}$
2	$\mathbb{M}_2 = \mathbb{M}_1 + \mathbb{M}_{iter2}$
3	$\mathbb{M}_3 = \mathbb{M}_2 + \mathbb{M}_{iter3}$
	$\dots$
n	$\mathbb{M}_n = \mathbb{M}_{n-1} + \mathbb{M}_{iter\,N}$

2. **(a)** The image below shows a feed forward artificial network. The computational units in the two hidden layers use rectified linear (relu) activation functions and the output layer unit uses a sigmoid activation function. The weights and biases are shown along the links in the network.



(i) Perform a **forward propagation** through the network using an input feature vector of (0.1, -0.7).

[10]

- *Question 2 (a) – it may be helpful to state explicitly 'Show your working' to ensure that the students do not just give the answer but also show how they have come up with it*

Setup Input, Weight and Bias Matrices

The network inputs:

$$\mathbf{d} = \begin{bmatrix} 0.1 \\ -0.7 \end{bmatrix} \tag{1}$$

Weights and biases for **Layer 1**:

$$\mathbf{W}^{[1]} = \begin{bmatrix} 0.63 & 0.33 \\ -0.51 & -0.03 \\ 0.41 & 0.17 \end{bmatrix} \tag{2}$$

$$\mathbf{b}^{[1]} = \begin{bmatrix} 1.2 \\ -0.7 \\ 2.3 \end{bmatrix} \tag{3}$$

Weights and biases for **Layer 2**:

$$\mathbf{W}^{[2]} = \begin{bmatrix} -0.3 & 0.8 & -0.1 \\ 0.7 & 1.4 & 0.8 \end{bmatrix} \tag{4}$$

$$\mathbf{b}^{[2]} = \begin{bmatrix} 0.9 \\ -0.7 \end{bmatrix} \tag{5}$$

Weights and biases for **Layer 3**:

$$\mathbf{W}^{[3]} = \begin{bmatrix} -0.51 & 0.54 \end{bmatrix} \tag{6}$$

$$\mathbf{b}^{[3]} = \begin{bmatrix} -0.3 \end{bmatrix} \tag{7}$$

Forward Propagate

To perform a forward propagation for the first layer in the network, first calculate $\mathbf{z}^{[1]}$:

$$\begin{aligned}
\mathbf{z}^{[1]} &= \mathbf{W}^{[1]}\,\mathbf{d} + \mathbf{b}^{[1]} && (8)\\
&= \begin{bmatrix} 0.63 & 0.33 \\ -0.51 & -0.03 \\ 0.41 & 0.17 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.7 \end{bmatrix} + \begin{bmatrix} 1.2 \\ -0.7 \\ 2.3 \end{bmatrix} && (9)\\
&= \begin{bmatrix} 1.032 \\ -0.73 \\ 2.222 \end{bmatrix} && (10)
\end{aligned}$$

then apply the activation function, in this case a **relu function**, to calculate the activation of the nodes at **Layer 1**:

$$\begin{aligned}
\mathbf{a}^{[1]} &= \text{relu}(\mathbf{z}^{[1]}) && (11)\\
&= \text{relu}\left(\begin{bmatrix} 1.032 \\ -0.73 \\ 2.222 \end{bmatrix}\right) && (12)\\
&= \begin{bmatrix} 1.032 \\ 0 \\ 2.222 \end{bmatrix} && (13)
\end{aligned}$$

To perform a forward propagation for the second layer in the network, first calculate $\mathbf{z}^{[2]}$:

$$\begin{aligned}
\mathbf{z}^{[2]} &= \mathbf{W}^{[2]}\,\mathbf{a}^{[1]} + \mathbf{b}^{[2]} && (14)\\
&= \begin{bmatrix} -0.3 & 0.8 & -0.1 \\ 0.7 & 1.4 & 0.8 \end{bmatrix} \begin{bmatrix} 1.032 \\ 0.000 \\ 2.222 \end{bmatrix} + \begin{bmatrix} 0.9 \\ -0.7 \end{bmatrix} && (15)\\
&= \begin{bmatrix} 0.3682 \\ 1.8 \end{bmatrix} && (16)
\end{aligned}$$

then apply the activation function, in this case a **relu function**, to calculate the activation of the output nodes of the network:

$$\begin{aligned}
\mathbf{a}^{[2]} &= \text{relu}(\mathbf{z}^{[2]}) && (17)\\
&= \text{relu}\left(\begin{bmatrix} 0.3682 \\ 1.8 \end{bmatrix}\right) && (18)\\
&= \begin{bmatrix} 0.3682 \\ 1.8 \end{bmatrix} && (19)
\end{aligned}$$

To perform a forward propagation for the third layer in the network, first calculate $\mathbf{z}^{[3]}$:

$$\begin{aligned}
\mathbf{z}^{[3]} &= \mathbf{W}^{[3]}\,\mathbf{a}^{[2]} + \mathbf{b}^{[3]} && (20)\\
&= \begin{bmatrix} -0.51 & 0.54 \end{bmatrix} \begin{bmatrix} 0.3682 \\ 1.8 \end{bmatrix} + \begin{bmatrix} -0.3 \end{bmatrix} && (21)\\
&= \begin{bmatrix} 0.484218 \end{bmatrix} && (22)
\end{aligned}$$

then apply the activation function, in this case a **sigmoid function**, to calculate the activation of the output nodes of the network:

$$\begin{aligned}
\mathbf{a}^{[3]} &= \text{sigmoid}(\mathbf{z}^{[3]}) && (23)\\
&= \text{sigmoid}\left(\begin{bmatrix} 0.859782 \end{bmatrix}\right) && (24)\\
&= \begin{bmatrix} 0.70261511 \end{bmatrix} && (25)
\end{aligned}$$

(ii) If the target feature value for the current input vector is 1.0, calculate the **loss** associated with this training instance using **log loss**.

[3]

Calculate log loss

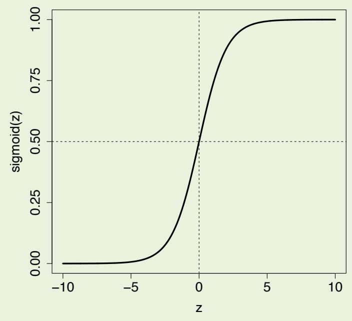
$$\begin{aligned}
\text{loss} &= -\left(t \times log\left(\mathbf{a}^{[3]}\right) + (1-t) \times log\left(1 - \mathbf{a}^{[3]}\right)\right) && (26)\\
&= -\left(1 \times log\left(\begin{bmatrix} 0.7026 \end{bmatrix}\right) + (1-1) \times log\left(1 - \begin{bmatrix} 0.7026 \end{bmatrix}\right)\right) && (27)\\
&= 0.3529 && (28)
\end{aligned}$$

	(b)	In modern artificial neural networks the previously popular **sigmoid** and **tanh** activation functions have been largely replaced with the **rectified linear** (**relu**) activation function. Describe each of these activation functions (include appropriate diagrams) and explain the advantage of using relu.
		[4]

Sample Answer

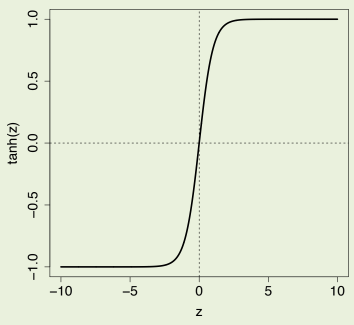
The three activation functions are described as follows.

Sigmoid:



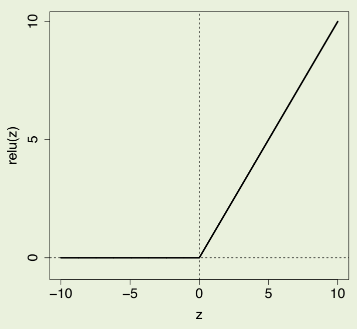
$$sigmoid(z) = \frac{1}{1+e^{-z}}$$

tanh:



$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

relu:



$$relu(z) = max(0, z)$$

Both the sigmoid and tanh squash their outputs into a small range, (0, 1) and (-1, 1) respectively. If these activation functions are used in a deep network signal can disappear due to repeated multiplication of the gradient associated with these functions be small weight values. This is one cause of what is referred to as the *vanishing* *gradient problem.* Because the relu activation function maintains positive values (which can be of arbitrary size) this does not happen.

(c) There are three common variants of the gradient descent algorithm when training deep neural networks: **batch gradient descent**, **mini-batch gradient descent**, and **stochastic gradient descent**. Describe each of these approaches and discuss the advantages and disadvantages of each.

[8]

Sample Answer

Students should first describe the three approaches as follows:

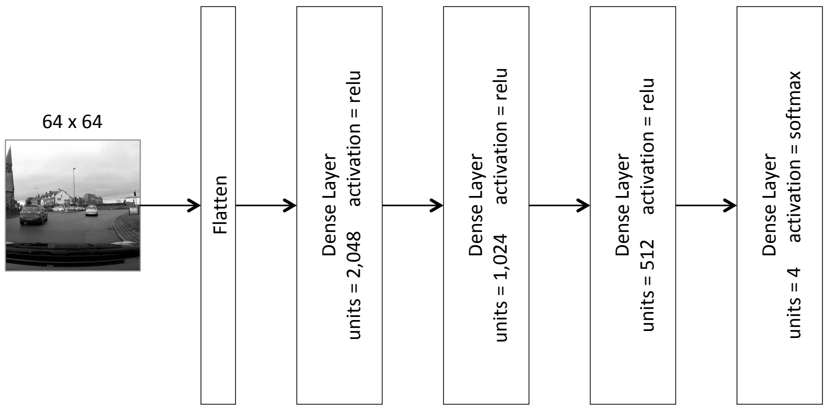
- **Stochastic gradient descent** a forward and backward pass through the network is made for each training instance and weights and biases are updated after each training instance is considered.
- **Batch gradient descent** a forward and pass through the network is made for each training instance and the losses are accumulated. Then a single backward pass through the network is made after all training instances have been considered. Weights and biases are updated only once after all training instances have been considered
- **Mini-batch gradient descent** the training dataset is divided into mini-batch of size b. For all of the training instances in a mini-batch a forward and pass through the network is made and the losses are accumulated. Then a single backward pass through the network is made after all training instances have been considered. Weights and biases are updated only once after all training instances in a mini-batch have been considered

The advantages and disadvantages of each approach are as follows:

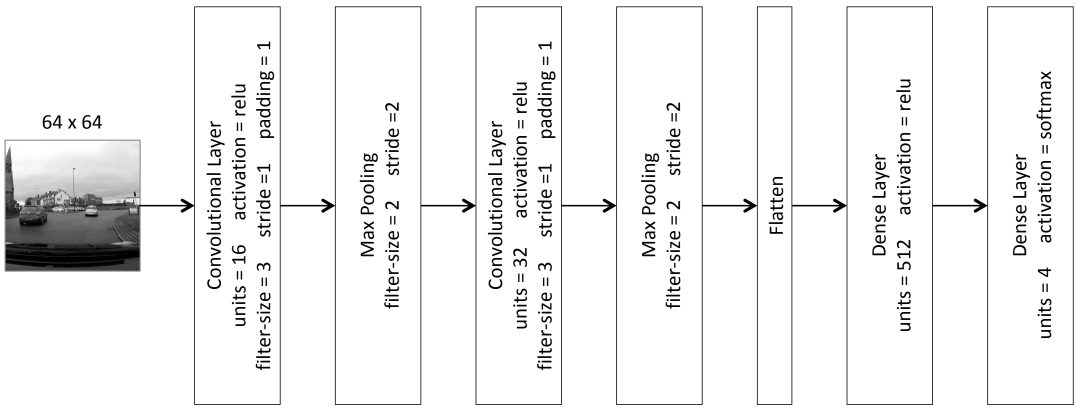
- **Stochastic gradient descent** is easy to implement and can result in fast learning, **but** is computationally expensive and can result in a noisy gradient signal
- **Batch gradient descent** is computationally efficient and can result in a stable gradient signal, **but** requires gradient accumulation, can result in premature convergence can require loading large datasets into memory and can become slow
- **Mini-batch gradient descent** is relatively computationally efficient, does not require full datasets to be loaded into memory, and can result in a stable gradient signal, **but** requires gradient accumulation, and introduces a new hyper-parameter - mini-batch size

Most modern implementations use mini-batch gradient descent.

3. **(a)** You have been tasked with building a neural network system for controlling a self-driving car. The car has just four controls: accelerate, brake, turn left, and turn right. The only input is a 64 pixel by 64 pixel image from a dashboard camera within the car. Image (a) shows a multi-layer perceptron neural network architecture designed for this problem. Image (b) shows a convolutional neural network architecture designed for this problem. Both architectures are composed of four layers.


64 x 64
Flatten
Dense Layer units = 2,048 activation = relu
Dense Layer units = 1,024 activation = relu
Dense Layer units = 512 activation = relu
Dense Layer units = 4 activation = softmax

(a) A multi-layer perceptron network for self-driving car control


64 x 64
Convolutional Layer units = 16 activation = relu filter-size = 3 stride =1 padding = 1
Max Pooling filter-size = 2 stride =2
Convolutional Layer units = 32 activation = relu filter-size = 3 stride =1 padding = 1
Max Pooling filter-size = 2 stride =2
Flatten
Dense Layer units = 512 activation = relu
Dense Layer units = 4 activation = softmax

(b) A convolutional neural network for self-driving car control

Calculate the number of parameters (weights and biases) that need to be learned in each network architecture.

[8]

<u>**Sample Answer**</u>

Multi-layer perceptron:

This is pretty straight-forward as it involves multiplying through the sizes of the network layers.

Input: 64 * 64 = 4,096

Layer 1: 4,096 * 2,048 + 2,048 = 8,390,656

Layer 2: 2,048 * 1,024 + 1,024 = 2,098,176

Layer 3: 1,024 * 512 + 512 = 524,800

Layer 4: 512 * 4 + 4 = 2,052

Total parameters: 8,390,656 + 2,098,176 + 524,800 + 2,052 = 11,015,684

Convolutional neural network:

Students need determine the number of weights based on the size of each filter and the size of each layer. To calculate the number of activations at the flattening layer they also need to keep track of the number of activations flowing through the network.

Layer 1: 3* 3 * 16 + 16 = 160

Layer 2: 3 * 3 * 16 * 32 + 32 = 4,640

Layer 3: (Activations = 16 * 16 * 32 = 8,192) 8,192 * 512 + 512 = 4,194,816

Layer 4: 512 * 4 + 4 = 2,052

Total parameters: 160 + 4,640 + 4,194,816 + 2,052 = 4,201,668

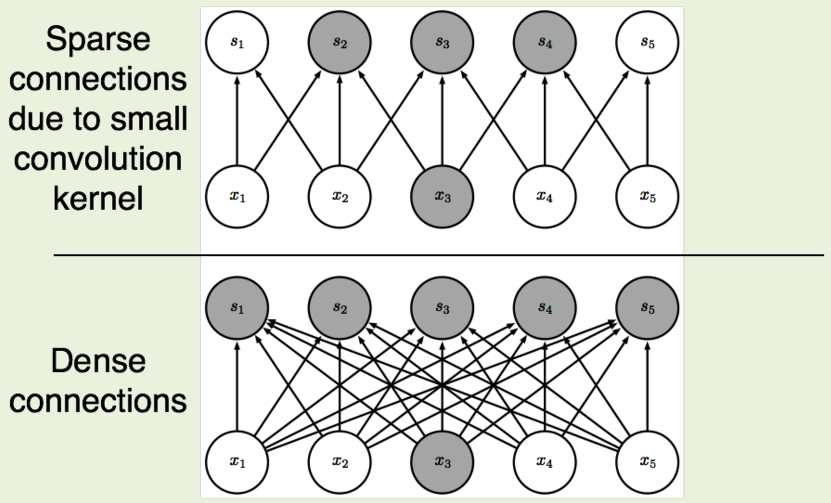
(b) The success of **convolutional neural networks** (**CNNs**) is often attributed to two characteristics: **sparse connections** and **shared weights**.

(i) Describe these two characteristics and the benefits they offer to CNNs.

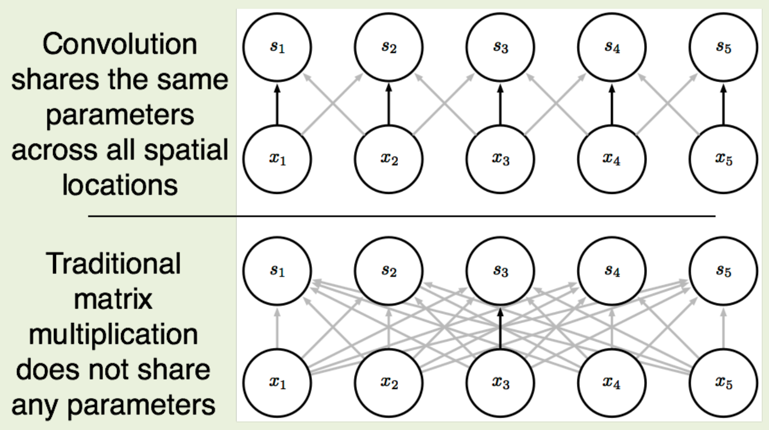
[6]

__**Sample Answer**__

The connections between layers in a convolutional neural network are much more less dense than the connections within simpler feed-forward networks (e.g. multi-layer perceptrons). The image below illustrates this. This arises from the fact that small convolutional kernels are used and so only a small number of the inputs to a network are actually connected to any hidden layer unit in the network.


Sparse connections due to small convolution kernel
s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5
Dense connections
s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5

Convolutions are similarly responsible for the fact that weights in a CNN are shared. The bottom part of the image below shows the connections between two dense layers. In this scenario the weight on each link only affects one pair of computational units. In the CNN scenario above, however, the weights in the convolutional filter are applied at multiple positions within a network and so are shared across hidden units.


Convolution shares the same parameters across all spatial locations
s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5
Traditional matrix multiplication does not share any parameters
s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5

(ii) The use of shared weights can make convolutional neural network models for image recognition **translation invariant**. Explain what this means.

[5]

<u>Sample Answer</u>

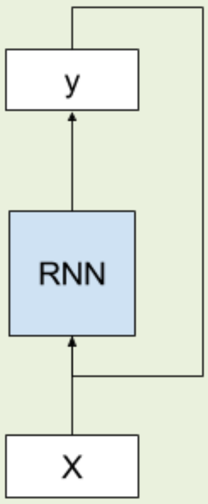
Because the same convolutional filters (with shared weights) are applied at every point in an image (or other input data structure) any pattern that the filter has been trained to recognise can be recognised at any point within the data structure. In this way CNN models are found to be translation invariant - features will be recognised even if they are moved to different part so an input data structure than those in which they were seen in the training data.

(c) If we have a **recurrent neural network** (**RNN**), we can view it as a different type of network by "*unrolling it through time*". Explain what this means.

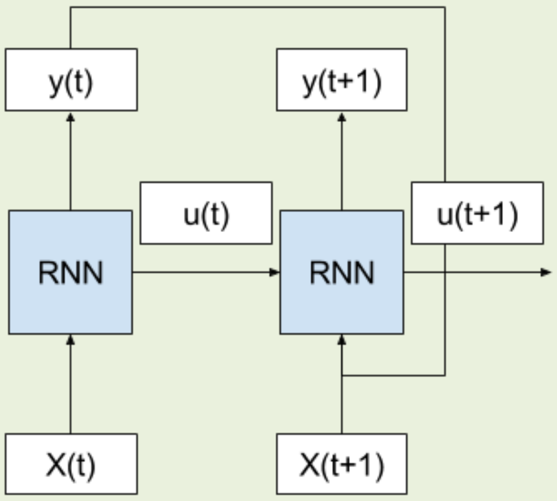
[6]

<u>Sample Answer</u>

Recurrent neural networks are networks that allow cyclical connections in their graphs so that previous outputs can affect the current output. the image below shows such a network.



Unrolling such a graph refers to the process of generating an equivalent directed a-cyclical graph the explicitly represents the recurrence over a finite time horizon. An example is shown in the image below.



Unrolling RNNs is useful both for conceptually understanding what a network does, but also they are often actually implemented in this way.

4.	**(a)**	Why is the **reward** only an indirect measure of the agent's performance in **reinforcement learning**? (Provide at least two reasons.)
		[6]
		<u>**Sample Answer**</u> 1. The reward is only an indirect measure as it is only a measure of the current environmental state. This state may have been affected by factors other than the agent (e.g. other agents, or random events). 2. The temporal credit assignment problem. Even if the rewards are interpreted by the agent as an evaluation of it's actions only, there were probably a whole sequence of actions leading to a result. Which actions in the sequence should be given credit or blame? 3. The rewards do not say which other actions would have been better (if any).
	(b)	Explain the difference between the (**State-Action-Reward-State-Action**) **SARSA** and **Q-learning** algorithms for reinforcement learning.
		[8]
		<u>**Sample Answer**</u> In SARSA, an agent starts in state 1, performs action 1, and gets a reward (reward 1). Now, it's in state 2 and performs another action (action 2) and gets the reward from this state (reward 2) before it goes back and updates the value of action 1 performed in state 1. In contrast, in Q-learning the agent starts in state 1, performs action 1 and gets a reward (reward 1), and then looks and sees what the maximum possible reward for an action is in state 2, and uses that to update the action value of performing action 1 in state 1. So the difference is in the way the future reward is found. In Q-learning it's simply the highest possible action that can be taken from state 2, and in SARSA it's the value of the *actual* action that was taken. This means that SARSA takes into account the control policy by which the agent is moving, and incorporates that into its update of action values, whereas Q-learning simply assumes that an optimal policy is being followed. We can write the Q-learning update policy as Q(s, a) = reward(s) + alpha * max(Q(s')) and the SARSA update policy as Q(s, a) = reward(s) + alpha * Q(s', a'). This difference is often summarised by saying that Q-learning is *off-policy*, while SARSA is *on-policy*.
	(c)	You are tasked with building an automated player of an endless runner video game (e.g. Temple Run, FlappyBird) with the following properties:

			• no a priori knowledge of the world • the character is constantly moving forward at a fixed speed • the character is aware only of incoming elements of the game (enemies or obstacles) that are within a certain distance • possible actions are jumping (to avoid low enemies or obstacles) or ducking (to avoid high enemies or obstacles) • touching an enemy or obstacle leads to a restart of the game • the goal is to maximise the distance covered by the character in a single run
		(i)	Which **reinforcement learning** method you would use to build this system? Give reasons for your answer.
			[8]
			Reasonable approaches for this scenario would include SARSA, Q-learning but any well argued online approach would be suitable. Students should discuss issues including whether the scenario is online or off-line, whether it is discrete or continuous, whether an on-policy or off-policy approach might work best, …
		(ii)	Describe what constitutes a **state**, an **action** and whatever **other parameters** are relevant to the chosen method for this scenario (there is no need to describe the actual algorithm in detail).
			[3]
			Sample Answer State: The position of the agent in the world and the relative positions of any obstacles or enemies. Action: There are only two actions - duck or jump.

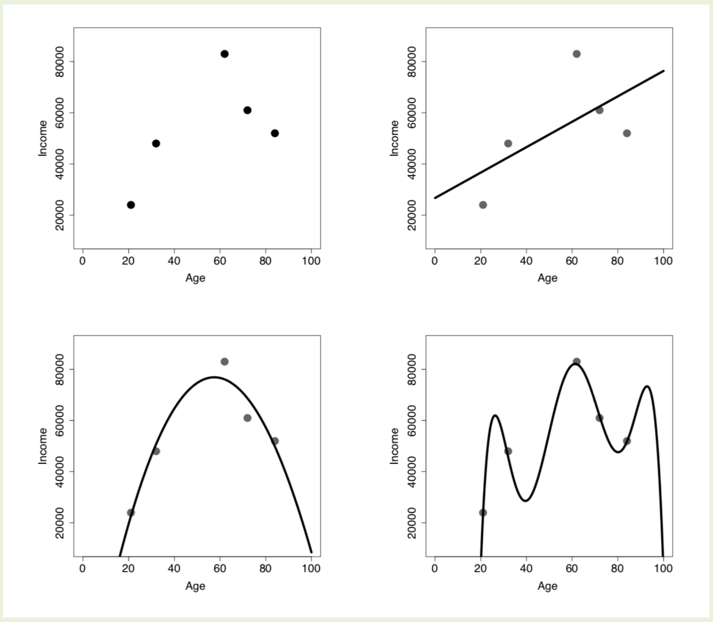
5. **(a)** Inductive machine learning is often referred to as an **ill-posed problem**. Explain why this is the case and discuss the implications that follow from it. In your answer be sure to refer to examples of specific inductive machine learning algorithms.

[8]

<u>Sample Answer</u>

1) What is meant by an ill-posed problem and what are the implications of this for machine learning.

- *Inductive machine learning algorithms essentially search through a hypothesis space to find the best hypothesis that is consistent with the training data used. It is possible to find multiple hypotheses that are consistent with a given training set (i.e. agrees with all training examples). It is for this reason that inductive machine learning is referred to as an ill-posed problem as there is typically not enough information in the training data used to build a model to choose a single best hypothesis. Inductive machine learning algorithms must somehow choose one of the available hypotheses as the **best**. An example like that shown in the figure below would be useful at this point*



2) How do machine learning algorithms deal with the fact that machine learning is ill posed.

- *Because inductive learning is ill-posed, we have to make some extra assumptions to have a unique solution with the data we have. The set of assumptions we make to have learning possible is called the **inductive bias** of the learning algorithm - this is the main implication of inductive machine learning being ill-posed.*

3) Define what is meant by inductive bias:

- *The inductive bias of a learning algorithm:*
    1. *is a set of assumption about what the true function we are trying to model looks like.*

2. *defines the set of hypotheses that a learning algorithm considers when it is learning.*
3. *guides the learning algorithm to prefer one hypothesis (i.e. the hypothesis that best fits with the assumptions) over the others.*
4. *is a necessary prerequisite for learning to happen because inductive learning is an ill posed problem.*

- *An example of the specific inductive bias introduced by particular machine learning algorithms would be good here. E.g.:*
  - *Maximum margin: when drawing a boundary between two classes, attempt to maximize the width of the boundary. This is the bias used in Support Vector Machines. The assumption is that distinct classes tend to be separated by wide boundaries.*
  - *Minimum cross-validation error: when trying to choose among hypotheses, select the hypothesis with the lowest cross-validation error.*

4) The importance and difficulty of selecting the right inductive bias:

- *As learning is not possible without inductive bias the question becomes* ***how to choose the right bias?*** *This, however, is important and difficult because:*
  - *If the inductive bias of the learning algorithm constrains the search to only consider simple hypotheses we may have excluded the real function from the hypothesis space. In other words, the true function is* ***unrealizable*** *in the chosen hypothesis space, (i.e., we are* ***underfitting****).*
  - *If the inductive bias of the learning algorithm allows the search to consider complex hypotheses, the model may hone in on irrelevant factors in the training set. In other words the model with* ***overfit*** *the training data.*

(b) The table below shows the results of a benchmark experiment to compare the performance of a number of variants of a new learning algorithm, *YALA*, against each other and two baseline methods (random forests and multi-layer perceptrons). The performance of these algorithms has been measured across five different classification datasets using *10-fold cross validation*. Performance is measured in all cases using *macro-averaged f1-score*.

	YALA-1	YALA-2	YALA-3	Random Forest	MLP
abalone	0.462	0.437	0.436	0.451	0.448
arcene	0.804	0.799	0.749	0.742	0.802
dorothea	0.697	0.541	0.692	0.676	0.659
ecoli	0.449	0.437	0.436	0.447	0.451
iris	0.944	0.943	0.911	0.851	0.951

(i) Explain why using a *macro-averaged f1-score* is more appropriate than a *micro-average f1 score* for this experiment.

[4]

Sample Answer

Macro-averaged f1-score is more appropriate than a *micro-average f1 score* for this experiment as it is very likely that some of the datasets have imbalanced class distribution. A micro average score would over emphasise the performance of the majority class while it is more useful to perform macro averaging

(ii) Convert the results table provided to a ranks table, including a row for average ranks.

[3]

Sample Answer

The table below shows the conversion into ranks and the average ranks row.

	YALA-1	YALA-2	YALA-3	Random Forest	MLP
abalone	1	4	5	2	3
arcene	1	3	4	5	2
dorothea	1	5	2	3	4
ecoli	2	4	5	3	1
iris	2	3	4	5	1
avg. rank	1.4	3.8	4	3.6	2.2

(iii) To further investigate the differences between the different algorithms statistical significance testing based on the ranks table is recommended. Describe an appropriate set of tests to perform. (You do not actually need to perform any tests).

[4]

There is no single answer to this so anything that is well thought out will score well.

Sample Answer

We recommend a Friedman test to first test whether a significant difference between the performance of the algorithms over the datasets exists. If a difference does exist then a pairwise Nemenyi test should be performed. This will show between which algorithm pairs the significant differences exist.It is possible to generate a critical differences plot based on the result of the Nemnyi test which can also be informative.

(c) The new EU General Data Protection Regulation (GDPR) comes into force this year on May 25th. Prof. Pedro Domingez, author of The Master Algorithm, recently stated that:

> "*Starting May 25, the European Union will require algorithms to explain their output,* ***making deep learning illegal***."

		Discuss this claim.
		[6]
		This is an open ended question. To score well students should discuss the following points: • The GDPR refers to automated processing which will include machine learning • The GDPR refers to "*automated processing, including profilng, which produces legal effects concerning him or her or similarly significantly effects him or her*". This does not cover all applications of machine learning but only those that have a legal or significant effect on a data subject. For example image classification for improving search (a large application area for deep learning) is unlikely to fall under this definition. • Many refer to a "*right to explanation*" within the GDPR but there is some debate over how much that is present. • If a right to explanation is required sophisticated machine learning models like deep neural networks will certainly be affected.