University College Dublin
An Coláiste Ollscoile, Baile Átha Cliath

SEMESTER II EXAMINATIONS

ACADEMIC YEAR 2018/2019

COMP 47590

Advanced Machine Learning

Assoc. Prof. V. Dimitrova

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.

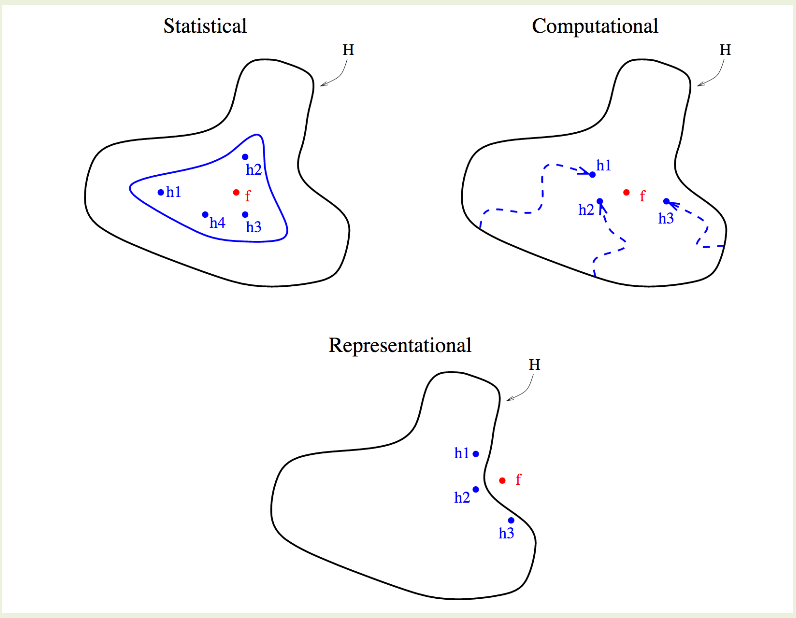
1. (a) Describe *three* different motivations for using **ensemble methods** in machine learning.

[9]

Sample Answer

Ideally students will refer to the three motivations described by Dietrich: statistical, computational, representational. 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.

Any other reasonable answer is also acceptable.

(b) 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.

[8]

<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.

(c) **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.

[8]

<u>Sample Answer</u>

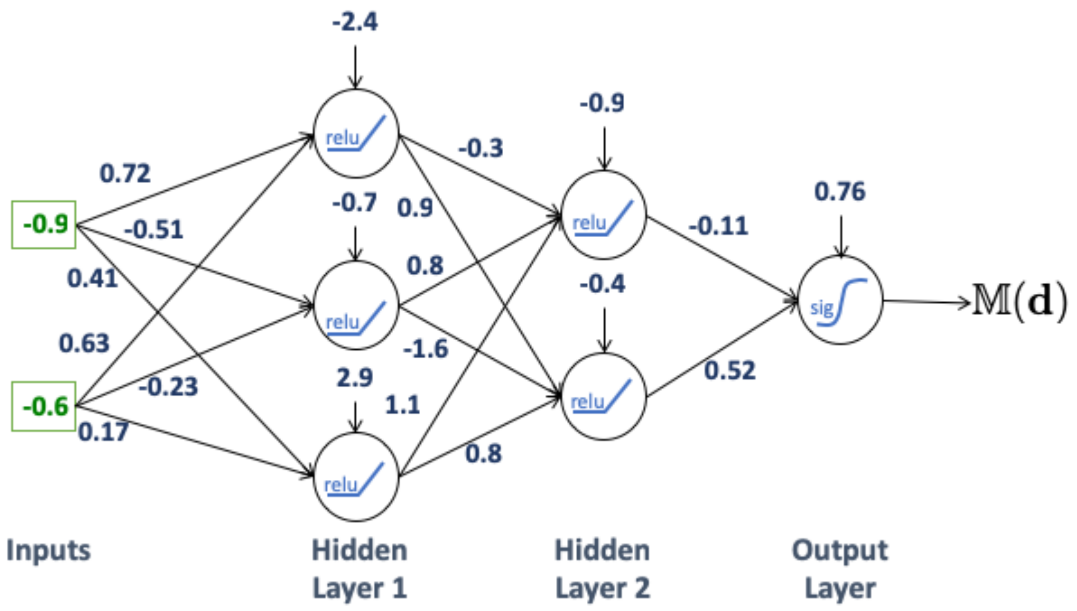
Students should explain that the gradient boosting approaching trains models sequentially like other boosting algorithms. However, rather than training each model to predict the target feature in the original training dataset, the algorithm trains each model to predict the errors made by the previous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

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

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	$\mathbb{M}_3 = \mathbb{M}_2 + \mathbb{M}_{iter3}$
	$\ldots$
n	$\mathbb{M}_n = \mathbb{M}_{n-1} + \mathbb{M}_{iterN}$

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 (-0.9, -0.6). Show your workings.

[12]

2 Setup Input, Weight and Bias Matrices

The network inputs:

$$\mathbf{d} = \begin{bmatrix} -0.9 \\ -0.6 \end{bmatrix}$$

Weights and biases for Layer 1:

$$\mathbf{W}^{[1]} = \begin{bmatrix} 0.72 & 0.63 \\ -0.51 & -0.23 \\ 0.41 & 0.17 \end{bmatrix}$$

$$\mathbf{b}^{[1]} = \begin{bmatrix} -2.4 \\ -0.7 \\ 2.9 \end{bmatrix}$$

Weights and biases for Layer 2:

$$\mathbf{W}^{[2]} = \begin{bmatrix} -0.3 & 0.8 & 1.1 \\ 0.9 & -1.6 & 0.8 \end{bmatrix}$$

$$\mathbf{b}^{[2]} = \begin{bmatrix} 0.9 \\ -0.4 \end{bmatrix}$$

Weights and biases for Layer 3:

$$\mathbf{W}^{[3]} = \begin{bmatrix} -0.11 & 0.52 \end{bmatrix}$$

$$\mathbf{b}^{[3]} = \begin{bmatrix} 0.76 \end{bmatrix}$$

3 Forward Propagate

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

$$\begin{aligned}\mathbf{z}^{[1]} &= \mathbf{W}^{[1]}\,\mathbf{d} + \mathbf{b}^{[1]}\\ &= \begin{bmatrix} 0.72 & 0.63\\ -0.51 & -0.23\\ 0.41 & 0.17\end{bmatrix}\begin{bmatrix}-0.9\\ -0.6\end{bmatrix} + \begin{bmatrix}-2.4\\ -0.7\\ 2.9\end{bmatrix}\\ &= \begin{bmatrix}-3.426\\ -0.103\\ 2.429\end{bmatrix}\end{aligned}$$

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

$$\begin{aligned}\mathbf{a}^{[1]} &= \mathrm{g}(\mathbf{z}^{[1]})\\ &= \mathrm{g}\left(\begin{bmatrix}-3.426\\ -0.103\\ 2.429\end{bmatrix}\right)\\ &= \begin{bmatrix}0\\ 0\\ 2\end{bmatrix}\end{aligned}$$

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

$$\begin{aligned}\mathbf{z}^{[2]} &= \mathbf{W}^{[2]}\,\mathbf{a}^{[1]} + \mathbf{b}^{[2]}\\ &= \begin{bmatrix}-0.3 & 0.8 & 1.1\\ 0.9 & -1.6 & 0.8\end{bmatrix}\begin{bmatrix}0\\ 0\\ 2\end{bmatrix} + \begin{bmatrix}0.9\\ -0.4\end{bmatrix}\\ &= \begin{bmatrix}3.1\\ 1.2\end{bmatrix}\end{aligned}$$

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

$$\begin{aligned}\mathbf{a}^{[2]} &= \mathrm{g}(\mathbf{z}^{[2]})\\ &= \mathrm{g}\left(\begin{bmatrix}3.1\\ 1.2\end{bmatrix}\right)\\ &= \begin{bmatrix}3.1\\ 1.2\end{bmatrix}\end{aligned}$$

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

$$\begin{aligned}\mathbf{z}^{[3]} &= \mathbf{W}^{[3]}\,\mathbf{a}^{[2]} + \mathbf{b}^{[3]}\\ &= \begin{bmatrix}-0.11 & 0.52\end{bmatrix}\begin{bmatrix}3.1\\ 1.2\end{bmatrix} + \begin{bmatrix}0.76\end{bmatrix}\\ &= \begin{bmatrix}1.043\end{bmatrix}\end{aligned}$$

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

$$\begin{aligned}\mathbf{a}^{[3]} &= \mathrm{g}(\mathbf{z}^{[3]})\\ &= \mathrm{g}\left(\begin{bmatrix}1.043\end{bmatrix}\right)\\ &= \begin{bmatrix}0.739\end{bmatrix}\end{aligned}$$

(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

4 Calculate Loss

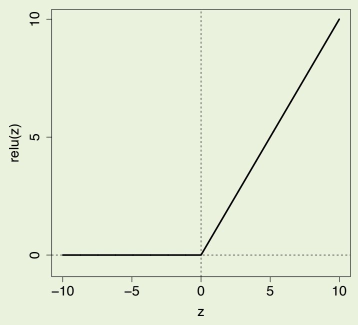
Calculate the loss based on the target and the model output:

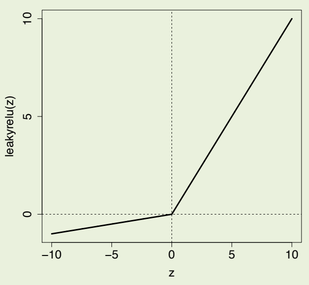
$$\begin{aligned}\text{loss} &= -\left(t \times log\left(\mathbf{a}^{[3]}\right) + (1-t) \times log\left(1 - \mathbf{a}^{[3]}\right)\right)\\ &= -\left(1 \times log\left(\begin{bmatrix}0.739\end{bmatrix}\right) + (1-1) \times log\left(1 - \begin{bmatrix}0.739\end{bmatrix}\right)\right)\\ &= 0.3018777652791483\end{aligned}$$

	(b)	Some practitioners favour replacing units using a **rectified linear** (**relu**) activation function with units using a **leaky relu** activation function. Describe each of these activation functions (include appropriate diagrams), and explain the potential advantages of using l**eaky relu**.
		[4]

Sample Answer

The three activation functions are described as follows.

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

leaky_relu:  $leakyrelu(z) = max(0.1\, z, z)$

Earlier activiation functions, e.g. sigmoid and tanh, squash their outputs into a small range, which led to signal disappearing due to repeated multiplication of the gradient associated with these functions by 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.

Relu, however, loses any gradient signal for negative values. Leaky relu fixes this by allowing some signal from negative values to still propagate through the network.

	(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.

		[6]

Sample Answer

Students should first describe the three approaches as follows:

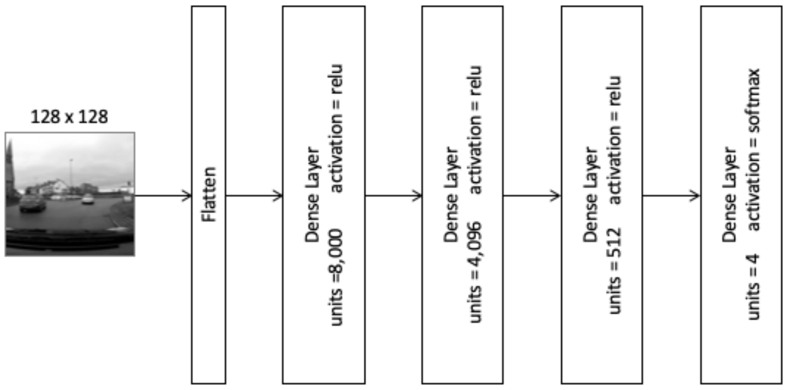
- **Stochastic gradient descent** a forward and backward pass through the network is made for each training instance and weights and biases are updated after each training instance is considered.
- **Batch gradient descent** a forward and pass through the network is made for each training instance and 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:

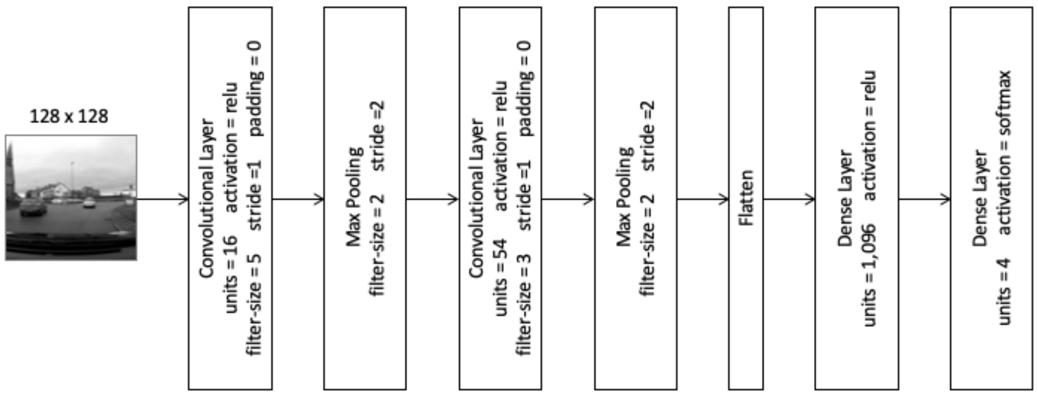
- **Stochastic gradient descent** is easy to implement and can result in fast learning, **but** is computationally expensive and can result in a noisy gradient signal
- **Batch gradient descent** is computationally efficient and can result in a stable gradient signal, **but** requires gradient accumulation, 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 128 pixel by 128 pixel greyscale 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.


128 x 128
Flatten
Dense Layer units =8,000 activation = relu
Dense Layer units = 4,096 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


128 x 128
Convolutional Layer units = 16 activation = relu filter-size = 5 stride =1 padding = 0
Max Pooling filter-size = 2 stride =2
Convolutional Layer units = 54 activation = relu filter-size = 3 stride =1 padding = 0
Max Pooling filter-size = 2 stride =2
Flatten
Dense Layer units = 1,096 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.

[12]

Sample Answer

Multi-layer perceptron:

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

Input: 128 * 128 = 16,384

Layer 1: 16,384 * 8,000 + 8000 = 131,080,000

Layer 2: 8,000 * 4,096 + 4,096 = 32,772,096

Layer 3: 4,096 * 512 + 512 = 2,097,664

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

Total parameters: 131,080,000+ 32,772,096+ 2,097,664 + 2,052 = 165,951,812

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 Dim: 128 * 128 * 1

Layer 1: 5 * 5 * 1 * 16 + 16 = 416

Layer 2 Dim: 62 * 62 * 16 = 61,504

Layer 2: 3 * 3 * 16 * 54 + 54 = 7,830

Layer 2 Dim: 30 * 30 * 54 = 48,600

Layer 3: 48,600 * 1,096 + 1,096 = 53,266,696

Layer 4: 1,096* 4 + 4 = 4,388

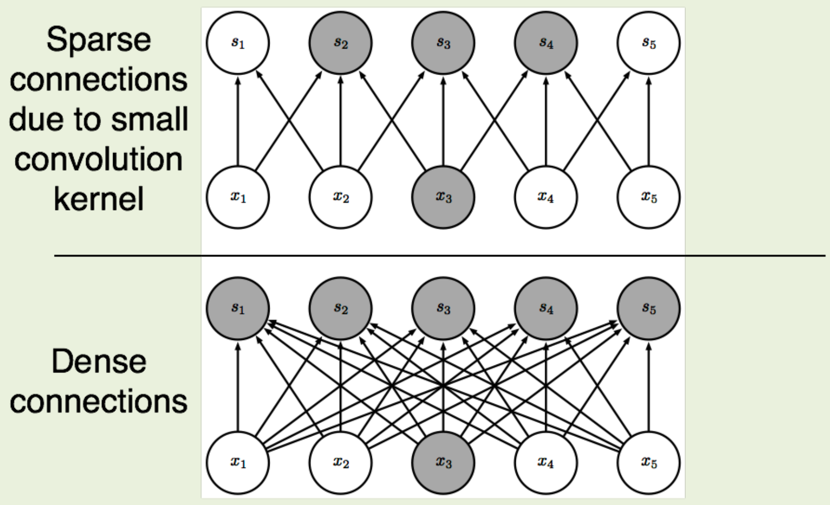
Total parameters: 53,279,330

(b) The convolutional neural network in part (a) has much fewer weights than the multi-layer perceptron network. Explain how the convolutional neural network is able to learn accurate models with so many fewer learned parameters.

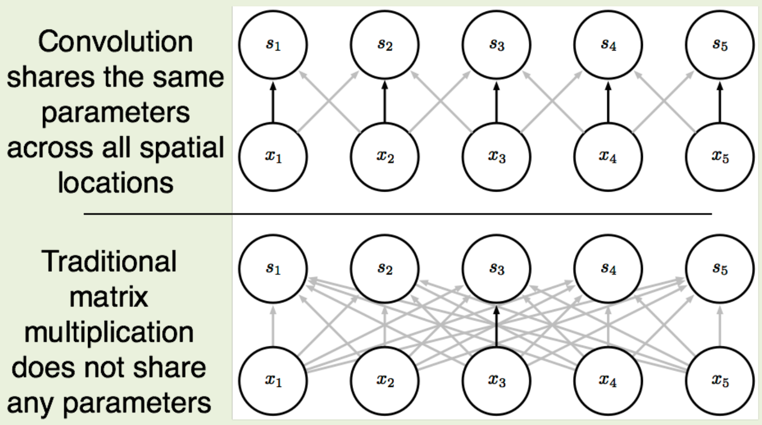
[5]

__**Sample Answer**__

The connections between layers in a convolutional neural network are much 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.

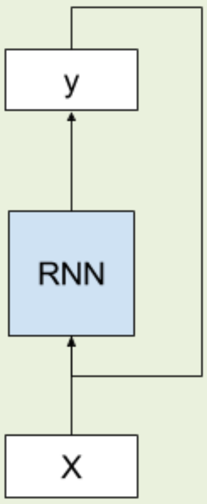

Convolution shares the same parameters across all spatial locations
s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5
Traditional matrix multiplication does not share any parameters
s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5

(c) It is often said that a **recurrent neural network** (**RNN**) can be "*unrolled through time*". Explain what this means.

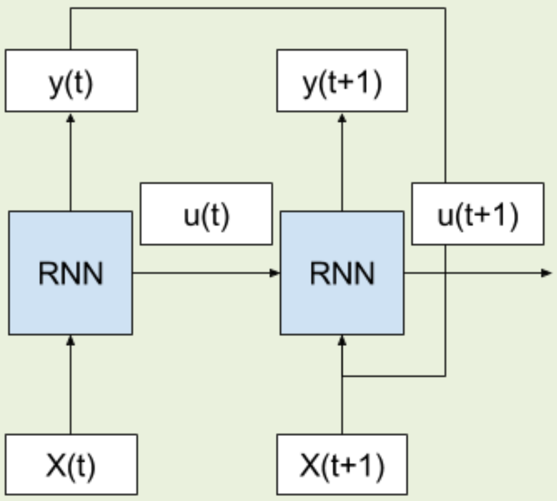
[8]

Sample Answer

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.

		 y(t)
		y(t+1)
		u(t)
		u(t+1)
		RNN
		RNN
		X(t)
		X(t+1) Unrolling RNNs is useful both for conceptually understanding what a network does, but also they are often actually implemented in this way.

4.	**(a)**	Describe the *four* key components of a **reinforcement learning** system.
		[8]

Sample Answer

Beyond the agent and the environment there are four main sub-elements of a reinforcement learning system:

- a policy
- a reward signal
- a value function
- a model of the environment (optional)

A policy defines the learning agent's way of behaving at a given time.

- A policy is a mapping from perceived states of the environment to actions to be taken when in those states
- Corresponds to what psychology would call a set of *stimulus–response rules*
- Policy implementations can range from a simple lookup table to extensive computation such as a search process or a deep learning model
- The policy is the core of a reinforcement learning agent

A reward signal defines the goal in a reinforcement learning problem

- On each time step, the environment sends the agent a single number called the reward
    - Perhaps analogous to the experiences of pleasure or pain
- The agent's sole objective is to maximize the total reward it receives over the long run
- The reward signal is the primary basis for altering the policy
    - if an action selected by the policy is followed by low reward, then the policy may be changed to select some other action in that situation in the future

Whereas the reward signal indicates what is good in an immediate sense, a value function specifies what is good in the long run

- The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state
- A state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards (or the reverse could be true)

- To make a human analogy eating a cake might have high reward (because it tastes nice) but low value (because it will make me fat)

Some reinforcement learning systems include a model of the environment

- The model should mimic the behaviour of the environment to allow an agent to predict what is likely to happen when they take an action.
- We can distinguish RL techniques as being *model-based* or *model-free* depending on whether or not they include this component

(b) The differences between the **SARSA** (**State-Action-Reward-State-Action**) and **Q-learning** algorithms for reinforcement learning are often summarised by saying that SARSA is *on-policy* while Q-learning is *off-policy*. Explain what this means.

[7]

Sample Answer

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

$$Q(s, a) = \text{reward}(s) + \text{alpha} * \max(Q(s'))$$

and the SARSA update policy as

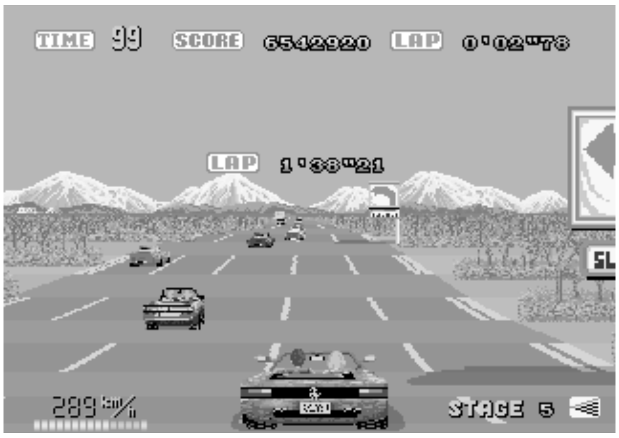
$$Q(s, a) = \text{reward}(s) + \text{alpha} * Q(s', a').$$

(c) OutRunLite is a simplified version of OutRun, the popular 1980s arcade game. In OutRunLite:

- the car is constantly moving forward at a fixed speed
- the player is aware only of obstacles (e.g. other cars) within a certain distance
- possible actions are moving left, moving right, or going straight
- crashing into an obstacle (e.g. another car) leads to the end of the game

- the goal of the game is to maximise the distance covered by the car in a single run

Describe how you could use reinforcement learning to build an automated OutRunLite player. In your answer describe how you would model **states**, **actions**, **rewards**, and any other important elements of the solution**.**



[10]

__**Sample Answer**__

State: There are lots of ways in which this could be modelled. A simple grid would make sense in which the player has a position and other cars occupy cells. It could be a screen grab of the game. It could be a vector of number desicrbing the current game scenario.

Action: There are three actions: left, right, straight.

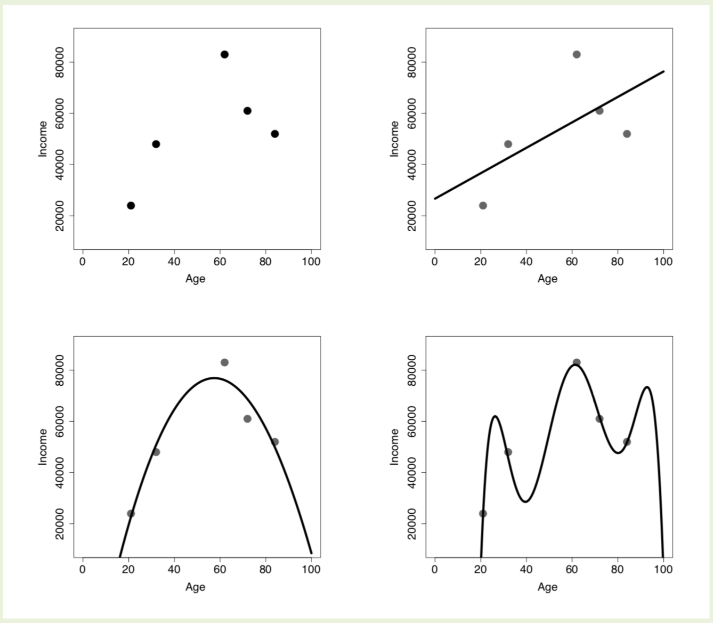
Reward: How far the player has driven

5.	(a)	Machine learning algorithms face a constant struggle between **over-fitting** and **under-fitting**. Explain what this means.
		[8]

Sample Answer

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 *micro-average accuracy*.

	YALA-1	YALA-2	YALA-3	Random Forest	MLP
MNIST	0.492	0.291	0.431	0.595	0.279
CiFAR-10	0.826	0.809	0.557	0.851	0.765
Flowers	0.728	0.456	0.635	0.467	0.733
Animals	0.296	0.641	0.522	0.495	0.362
Cars	0.941	0.701	1.000	0.976	0.993

(i) Describe concerns that you would have with the use of micro-average accuracy in this experiment.

[4]

Sample Answer

Micro averaged performance metrics are dangerous in this type of benchmark experimtn 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) Based on these performance scores determine which algorithm is performing *best* in this experiment.

[5]

Sample Answer

A good approach in this kind of experiment is to calculate average ranks. The method with the lowest average rank is performing *best*. The table below shows the conversion into ranks and the average ranks row.

	YALA-1	YALA-2	YALA-3	Random Forest	MLP
MNIST	2	4	3	1	5
CiFAR-10	2	3	5	1	4
Flowers	2	5	3	4	1
Animals	5	1	3	2	4
Cars	4	5	1	3	2
avg. rank	3	3.6	3	2.2	3.2

The random forest is out-performing the other methods in this case.

(c) The new EU General Data Protection Regulation (GDPR) came into effect May 25th 2018. In the run up to this Prof. Pedro Domingez, author of The Master Algorithm, stated that:

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

Discuss this claim.

[8]

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 profiling, 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.