University College Dublin
An Coláiste Ollscoile, Baile Átha Cliath

SPRING TRIMESTER EXAMINATIONS

ACADEMIC YEAR 2019/2020

COMP 47590

Advanced Machine Learning

Prof. Steve Counsell

Assoc. Prof. Chris Bleakley

Assoc. Prof. Brian Mac Namee*

Time Allowed: 2 Hours

Instructions for Candidates

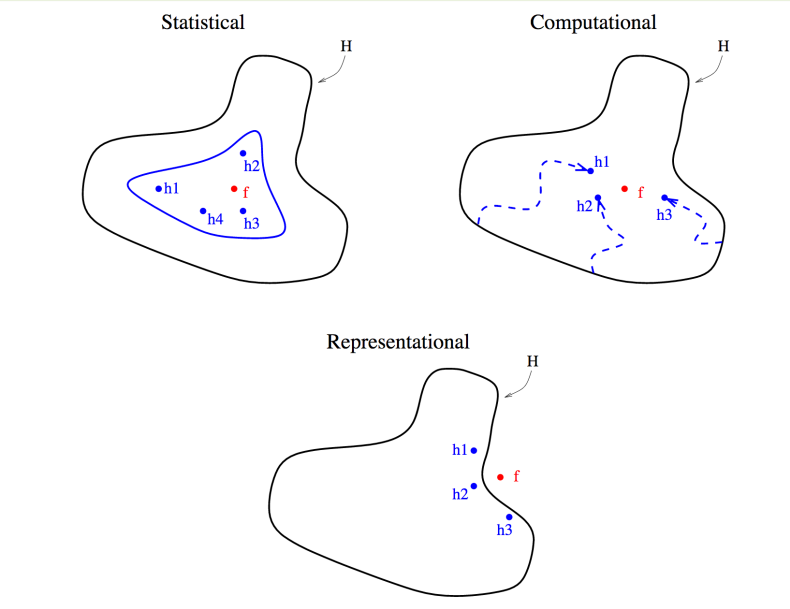
Answer any **four** out of five questions. All questions carry equal marks. Total marks available **100**. The value of each part of each question is shown in brackets next to it.

Student Number

Seat Number

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)**	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.
		[9 marks]
		Sample Answer Reproduction of the diagram below from (Deittrich, 2000) would be useful.  The statistical motivation 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 closer 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.

	(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 marks]
		Sample Answer 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)	The table below shows a simple dataset that describes the average temperature on a given day and the number of bicycles rented by a rental shop on that day. This dataset is being used to train a gradient boosting model to predict the number of rentals from the daily temperature. The column labelled $\mathbb{M}_0(\mathbf{d})$ shows the output of the first model added to the gradient boosting model for each instance in the training dataset.

ID	Temperature	Rentals	$\mathbb{M}_0(\mathbf{d})$
1	4	602	1,287.1
2	5	750	1,287.1
3	7	913	1,287.1
4	12	1229	1,287.1
5	18	1827	1,287.1
6	23	2246	1,287.1
7	27	2127	1,287.1
8	28	1714	1,287.1
9	32	838	1,287.1
10	35	625	1,287.1

		Calculate the target values that the next base model added to the gradient boosting ensemble will be trained to predict.
		[8 marks]
		Sample Answer This question explores students' understanding of what gradient boosting models are trained to predict. Rather than training each model to predict the target feature in

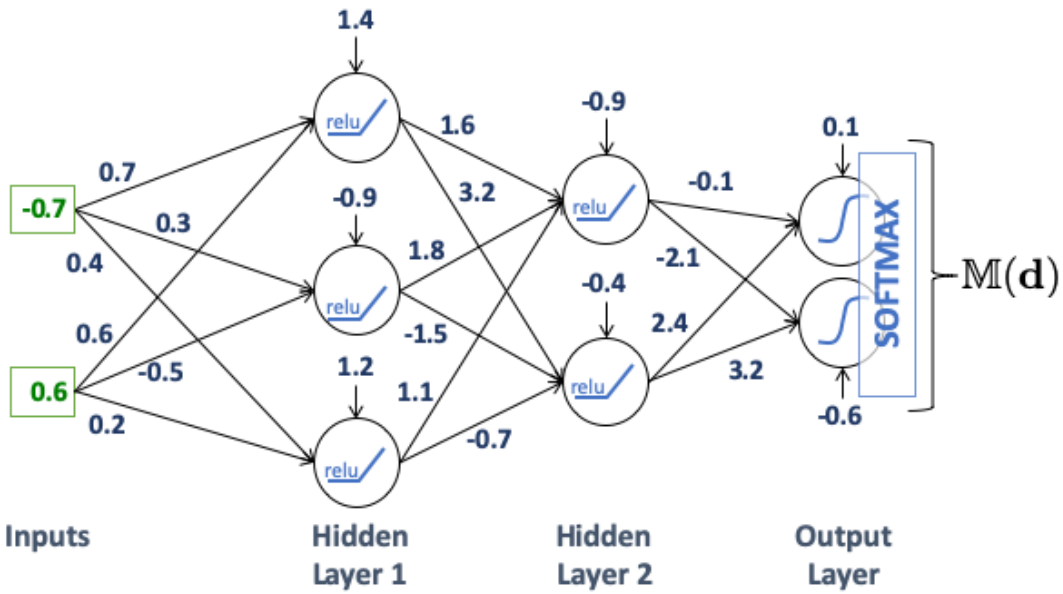
the original training dataset, the algorithm trains each model to predict the errors made by the previous model:

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

So, next model will be trained to predict the different between the target values and the outputs of the first model added to the ensemble.

ID	Temperature	Rentals	$\mathbb{M}_0(\mathbf{d})$	t - $\mathbb{M}_0(\mathbf{d})$
1	4	602	1,287.1	-685.1
2	5	750	1,287.1	-537.1
3	7	913	1,287.1	-374.1
4	12	1229	1,287.1	-58.1
5	18	1827	1,287.1	539.9
6	23	2246	1,287.1	958.9
7	27	2127	1,287.1	839.9
8	28	1714	1,287.1	426.9
9	32	838	1,287.1	-449.1
10	35	625	1,287.1	-662.1

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 *softmax* 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 [-0.7, 0.6]. Show your workings.

[14 marks]

2 Setup Input, Weight and Bias Matrices

The network inputs:

$$\mathbf{d} = \begin{bmatrix} -0.7 \\ 0.6 \end{bmatrix}$$

Weights and biases for Layer 1:

$$\mathbf{W}^{[1]} = \begin{bmatrix} 0.7 & 0.6 \\ 0.3 & -0.5 \\ 0.4 & 0.2 \end{bmatrix}$$

$$\mathbf{b}^{[1]} = \begin{bmatrix} 1.4 \\ -0.9 \\ 1.2 \end{bmatrix}$$

Weights and biases for Layer 2:

$$\mathbf{W}^{[2]} = \begin{bmatrix} 1.6 & 1.8 & 1.1 \\ 3.2 & -1.5 & -0.7 \end{bmatrix}$$

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

Weights and biases for Layer 3:

$$\mathbf{W}^{[3]} = \begin{bmatrix} -0.1 & 2.4 \\ -2.1 & 3.2 \end{bmatrix}$$
$$\mathbf{b}^{[3]} = \begin{bmatrix} 0.1 \\ -0.6 \end{bmatrix}$$

3 Forward Propagate

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

$$\begin{aligned} \mathbf{z}^{[1]} &= \mathbf{W}^{[1]}\,\mathbf{d} + \mathbf{b}^{[1]} \\ &= \begin{bmatrix} 0.7 & 0.6 \\ 0.3 & -0.5 \\ 0.4 & 0.2 \end{bmatrix} \begin{bmatrix} -0.7 \\ 0.6 \end{bmatrix} + \begin{bmatrix} 1.4 \\ -0.9 \\ 1.2 \end{bmatrix} \\ &= \begin{bmatrix} 1.27 \\ -1.41 \\ 1.04 \end{bmatrix} \end{aligned}$$

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

$$\begin{aligned} \mathbf{a}^{[1]} &= \mathrm{g}(\mathbf{z}^{[1]}) \\ &= \mathrm{g}\left(\begin{bmatrix} 1.27 \\ -1.41 \\ 1.04 \end{bmatrix} \right) \\ &= \begin{bmatrix} 1.27 \\ 0.0 \\ 1.04 \end{bmatrix} \end{aligned}$$

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

$$\begin{aligned} \mathbf{z}^{[2]} &= \mathbf{W}^{[2]}\,\mathbf{a}^{[1]} + \mathbf{b}^{[2]} \\ &= \begin{bmatrix} 1.6 & 1.8 & 1.1 \\ 3.2 & -1.5 & -0.7 \end{bmatrix} \begin{bmatrix} 1.27 \\ 0.0 \\ 1.04 \end{bmatrix} + \begin{bmatrix} -0.9 \\ -0.4 \end{bmatrix} \\ &= \begin{bmatrix} 2.276 \\ 2.936 \end{bmatrix} \end{aligned}$$

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

$$\begin{aligned}\mathbf{a}^{[2]} &= g(\mathbf{z}^{[2]}) \\ &= g\left(\begin{bmatrix}2.276\\2.936\end{bmatrix}\right) \\ &= \begin{bmatrix}2.276\\2.936\end{bmatrix}\end{aligned}$$

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

$$\begin{aligned}\mathbf{z}^{[3]} &= \mathbf{W}^{[3]}\,\mathbf{a}^{[2]} + \mathbf{b}^{[3]} \\ &= \begin{bmatrix}-0.1 & 2.4\\-2.1 & 3.2\end{bmatrix}\begin{bmatrix}2.276\\2.936\end{bmatrix} + \begin{bmatrix}0.1\\-0.6\end{bmatrix} \\ &= \begin{bmatrix}6.919\\4.016\end{bmatrix}\end{aligned}$$

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

$$\begin{aligned}\mathbf{a}^{[3]} &= g(\mathbf{z}^{[3]}) \\ &= g\left(\begin{bmatrix}6.919\\4.016\end{bmatrix}\right) \\ &= \begin{bmatrix}0.948\\0.052\end{bmatrix}\end{aligned}$$

(ii) If the target feature vector for the current input vector is [1.0, 0.0], calculate the **loss** associated with this training instance using **cross entropy loss**.

[3 marks]

Calculate cross entropy loss

```
Loss = -(1*log(0.9480) + 0*log(0.0520))
     = 0.05340
```

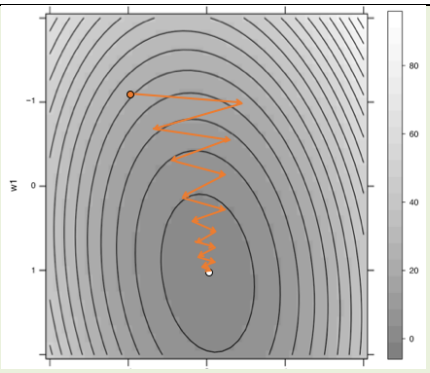
(b) **Gradient descent with momentum**, **RMSprop**, and **adam** are three common adaptations to the basic gradient descent algorithm used to train neural networks. Explain how these approaches improve upon basic gradient descent and how they relate to each other.

[8 marks]

Sample Answer

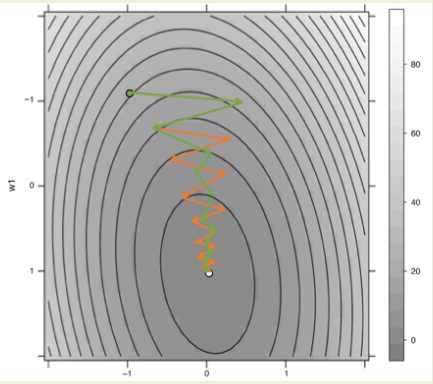
(Diagrams such as those included in this answer would be useful.)

The gradient descent algorithm optimizes network weight values during a journey across an error (or loss) surface.



All of the adaptations listed improve this process by taking a more direct route across the error surface. This is achieved by adding different types of momentum terms.

Gradient descent with momentum adds a momentum term to the gradient descent process to avoid sudden changes in exploration of the error surface. A diagram such as the following would help.



This is achieved by using an exponentially weighted moving average applied to the gradient terms over subsequent iterations.

$$v_{d\mathbf{W}} = \beta v_{d\mathbf{W}} + (1-\beta)\, d\mathbf{W}$$
$$v_{d\mathbf{b}} = \beta v_{d\mathbf{b}} + (1-\beta)\, d\mathbf{b}$$

Weight and bias terms are then updated with these averaged gradients rather than raw gradient values.

$$\mathbf{W} = \mathbf{W} - \alpha v_{d\mathbf{W}}$$
$$\mathbf{b} = \mathbf{b} - \alpha v_{d\mathbf{b}}$$

RMSprop builds on top of the same idea as gradient descent with momentum but uses the square of the gradients in the exponentially weighted moving average.

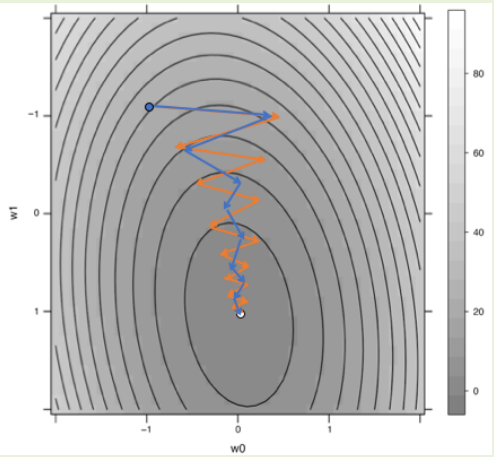
$$v_{d\mathbf{W}} = \beta\, v_{d\mathbf{W}} + (1-\beta)\, d\mathbf{W}^{*2}$$
$$v_{d\mathbf{b}} = \beta v_{d\mathbf{b}} + (1-\beta)\, d\mathbf{b}^{*2}$$

These values are then used in the weight and bias update rules.

$$\mathbf{W} = \mathbf{W} - \alpha \frac{d\mathbf{W}}{\sqrt{v_{d\mathbf{W}}} + \epsilon}$$

$$\mathbf{b} = \mathbf{b} - \alpha \frac{d\mathbf{b}}{\sqrt{v_{d\mathbf{b}}} + \epsilon}$$

This gives a more direct route across the error surface.



Adam mixes the gradient descent with momentum and RMSprop approaches in a weighted sum:

$$v_{d\mathbf{W}} = \beta_1 v_{d\mathbf{W}} + (1 - \beta_1)\, d\mathbf{W}$$

$$v_{d\mathbf{b}} = \beta_1 v_{d\mathbf{b}} + (1 - \beta_1)\, d\mathbf{b}$$

Gradient Descent with Momentum

$$s_{d\mathbf{W}} = \beta_2\, s_{d\mathbf{W}} + (1 - \beta_2)\, d\mathbf{W}^{*2}$$

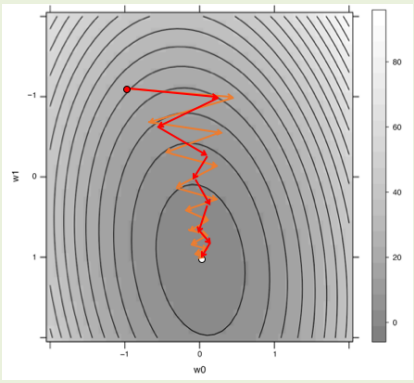
$$s_{d\mathbf{b}} = \beta_2 s_{d\mathbf{b}} + (1 - \beta_2)\, d\mathbf{b}^{*2}$$

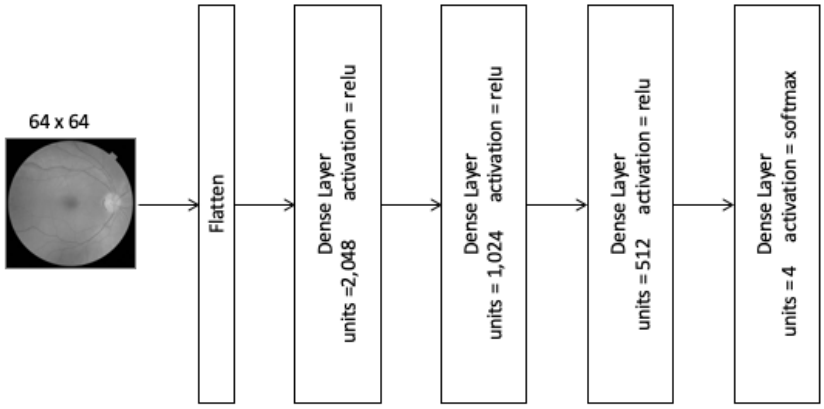
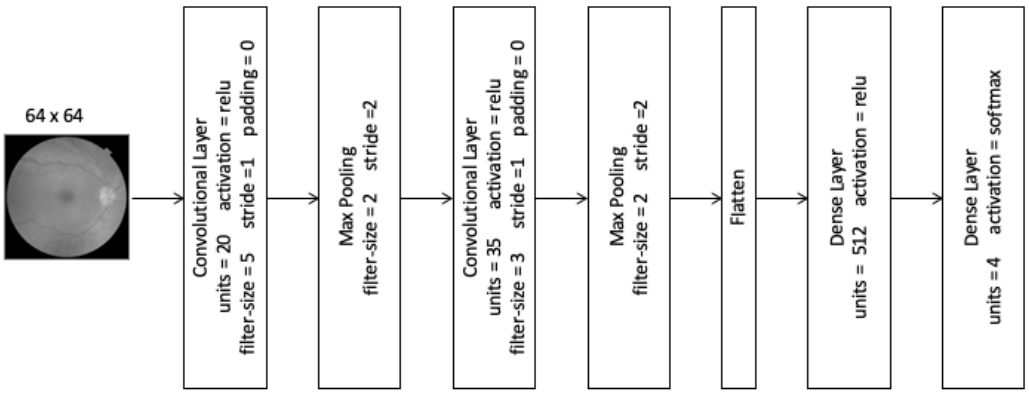
RMSprop

$$\mathbf{W} = \mathbf{W} - \alpha \frac{v_{d\mathbf{W}}}{\sqrt{s_{d\mathbf{W}}} + \epsilon}$$

$$\mathbf{b} = \mathbf{b} - \alpha \frac{v_{d\mathbf{b}}}{\sqrt{s_{d\mathbf{b}}} + \epsilon}$$

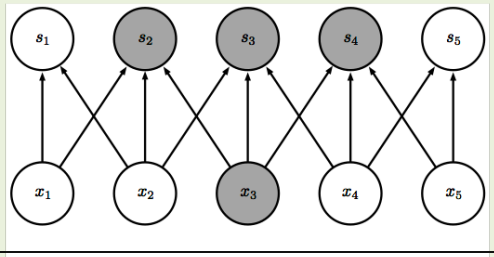
This gives a very efficient route across an error surface.



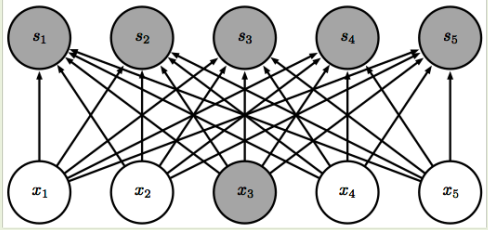
3.	**(a)**	You have been tasked with training a neural network to diagnose diabetes from images of a person's retina. The model should produce diagnoses on a four-point scale: *grade-0*= no diabetes, *grade-1*= early stage diabetes, *grade-2*= diabetes, *grade-3*= advanced diabetes. The only input to the model is a 64 pixel by 64 pixel greyscale image of the retina of a person's right eye. Image (a) shows the architecture of a multi-layer perceptron neural network designed for this problem. Image (b) shows the architecture of a convolutional neural network designed for this problem. Both architectures are composed of four layers.  (a) A multi-layer perceptron network for diabetes diagnosis  (b) A convolutional neural network for diabetes diagnosis Calculate the number of parameters (weights and biases) that need to be learned for each network architecture.
		[12 marks]
		<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: 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 Dim: 64 * 64 * 1 Layer 1: 5 * 5 * 1 * 20 + 20 = 520 Layer 1 Output Dim: 60 * 60 * 20 = 72,000 Layer 1 Output Dim After Pooling: 30 * 30 * 20 = 18,000 Layer 2: 3 * 3 * 20 * 35 + 35 = 6,335 Layer 2 Output Dim: 28 * 28 * 35 = 27,440 Layer 2 Output Dim After Pooling: 14 * 14 * 35 = 6,860 Layer 3: 6,860 * 512 + 1,096 = 3,512,832 Layer 4: 512 * 4 + 4 = 2,052 Total parameters: 3,521,739
	(b)	The ability of convolutional networks to learn accurate models with many fewer weights than multi-layer perceptrons of similar depth is often attributed to **shared weights** and **sparse connections**. Explain the meaning of these two terms.
		[5 marks]
		<u>Sample Answer</u> 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

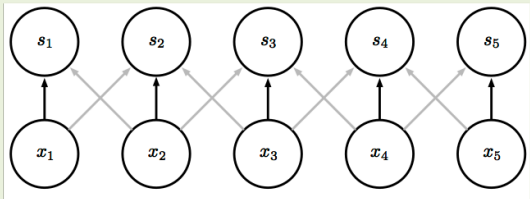


Dense connections

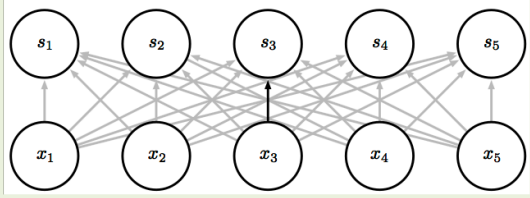


Convolutions are similarly responsible for the fact that weights in a CNN are shared. The bottom part of the image below shows the connections between two dense layers. In this scenario the weight on each link only affects one pair of computational units. In the CNN scenario above, however, the weights 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

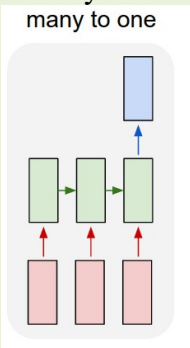


Traditional matrix multiplication does not share any parameters

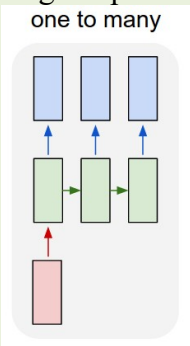


	(c)	R**ecurrent neural networks** can be built to model *many-to-one*, *one-to-many,* or *many-to-many* relationships. For each of these three types of relationships briefly describe the network architecture that could be used to model it, and give an example application for which it would be appropriate to model that type of relationship.
		[8 marks]
		Sample Answer Students should explain that recurrent models can be used as follows: • **many-to-one:** learn patterns in a sequence that predict a single output. One example application is sentiment analysis in which a sentence (the sequence)

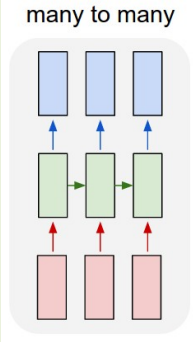
is analysed to determine a single sentiment value).



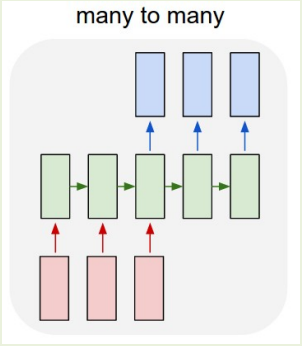
- **one-to-many:** one-to-many networks learn a model that generates a sequence from a single input. One example application is image captioning in which a single input image generates a sentence (the sequence) describing it.



- **many-to-many:** models generate an output sequence based on an input sequence (they are also often known as sequence-to-sequence models). There are two common varieties. A simple model in which each new input sequence element leads to a new output sequence element. Part of speech tagging is an example application of this type of network as a part of speech tag is generated for each word in an input sequence.



The second type is an encoder-decoder model in which an input sequence is encoded by the model before being decoded to generate an output sequence.

		Machine translation models commonly use this type of architecture. 

4.	**(a)**	Describe the concept of **discounted return** that is frequently used in reinforcement learning.
		[3 marks]
		Sample Answer Calculating the expected return from a sequence of actions is key in developing reinforcement learning systems. In a basic formulation of expected return that simply sums rewards, e.g. $$G = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_e$$ expected future rewards are considered to be as valuable as the immediate reward that the agent will receive from taking the next immediate action. Just like we might be more excited about receiving a gift of $100 today than a promise to receive a gift of $100 in a year's time, it is reasonable when calculating expected return to pay more attention to the immediate reward we expect to receive from taking the next action, than to the rewards that we expect to receive in 10 or even 100 action's time. This is known as discounted return. We can define discounted return as: $$G_\gamma = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots + \gamma^{e-t} r_e$$ where γ is a discount factor that can take a value between 0 and 1.
	(b)	An intelligent agent trained to play a video game completes an episode and receives the following sequence of rewards over six timesteps: $\{r_0 = -33, r_1 = -11, r_2 = -12, r_3 = 27, r_4 = 87, r_5 = 156\}$ Compare the discounted returns calculated at time $t = 0$ based on this reward sequence when discounting factors of 0.72 and 0.22 are used.
		[5 marks]
		Sample Answer Students should begin by calculating the discounted returns using: $$G_\gamma = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots + \gamma^{e-t} r_e$$ For discounting factor of 0.72: $$\begin{aligned} G = & (0.72^0 \times -33) + (0.72^1 \times -11) + (0.72^2 \times -12) \\ & + (0.72^3 \times 27) + (0.72^4 \times 87) + (0.72^5 \times 156) \\ = & (-33) + (0.72 \times -11) + (0.5184 \times -12) \\ & + (0.3732 \times 27) + (0.2687 \times 87) + (0.1935 \times 156) \\ = & \ 16.4985 \end{aligned}$$ For discounting factor of 0.22: $$\begin{aligned} G = & (0.22^0 \times -33) + (0.22^1 \times -11) + (0.22^2 \times -12) \\ & + (0.22^3 \times 27) + (0.22^4 \times 87) + (0.22^5 \times 156) \\ = & (-33) + (0.22 \times -11) + (0.0484 \times -12) \\ & + (0.0106 \times 27) + (0.0023 \times 87) + (0.0005 \times 156) \\ = & \ -35.4365 \end{aligned}$$ Students should then discuss that with the lower discount factor much less attention is paid to later rewards and so the overall return is much lower.

(c) To try to better understand the slightly baffling behaviour of her new baby girl, Maria - a scientifically minded new mother - monitored her baby over the course of a day recording her activity at 20 minute intervals. The activity stream looked like this (with time flowing down through the columns):

SLEEPING	SLEEPING	SLEEPING	CRYING	SLEEPING	SLEEPING
CRYING	SLEEPING	HAPPY	HAPPY	CRYING	HAPPY
SLEEPING	SLEEPING	CRYING	HAPPY	SLEEPING	HAPPY
SLEEPING	CRYING	SLEEPING	HAPPY	SLEEPING	HAPPY
SLEEPING	CRYING	SLEEPING	HAPPY	SLEEPING	HAPPY
HAPPY	SLEEPING	HAPPY	HAPPY	SLEEPING	HAPPY
HAPPY	SLEEPING	HAPPY	SLEEPING	HAPPY	HAPPY
HAPPY	HAPPY	HAPPY	SLEEPING	HAPPY	SLEEPING
SLEEPING	SLEEPING	HAPPY	SLEEPING	HAPPY	SLEEPING
SLEEPING	HAPPY	HAPPY	SLEEPING	SLEEPING	SLEEPING
SLEEPING	HAPPY	HAPPY	SLEEPING	HAPPY	SLEEPING
SLEEPING	CRYING	CRYING	SLEEPING	SLEEPING	SLEEPING

Maria noticed that her baby could occupy one of three states - HAPPY, CRYING, or SLEEPING - and moved quite freely between them.

(i) Based on the sequence of states given above calculate a transition matrix that gives the probability of moving between each of the three states.

[6 marks]

Sample Answer

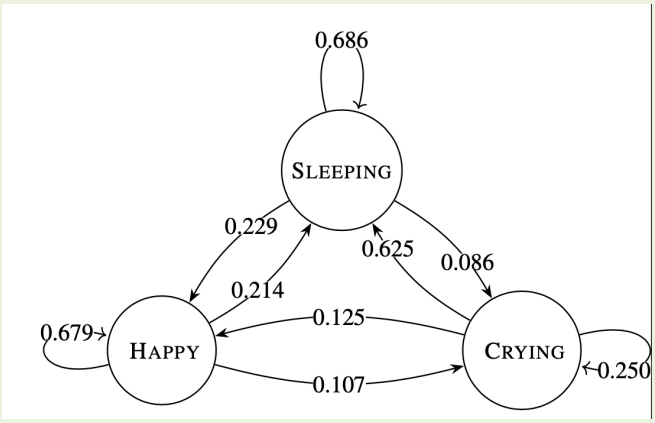
The first step to building the transition matrix is to count the frequency of each possible state transition. Working down through the list of states we can count the number of times we move from one state to the next. This gives a transition frequency table:

	SLEEPING	CRYING	HAPPY
SLEEPING	24	3	8
CRYING	5	2	1
HAPPY	6	3	19

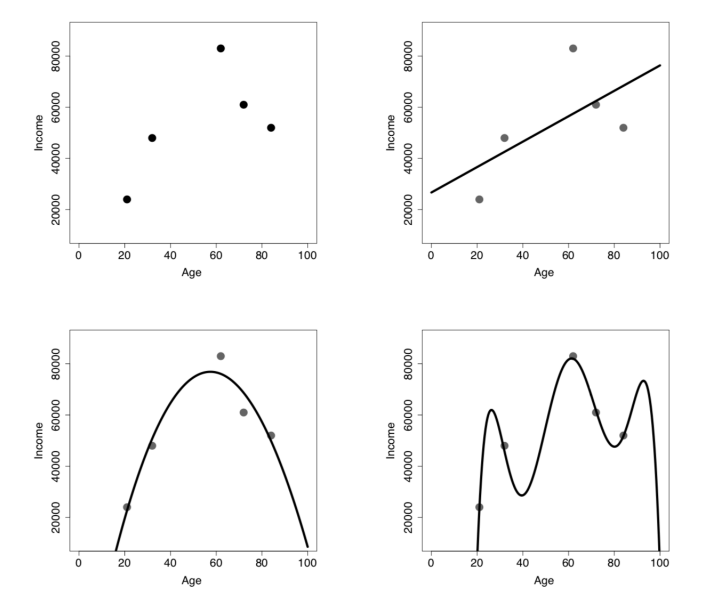
By normalising each row in the table (dividing each value by the sum of values in the row) we can calculate the final transition matrix:

	SLEEPING	CRYING	HAPPY
SLEEPING	0.686	0.086	0.229
CRYING	0.625	0.250	0.125
HAPPY	0.214	0.107	0.679

(ii) Draw a Markov process diagram to capture the behaviour of a small baby as described above.

			[3 marks]
			Sample Answer A diagram like this one is appropriate for this answer. 
	(c)	TempleRunLite is a simplified version of Temple Run, the popular endless runner game. In TempleRunLite: • the player is constantly moving forward at a fixed speed • the player is aware only of obstacles (e.g. fallen columns) within a certain distance of its position • possible actions are moving left, moving right, or going straight • crashing into an obstacle (e.g. a fallen column) leads to the end of the game • the player can collect coins as they run • the goal of the game is to maximise the number of coins collected by the player in a single run  Describe how you could use reinforcement learning to build an automated TempleRunLite player. In your answer describe how you would model **states**, **actions**, **rewards**, and any other important elements of the solution**.**	
			[8 marks]
			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 obstacles occupy cells. It could be a screen grab of the game. It could be a vector of number describing the current game scenario.

		Action: There are three actions: left, right, straight. Reward: How many coins the player has collected would be the most obvious approach, but others would be interesting too.

5.	(a)	Machine learning algorithms face a constant struggle between **over-fitting** and **under-fitting**. Explain what this means.
		[8 marks]
		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.:* o *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.* o *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:* o *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**).* o *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)	When developing a machine learning model that will be deployed to perform a task for a user, we can describe three different goals of evaluation: 1. to determine which model is the most suitable for a task 2. to estimate how the model will perform after deployment 3. to convince users that the model will meet their needs Describe the differences between these goals, and how the evaluation methods used to achieve each of them can be different.
		[8 marks]
		<u>Sample Answer</u> *Students should outline that when performing an evaluation to prepare a model for deployment the first goal is to determine which approach might work best. Decision s to be made based on this evaluation include which modelling approach will be used, which data pre-processing techniques might be used, and the optimal algorithm hyper-parameters to be used. This evaluation is very much driven by the machine learning practitioner. Typically, k-fold cross validation can be used as the main evaluation method for this kind of evaluation.*

		Once all of the decisions about what modelling approach to take have been made the next goal attempts to evaluate the likely performance of a model after deployment. This is largely concerned with estimating generalisation error of the model. The only sensible way to evaluate this is to use a hold out test set that has been involved at all in training the model. This is the only reasonable way to evaluate generalisation error. *Once practitioners are convinced that a modelling approach will achieve acceptable performance after deployment the last step is to convince the people for whom the model is being built that it will meet their needs. This can be done with the same kinds of experiments used to evaluate likely generalisation error, but often different performance measures need to be used so as to speak to a different audience (not machine learning practitioners). It is also important to consider a deployment experiment at this stage too.*
	(c)	The EU General Data Protection Regulation (GDPR) came into effect on May 25th, 2018. What effect does this regulation have on the use of machine learning?
		[8 marks]
		<u>Sample Answer</u> 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.