CPU9444_summary_yann

Exercise
- Ex 1
- Ex 2 (Different with 9414)
- Ex 3
- Ex 4
- Ex 5
- Ex 6
- Ex 7
- Ex 8
- Ex 9
- Ex 10
- Ex 11
- Ex 12
- Ex 13
- Ex 14
- Ex 15
- Ex 16
- Ex 17
  - Entropy
  - 纯度（Purity）
  - 条件熵（Conditional Entropy）
  - 定义：
- Ex 18
- Ex 19
  - Log softmax
- Ex 20
- Ex 21
- Ex 22
- Ex 23
- Ex 24
- Ex 25
- Ex 26
- Ex 27
- Ex 28
- Ex 29

Ex 30
Ex 31
Ex 32
Ex 33
Ex 34
Ex 35
Ex 36
Ex 37
Ex 38
Ex 39
Ex 40
Ex 41
Ex 42
Ex 43
Actor-Critic 方法
Ex 44
Ex 45
Denoising AutoEncoder
Ex 46
Ex 47
Ex 48
Ex 49
Ex 50

Exercise

Ex 1

a. Construct by hand a Perceptron which correctly classifies the following data; use your knowledge of plane geometry to choose appropriate values for the weights w_0, w_1 and w_2.

Training Example	x_1	x_2	Class
a.	0	1	-1
b.	2	0	-1
c.	1	1	+1

Ex 2 (Different with 9414)

b. Demonstrate the Perceptron Learning Algorithm on the above data, using a learning rate of 1.0 and initial weight values of

$w_0 = -1.5$
$w_1 = 0$
$w_2 = 2$

Iteration	w_0	w_1	w_2	Training Example	x_1	x_2	Class	$s=w_0+w_1x_1+w_2x_2$	Action
1	-1.5	0	2	a.	0	1	-	+0.5	Subtract
2	-2.5	0	1	b.	2	0	-	-2.5	None
3	-2.5	0	1	c.	1	1	+	-1.5	Add
4	-1.5	1	2	a.	0	1	-	+0.5	Subtract
5	-2.5	1	1	b.	2	0	-	-0.5	None
6	-2.5	1	1	c.	1	1	+	-0.5	Add
7	-1.5	2	2	a.	0	1	-	+0.5	Subtract
8	-2.5	2	1	b.	2	0	-	+1.5	Subtract
9	-3.5	0	1	c.	1	1	+	-2.5	Add
10	-2.5	1	2	a.	0	1	-	-0.5	None
11	-2.5	1	2	b.	2	0	-	-0.5	None
12	-2.5	1	2	c.	1	1	+	+0.5	None

Ex 3

(2 marks) Briefly explain the difference between Perceptron learning and backpropagation.

Ex 4

(2 marks) Briefly explain the difference between batch learning and online learning.

Ex 5

(3 marks) What would happen if the transfer functions at the hidden layers in a multi-layer perceptron would be omitted; i.e. if the activation would simply be the weighted sum of the activations at the previous layer? Explain why this (simpler) activation scheme is not normally used in MLPs although it would simplify and accelerate the calculations for the backpropagation algorithm?

Ex 6

(3 marks) Sketch the following activation functions, and write their formulas:

(i) sigmoid
(ii) tanh
(iii) ReLU

Ex 7

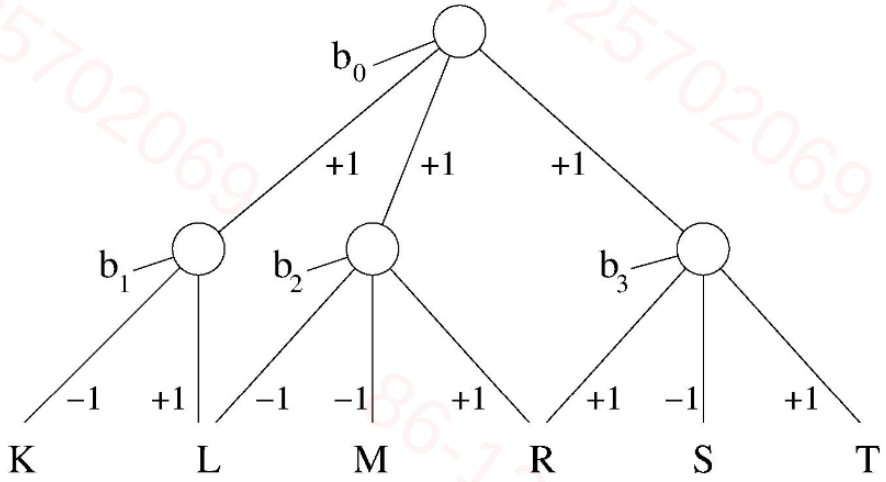
Construct by hand a Neural Network (or Multi-Layer Perceptron) that computes the XOR function of two inputs. Make sure the connections, weights and biases of your network are clearly visible.

Ex 8

Two-layer Neural Network to compute the function $(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$:

Ex 9

Consider the following multi-layer perceptron, using the threshold activation function, and assume that TRUE is represented by 1; FALSE by 0.



For which values of the biases b_0, b_1, b_2, b_3 would this network compute this logical function ?

$(\neg K \vee L) \wedge (\neg L \vee \neg M \vee R) \wedge (R \vee \neg S \vee T)$

i. value of b_0 :

ii. value of b_1 :

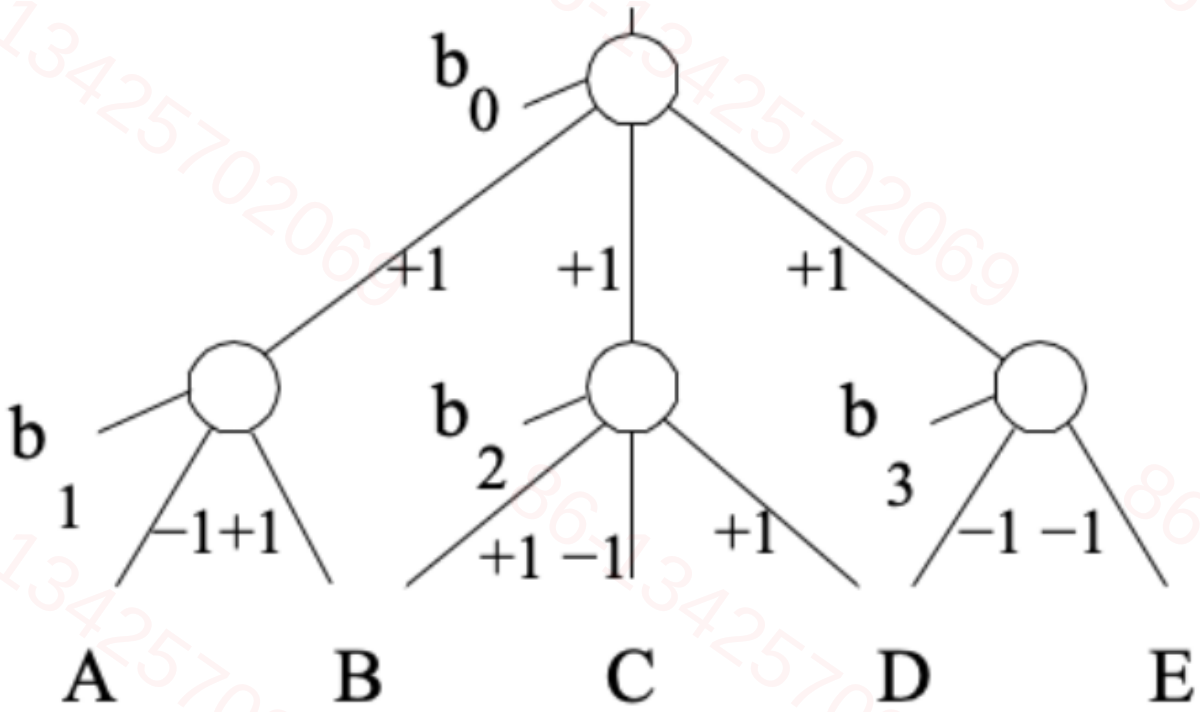
iii. value of b_2 :

iv. value of b_3 :

Ex 10

Consider the following multi-layer perceptron, with threshold activation function, and assume that TRUE is represented by 1; FALSE by 0. For which values of the biases b0, b1, b2 and b3 would this network compute the logical function?

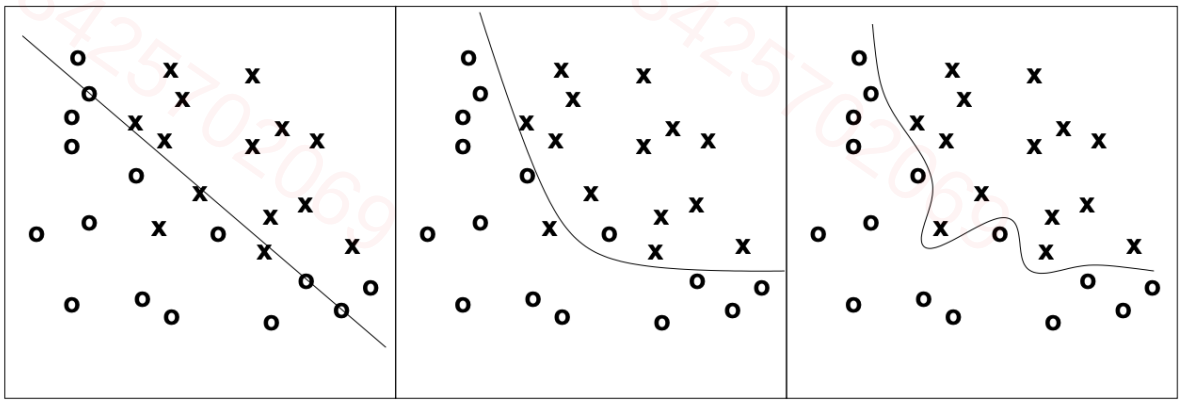
$$(\neg A \vee B) \wedge (B \vee \neg C \vee D) \wedge (\neg D \vee \neg E)$$



- b0=-2.5, b1=+0.5, b2=+0.5, b3= +1.5
- b0=-2.5, b1=-0.5, b2=-1.5, b3=+0.5
- b0=-2.5, b1=-0.5, b2=+0.5, b3= +1.5
- b0=-0.5, b1=-0.5, b2=-1.5, b3=+0.5

Ex 11

“The most likely hypothesis is the **simplest** one consistent with the data.”



inadequate | good compromise | over-fitting

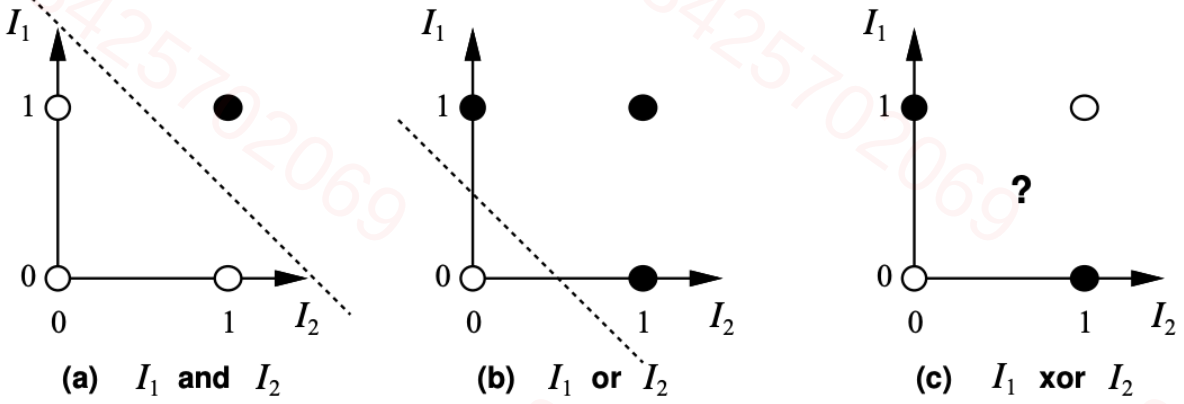
Since there can be **noise** in the measurements, in practice need to make a tradeoff between simplicity of the hypothesis and how well it fits the data.

The principle "The most likely hypothesis is the simplest one consistent with the data" is called:

- Overfitting
- Occam's Razor
- Resolution by refutation
- Entropy minimisation

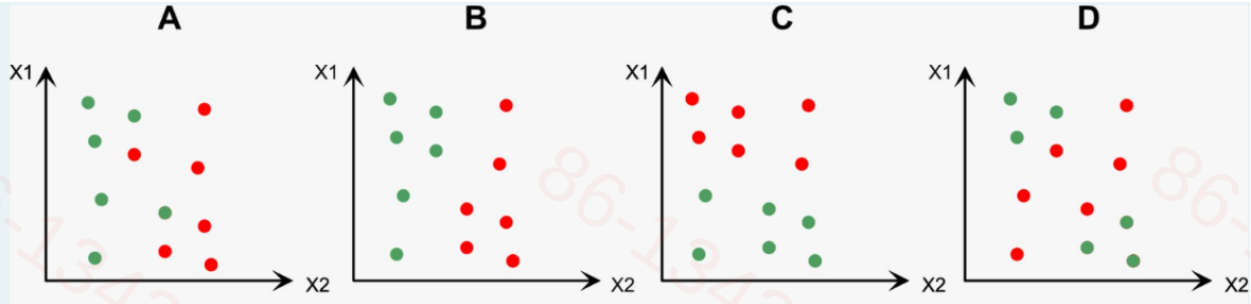
Ex 12

Problem: many useful functions are not linearly separable (e.g. XOR)



Possible solution:
x_1 XOR x_2 can be written as: (x_1 AND x_2) NOR (x_1 NOR x_2)
Recall that AND, OR and NOR can be implemented by perceptrons.

Consider the data distribution diagrams below, what case is it possible to construct a perceptron that correctly classifies all data samples that belong to either the green or red class?



- A, C
- B, C
- A
- A, B

Ex 13

Which of these statements about Dropout is FALSE:

- a. Dropout simulates an ensemble of network architectures

- b. Dropout helps to prevent overfitting
- c. Dropout encourages redundancy
- d. Dropout encourages the weight values to be small

Ex 14

When training on linearly separable data using the Perceptron Learning Rule, what will happen if both the learning rate and the initial weights are scaled up by a large factor?

- a. The data will be learned successfully, but in a larger number of epochs
- b. The data will be learned successfully, in a smaller number of epochs
- c. The data will be learned successfully, in about the same number of epochs
- d. Learning may become unstable and fail to converge

Ex 15

When using Batch Normalization, in the Testing phase, the Mean and Variance of the activations at each node are typically:

- a. pre-computed from the training set
- b. estimated using running averages
- c. either of the above
- d. none of the above

Ex 16

Which of these is NOT a method for dealing with the problem of vanishing or exploding gradients?
Select one:

- a. Batch Normalization
- b. Rectified Linear Unit
- c. Weight Initialization

- d. Conjugate Gradients

Ex 17

Entropy

熵是一个衡量集合中的不确定性或混乱程度的度量。在决策树中，给定一组样本和它们的类别，熵就用来衡量这组样本相对于其类别的同质性（纯度）。

(Shannon) entropy

- Shannon entropy of a random variable X:

$$\mathrm{H}(X) = \sum_{x \in A_x} -p(x)\log_2 p(x)$$

熵的计算公式是：

$$\text{Entropy}(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

其中
p_i 是第 i 个类别在样本集 S 中出现的概率。

所以，对于我们的这个数据集来说，计算出来的entropy为：

- For our example: weather data - 9 yes and 5 no examples:

$$H(S) = -P_{yes}\log_2 P_{yes} - P_{no}\log_2 P_{no} = I(P_{yes}, P_{no}) = I(\frac{9}{14}, \frac{5}{14}) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940\ bits$$

- The entropy is measured in bits
- When calculating entropy, we will assume that $\log_2 0 = 0$

纯度（Purity）

- **高熵 = 低纯度**：如果一个数据集的熵值很高，意味着数据集中包含多种类别的样本，纯度较低。
- **低熵 = 高纯度**：相反，如果一个数据集的熵值很低，意味着数据集大多数样本属于同一类别，纯度较高。

条件熵（Conditional Entropy）

定义：

条件熵 $H(X|Y)$ 量化了在已知随机变量 Y 的值的条件下，随机变量 X 的不确定性。其数学定义为：

$$H(X|Y) = \sum_{y \in Y} p(y) H(X|Y = y)$$

这里，$p(y)$ 是 Y 取特定值 y 的概率，而 $H(X|Y = y)$ 是 $Y = y$ 的条件下 X 的熵。

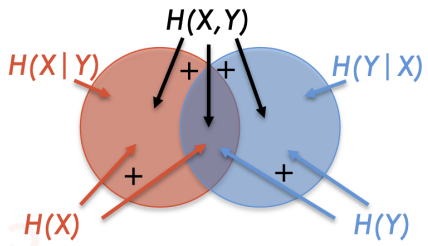
另一种更直接的表达式是：

$$H(X|Y) = -\sum_{x \in X, y \in Y} p(x, y) \log_2 \left(\frac{p(x|y)}{p(x)} \right)$$

Conditional entropy

- What if we already know something that may pertain to X? Does this change our surprise/uncertainty?
- Conditional entropy: (average) surprise remaining about sample *x* of *X* if we already know the sample *y* of *Y*

$$h(x|y) = h(x,y) - h(y)$$
$$h(x|y) = -\log_2 p(x|y)$$
$$\mathrm{H}(X|y) = \sum_{x \in A_x} p(x|y) \log_2 \frac{1}{p(x|y)}$$
$$\mathrm{H}(X|Y) = \sum_{y \in A_y} p(y) \mathrm{H}(X|y)$$
$$\mathrm{H}(X|Y) = \sum_{x \in A_x} \sum_{y \in A_y} p(x,y) \log_2 \frac{1}{p(x|y)}$$
$$\mathrm{H}(X|Y) = \mathrm{H}(X,Y) - H(Y)$$



Venn diagram

Properties:

- $0 \le H(X|Y) \le H(X)$
- $H(X|Y) = H(X)$ iff *X* and *Y* are independent
- $H(X|Y) = 0$ means there is no surprise left in *X* once we know *Y*.

Entropy and KL-Divergence (3.13)

The *entropy* of a discrete probability distribution $p = \langle p_1, \ldots, p_n \rangle$ is

$$H(p) = \sum_{i=1}^{n} p_i \left(-\log_2 p_i\right)$$

Given two probability distributions $p = \langle p_1, \ldots, p_n \rangle$ and $q = \langle q_1, \ldots, q_n \rangle$ on the same set Ω, the *Kullback-Leibler Divergence* between p and q is

$$D_{\mathrm{KL}}(p \parallel q) = \sum_{i=1}^{n} p_i \left(\log_2 p_i - \log_2 q_i\right)$$

KL-Divergence is like a "distance" from one probability distribution to another. But, it is not symmetric.

$$D_{\mathrm{KL}}(p \parallel q) \neq D_{\mathrm{KL}}(q \parallel p)$$

16 UNSW

Consider a probability distribution p on the same space $\Omega = \{A, B, C, D, E\}$ given by:

$$p = \left\langle \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16} \right\rangle$$

Compute the Entropy of p (give your answer in terms of log base 2 (bits) correct to at least two decimal places):

$$H(p) =$$

Ex 18

Given the probability distributions p and q:

$$p = \left\langle \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16} \right\rangle$$

$$q = \left\langle \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{2}, \frac{1}{16} \right\rangle$$

Compute the KL-Divergence between p and q (give your answer in terms of log base 2 (bits) correct to at least two decimal places):

$$D_{KL}(p \parallel q) =$$

Ex 19

Log softmax

Softmax (6.2.2)

- classification task with N classes
- neural network with N outputs $z_1, \ldots, z_N$
- assume the network's estimate for the probability of the correct class being j is proportional to $\exp(z_j)$
- because the probabilites must add up to 1, we need to *normalize* by dividing by their sum:

$$\begin{aligned} \text{Prob}(i) &= \frac{\exp(z_i)}{\sum_{j=1}^{N} \exp(z_j)} \\ \log \text{Prob}(i) &= z_i - \log \sum_{j=1}^{N} \exp(z_j) \end{aligned}$$

Log Softmax and Backprop

If the correct class is k, we can treat $-\log \text{Prob}(k)$ as our cost function, and the gradient is

$$\frac{d}{dz_i} \log \text{Prob}(k) = \delta_{ik} - \frac{\exp(z_i)}{\sum_{j=1}^{N} \exp(z_j)} = \delta_{ik} - \text{Prob}(i),$$

where δ_{ik} is the *Kronecker delta*.

This gradient pushes up the correct class $i = k$ in proportion to the difference between its assigned probability and 1, and it pushes down the incorrect classes $i \neq k$ in proportion to the probabilities assigned to them by the network.

Consider a neural network trained using **softmax** for a classification task with three classes 1, 2, 3. Suppose a particular input is presented, producing outputs:

$$z_1 = 2.5,\ z_2 = 1.5,\ z_3 = 0$$

Assuming the correct class for this input is Class 2, that Prob(2) is the softmax probability of the network choosing Class 2, and that $\log_e$ is the natural logarithm ($\ln$), compute the following (correct to at least two decimal places):

- $\frac{d(\log_e \mathrm{Prob}(2))}{dz_1} =$
- $\frac{d(\log_e \mathrm{Prob}(2))}{dz_2} =$
- $\frac{d(\log_e \mathrm{Prob}(2))}{dz_3} =$

Ex 20

Base rates of women having breast cancer and having no breast cancer are 0.02% and 99.98% respectively. The true positive rate or sensitivity P(positive mammography | breast cancer) = 85% and the true negative or specificity P(negative mammography | ~breast cancer) = 95%. Compute P(C | M).

Ex 21

Write the formula for Bayes' Rule, in terms of a cause A and an effect B.

Ex 22

In the context of supervised learning, explain the difference between maximum likelihood estimation and Bayesian inference.

Ex 23

Briefly explain the concept of Data Augmentation, and how it has been used in a neural network application of your choosing.

Ex 24

Consider a fully connected feedforward neural network with 6 inputs, 2 hidden units and 4 outputs, using tanh activation at the hidden units and sigmoid at the outputs. Suppose this network is trained on the following data, and that the training is successful.

Item	Inputs	Outputs

1	100000	0001
2	010000	0011
3	001000	0100
4	000100	1010
5	000010	1011
6	000001	1110

Ex 25

Consider a convolutional neural network which takes as input a 65 × 77 color image (i.e., with three channels R, G, B). The first convolutional layer has 18 filters that are 5-by-5, with stride 3 and no zero-padding.

Compute the number of:

1. Weights per filter in this layer (including bias):
2. Neurons in this layer:
3. Connections into the neurons in this layer:
4. Independent parameters in this layer:

Ex 26

The Context Layer in a Simple Recurrent Network:

- a. is computed from the current input and the previous hidden layer
- b. is comprised of the inputs in a sliding window around the current timestep
- c. is a copy of the hidden layer from the previous timestep
- d. is computed from the current input and the previous output

Ex 27

Which of these architectures would have the best chance of learning long-range dependencies?

- a. Long Short Term Memory

- b. Elman Network
- c. Simple Recurrent Network
- d. Feedforward network with sliding window

Ex 28

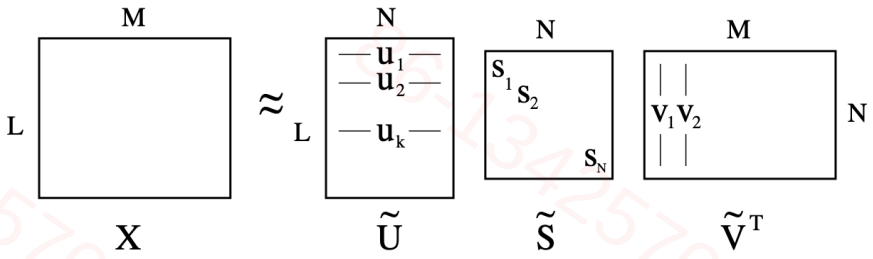
Two common methods for unsupervised pre-training of neural networks are:

- a. Deep Boltzmann Machine and Bayesian Inference
- b. Weight Initialization and Autoencoder
- c. Bayesian Inference and Weight Initialization
- d. Autoencoder and Deep Boltzmann Machine

Ex 29

Singular Value Decomposition

Co-occurrence matrix $X_{(L\times M)}$ can be decomposed as $\mathrm{X} = \mathrm{U}\,\mathrm{S}\,\mathrm{V}^{\mathrm{T}}$ where $U_{(L\times L)}$, $V_{(M\times M)}$ are unitary (all columns have unit length) and $S_{(L\times M)}$ is diagonal, with diagonal entries $s_1 \geq s_2 \geq \ldots \geq s_M \geq 0$



We can obtain an approximation for X of rank $N < M$ by truncating U to $\tilde{\mathrm{U}}_{(L\times N)}$, S to $\tilde{\mathrm{S}}_{(N\times N)}$ and V to $\tilde{\mathrm{V}}_{(N\times M)}$. The kth row of $\tilde{\mathrm{U}}$ then provides an N-dimensional vector representing the k^{th} word in the vocabulary.

20 UNSW

Considering a Singular Value Decomposition $X = USV^T$, what are the special properties of matrices U , S , and V ?

- a. U is orthogonal, V is upper triangular, and S is symmetric.

- b. U , V are upper triangular, and S is diagonal.
- c. U , V are symmetric and S is orthogonal.
- d. U , V are unitary and S is diagonal.

Ex 30

word2vec (Mikolov et al., 2013)

- **Idea**: predict rather than count
- Instead of counting how often each word *w* occurs near ”university” train a classifier on a **binary prediction task**:
  - Is *w* likely to show up near ”university”?
- We don’t actually care about this task
  - But we’ll take the learned classifier weights as the word embeddings
- Use running text as implicitly supervised training data
- No need for hand-labeled supervision

Word2Vec issues

- Word2Vec is a linear model in the sense that there is no activation function at the hidden nodes
- this 1-word prediction model can be extended to multi-word prediction in two different ways:
  - Continuous Bag of Words
  - Skip-Gram
- need a computationally efficient alternative to Softmax (Why?)
  - Hierarchical Softmax
  - Negative Sampling
- need to sample frequent words less often

Cost Function

Softmax can be used to turn these linear sums u_j into a probability distribution estimating the probability of word j occurring in the context of word k

$$\text{prob}(j|k) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})} = \frac{\exp(\mathbf{v}_j'^{\mathrm{T}}\mathbf{v}_k)}{\sum_{j'=1}^{V} \exp(\mathbf{v}_{j'}'^{\mathrm{T}}\mathbf{v}_k)}$$

We can treat the text as a sequence of numbers $w_1, w_2, \ldots, w_T$ where $w_i = j$ means that the i^{th} word in the text is the j^{th} word in the vocabulary.
We then seek to maximize the log probability

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-c \le r \le c, r \neq 0} \log\ \text{prob}(w_{t+r}|w_t)$$

where c is the size of training context (which may depend on w_t)

Which statement about word2vec is FALSE?

- a. Representations for the same word at the input and output layers are different

- b. It aims to maximise the log probability of a word, based on the surrounding words
- c. The tanh activation function is used at the hidden nodes
- d. Performance improves if frequent words are sampled less often

Ex 31

In reinforcement learning, the Q-value is:

A. The immediate reward for an action, a, in state s, plus the discounted value of following the optimal policy after that action

B. The expected value of following policy, π in state, s

C. The maximum discounted reward obtainable from state, s

D. The maximum discounted reward obtainable from state s, following the action, a

Answer: A

Ex 32

In reinforcement learning, what is the **policy**?

A. A deterministic plan to choose fixed actions for each state

B. A probability distribution over actions given states

C. A function to calculate state values

D. A function to predict future rewards

Answer: B

Ex 33

What does the **value function** represent in reinforcement learning?

A. The immediate reward for taking an action in a state

B. The expected cumulative reward for being in a state

C. The expected cumulative reward for following a policy starting from a state

D. A function to predict future states

Answer: C

Ex 34

What is the role of the **discount factor** (γ) in reinforcement learning?

A. Directly affects the size of the immediate reward

B. Controls the speed of policy updates

C. Determines the present value of future rewards

D. Reduces the number of actions available in each state

Answer: C

Ex 35

Which of the following is true about **off-policy learning**?

A. It learns the value of the policy being followed

B. It learns the value of the optimal policy irrespective of the agent's actions

C. It requires knowledge of the environment's model

D. It cannot be used with Q-learning

Answer: B

Ex 36

In reinforcement learning, the **reward signal**:

A. Tells the agent what is good in the immediate sense

B. Predicts the future rewards

C. Defines the policy

D. Determines the state transitions

Answer: A

Ex 37

The **state-action value function (Q-function)** is used to:

A. Estimate the value of a state

B. Estimate the value of an action in a state

C. Determine the next state given an action

D. Calculate the reward for a given state

Answer: B

Ex 38

What is **exploration** in reinforcement learning?

A. Using the learned policy to maximize rewards

B. Updating the policy based on observed rewards

C. Trying out new actions to discover their effects

D. Reducing the variance of the value function estimates

Answer: C

Ex 39

In the context of reinforcement learning, **SARSA** stands for:

A. State-Action-Reward-State-Action

B. Sequential-Action-Reinforcement-State-Action

C. State-Action-Reward-Sequential-Action

D. Stochastic-Action-Reinforcement-State-Algorithm

Answer: A

Ex 40

Consider a world with two states $S = \{S_1, S_2\}$ and two actions $A = \{a_1, a_2\}$, where the transitions δ and reward r for each state and action are as follows:

$$\begin{aligned} \delta(S_1, a_1) &= S_1 \quad r(S_1, a_1) = 0 \\ \delta(S_1, a_2) &= S_2 \quad r(S_1, a_2) = -1 \\ \delta(S_2, a_1) &= S_2 \quad r(S_2, a_1) = +1 \\ \delta(S_2, a_2) &= S_1 \quad r(S_2, a_2) = +5 \end{aligned}$$

i. Draw a picture of this world, using circles for the states and arrows forthe transitions.

ii. Assuming a discount factor of $\gamma = 0.9$, determine:

(a) the optimal policy $\pi^* : S \to A$

Answer: The optimal policy is:

$\pi^*(S_1) = a_2$

$\pi^*(S_2) = a_2$

(b) the state-value function $V^* : S \to R$

Remember $Q(s, a) = r(s, a) + \gamma V^*(s')$, if we follow the optimal policy then previous equation is also equal to the optimal state-value V^*

iii. Write the Q-values in a table (a.k.a. Q-table) as follows:

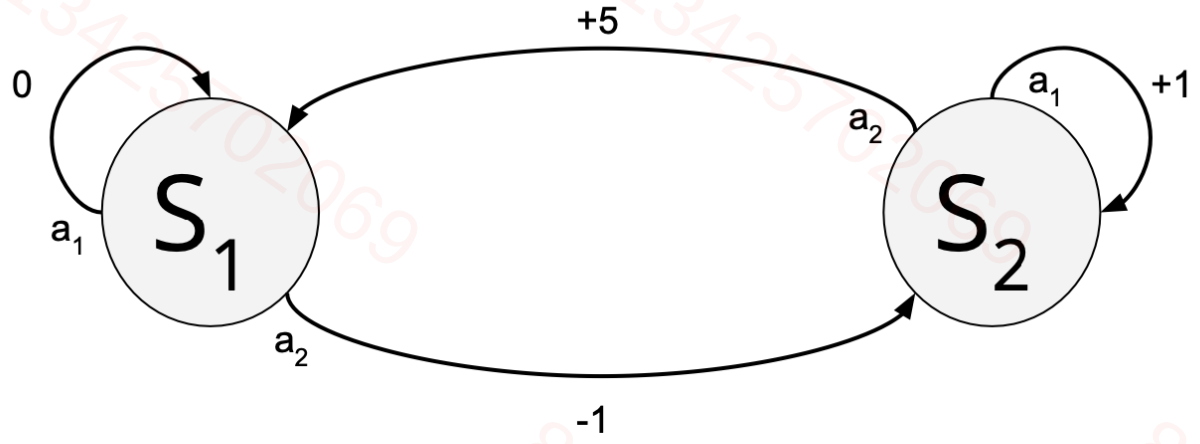


Figure 1: Draw of the world transitions.

Answer: The optimal state-value function V^* is calculated as follows:

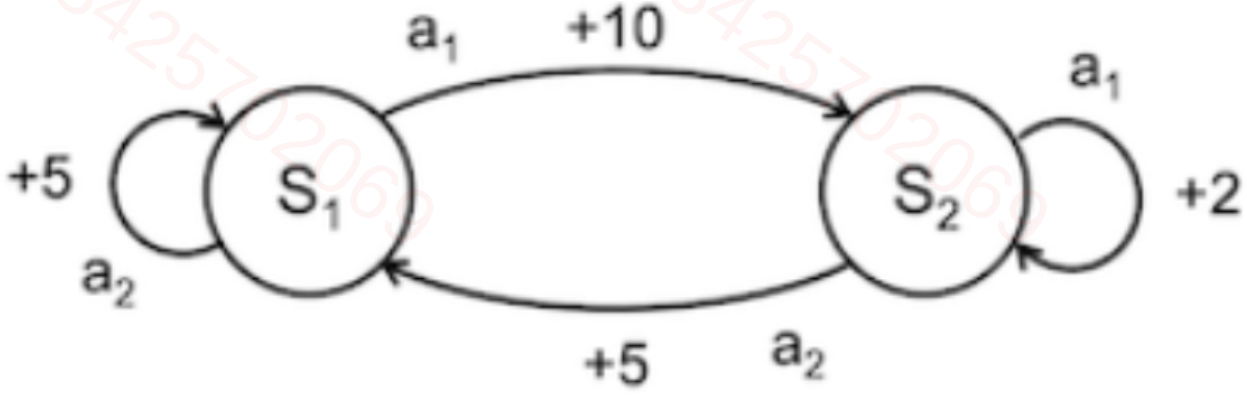
$V^*(S_1) = -1 + \gamma V^*(S_2)$
$V^*(S_2) = 5 + \gamma V^*(S_1)$
So $V^*(S_1) = -1 + 5\gamma + \gamma^2 V^*(S_1)$
$V^*(S_1) - \gamma^2 V^*(S_1) = -1 + 5\gamma$
$(1 - \gamma^2)V^*(S_1) = -1 + 5\gamma$
i.e. $V^*(S_1) = (-1 + 5\gamma)/(1 - \gamma^2) = 3.5/0.19 = 18.42$
And $V^*(S_2) = 5 + \gamma V^*(S_1)$
i.e. $V^*(S_2) = 5 + 0.9 * 3.5/0.19 = 21.58$

Answer: As in the previous question $Q(s,a) = r(s,a) + \gamma V^*(s')$. So we only need to complete it for the other state-action pairs. The action-value function for the optimal policy is calculated as follows:

$Q(S_1, a_1) = 0 + \gamma V^*(S_1) = 16.58$
$Q(S_1, a_2) = V^*(S_1) = 18.42$
$Q(S_2, a_1) = 1 + \gamma V^*(S_2) = 20.42$
$Q(S_2, a_2) = V^*(S_2) = 21.58$

Q	a_1	a_2
S_1	16.58	18.42
S_2	20.42	21.58

Ex 41



What is the relationship of value function V* for S1 and S2 under decay of 0.8?

a. $V^*(S_1) = 5 + 0.8 * V^*(S_1)$

b. $V^*(S_1) = 10 + 0.8 * (2 + 0.8 * V^*(S_2))$

c. $V^*(S_1) = 10 + 0.8 * V^*(S_2)$

d. $V^*(S_1) = 5 + V^*(S_1)$

What is the optimal policy pi* under decay of 0.8?

a. $\pi^*(S_1) = a_1, \pi^*(S_2) = a_2$

b. $\pi^*(S_1) = a_1, \pi^*(S_2) = a_1$

c. $\pi^*(S_1) = a_2, \pi^*(S_2) = a_1$

d. $\pi^*(S_1) = a_2, \pi^*(S_2) = a_2$

What is the value of value function V*?

a. $V^*(S_1) = 76.32, V^*(S_2) = 73.68$

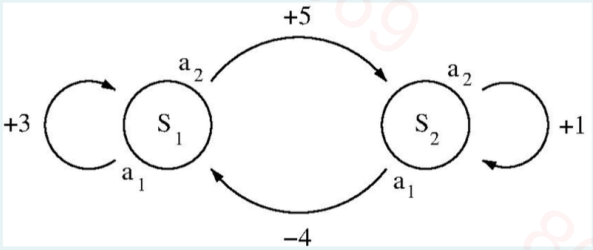
b. $V^*(S_1) = 50, V^*(S_2) = 50$

c. $V^*(S_1) = 38.89, V^*(S_2) = 36.11$

d. $V^*(S_1) = 25, V^*(S_2) = 25$

Ex 42

Consider an environment with two states S = {S_1, S_2} and two actions A = {a_1, a_2}, where the (deterministic) transitions δ and reward R for each state and action are as follows:



Assuming a discount factor of γ = 0.6, determine:

* $\pi^*(S_1)$= a1
* $\pi^*(S_2)$= a2

Again assuming γ = 0.6, compute these values (correct to two decimal places):

* $Q^*(S_1,a_1)$ 7.50
* $Q^*(S_1,a_2)$ 6.50
* $Q^*(S_2,a_1)$ 0.50
* $Q^*(S_2,a_2)$ 2.50

If γ is allowed to vary between 0 and 1, for which range of values of γ is this policy optimal (correct to two decimal places)?

* Minimum value of γ: 0.50
* Maximum value of γ: 0.71

Ex 43

Actor-Critic

Recall:

$$\nabla_\theta \,\text{fitness}(\pi_\theta) = \mathbf{E}_{\pi_\theta}[\, Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s)\,]$$

For non-episodic games, we cannot easily find a good estimate for $Q^{\pi_\theta}(s,a)$.

One approach is to consider a family of Q-Functions Q_w determined by parameters w (different from θ) and learn w so that Q_w approximates Q^{π_θ}, at the same time that the policy π_θ itself is also being learned.

This is known as an *Actor-Critic* approach because the policy determines the action, while the Q-Function estimates how good the current policy is, and thereby plays the role of a critic.

37 UNSW

Actor Critic Algorithm

for each trial
  sample a_0 *from* $\pi(a|s_0)$
  for each timestep t *do*
    sample reward r_t *from* $\mathcal{R}(r \mid s_t, a_t)$
    sample next state s_{t+1} *from* $\delta(s \mid s_t, a_t)$
    sample action a_{t+1} *from* $\pi(a \mid s_{t+1})$
    $\frac{dE}{dQ} = -[r_t + \gamma\, Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t)]$
    $\theta \leftarrow \theta + \eta_\theta\, Q_w(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t)$
    $w \leftarrow w - \eta_w \frac{dE}{dQ} \nabla_w\, Q_w(s_t, a_t)$
  end
end

38 UNSW

在强化学习中，Actor-Critic 方法是一种结合了策略梯度（Policy Gradient）和值函数近似（Value Function Approximation）的方法。这种方法通过两个主要组件来工作：Actor 和 Critic。

The Actor-Critic algorithm combines:

Select one:

- a. adversarially trained Generator and Discriminator networks
- b. Q-Learning and TD-Learning
- c. two different Q-Learners (one for action selection, the other for value estimation)
- d. Q-Learning and Policy Gradients

Clear my choice

特别是在处理非情节性（non-episodic）游戏或任务时的应用。在这种环境中，游戏或任务不会有明确的结束点，这使得直接计算$Q^{\pi_\theta}(s,a)$（即在策略π_θ 下状态s和行动 a的期望回报）变得困难。

Actor-Critic 方法

1. **Actor（行动者）**：由参数θ 确定的策略 π_θ，它负责根据当前状态选择行动。
2. **Critic（评论者）**：由不同的参数w 确定的Q-函数Q_w，它估计在给定的状态和行动下的回报。这个 Q-函数试图近似 Q^{π_θ}。

Ex 44

Deep Q-Learning with Experience Replay

- choose actions using current Q function (ε-greedy)
- build a database of experiences (s_t, a_t, r_t, s_{t+1})
- sample asynchronously from database and apply update, to minimize

$$[r_t + \gamma \max_b Q_w(s_{t+1}, b) - Q_w(s_t, a_t)]^2$$

- removes temporal correlations by sampling from variety of game situations in random order
- makes it easier to parallelize the algorithm on multiple GPUs

13 UNSW

Question 9

Answer saved

Marked out of 1.00

Flag question

The best way to deal with the problem of temporal correlations in Deep Q-Learning is:

Select one:

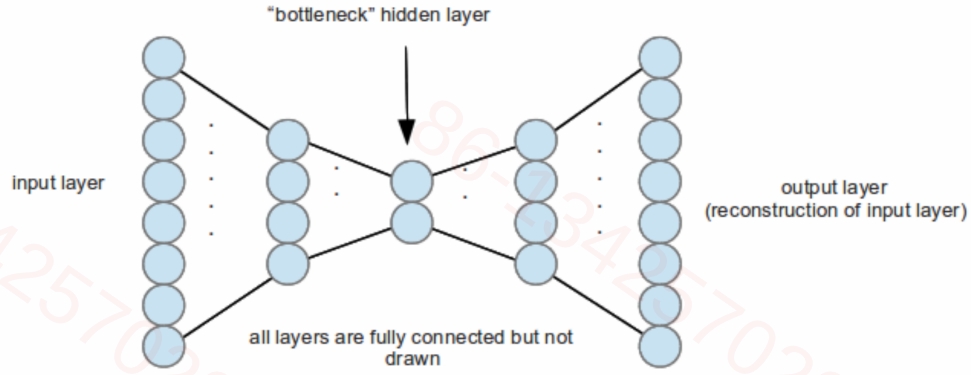
- a. Back Propagation Through Time
- b. Experience Replay
- c. Cross Entropy Minimization
- d. Increased Momentum

Clear my choice

在强化学习中，Experience Replay（经验回放）是一种重要的技术，它用于提高学习效率和稳定性。**经验回放**是一种机制，它存储了代理（agent）在环境中的互动历史，如状态转换、行动、奖励等信息，通常存储在一个称为回放缓冲区（replay buffer）的数据结构中。存储的经验（例如状态 s，行动a，奖励 r，下一个状态 s'）被用于训练过程中的样本。在训练过程中，算法会从回放缓冲区中随机采样以更新模型，而不是仅仅使用最新的经验。这意味着每个经验可能被用于训练多次。

Ex 45

Autoencoder Networks



- output is trained to reproduce the input as closely as possible
- activations normally pass through a bottleneck, so the network is forced to compress the data in some way
- Autoencoders can be used to generate "fake" items, or to automatically extract abstract features from the input

4

UNSW

Autoencoder as Pretraining

- after an autoencoder is trained, the decoder part can be removed and replaced with, for example, a classification layer
- this new network can then be trained by backpropagaiton
- the features learned by the autoencoder then serve as initial weights for the supervised learning task

Sparse Autoencoder (14.2.1)

- One way to regularize an autoencoder is to include a penalty term in the loss function, based on the hidden unit activations.
- This is analagous to the weight decay term we previously used for supervised learning.
- One popular choice is to penalize the sum of the absolute values of the activations in the hidden layer

$$E = L(x, g(f(x)) + \lambda \sum_i |h_i|$$

- This is sometimes known as L_1-regularization (because it involves the absolute value rather than the square); it can encourage some of the hidden units to go to zero, thus producing a sparse representation.

Contractive Autoencoder (14.2.3)

- Another popular penalty term is the L_2-norm of the derivatives of the hidden units with respect to the inputs

$$E = L(x, g(f(x)) + \lambda \sum_i ||\nabla_x h_i||^2$$

- This forces the model to learn hidden features that do not change much when the training inputs x are slightly altered.

Denoising AutoEncoder

Denoising Autoencoder (14.2.2)

Another regularization method, similar to contractive autoencoder, is to add noise to the inputs, but train the network to recover the original input

repeat:
  sample a training item $x^{(i)}$
  generate a corrupted version $\tilde{x}$ *of* $x^{(i)}$
  train to reduce $E = L\big(x^{(i)}, g(f(\tilde{x}))\big)$
end

Variational Autoencoder (20.10.3)

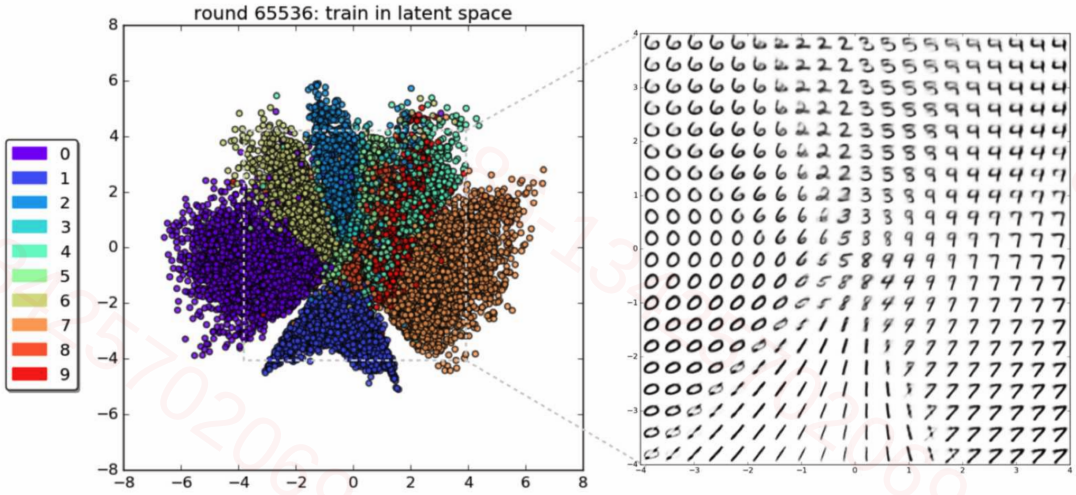
Instead of producing a single z for each $x^{(i)}$, the encoder (with parameters ϕ) can be made to produce a mean $\mu_{z|x^{(i)}}$ and standard deviation $\sigma_{z|x^{(i)}}$
This defines a conditional (Gaussian) probability distribution $q_\phi(z|x^{(i)})$
We then train the system to maximize

$$\mathbf{E}_{z\sim q_\phi(z|x^{(i)})}[\log\, p_\theta(x^{(i)}|z)] \;-\; D_{\mathrm{KL}}\big(q_\phi(z|x^{(i)})\|p(z)\big)$$

- the first term enforces that any sample z drawn from the conditional distribution $q_\phi(z|x^{(i)})$ should, when fed to the decoder, produce somthing approximating $x^{(i)}$
- the second term encourages $q_\phi(z|x^{(i)})$ to approximate $p(z)$
- in practice, the distributions $q_\phi(z|x^{(i)})$ for various $x^{(i)}$ will occupy complementary regions within the overall distribution $p(z)$

18 UNSW

Variational Autoencoder Digits



19 UNSW

What type of Autoencoder explicitly forces the hidden features not to change much when the inputs are slightly altered?

- a. Variational Autoencoder

- b. Sparse Autoencoder
- c. Denoising Autoencoder
- d. Contractive Autoencoder

Ex 46

Reinforcement Learning is when an agent is:

- a. presented multiple times (over time) with the same examples of inputs and their target outputs
- b. only presented with the inputs and not target outputs, so it aims to find structure in these inputs
- c. not presented with target outputs, but instead given a reward signal that it aims to maximize
- d. presented once with examples of inputs and their target outputs

Ex 47

Generative Adversarial Networks

Generator (Artist) G_θ and Discriminator (Critic) D_ψ are both Deep Convolutional Neural Networks.

Generator $G_\theta : z \mapsto x$, with parameters θ, generates an image x from latent variables z (sampled from a standard Normal distribution).

Discriminator $D_\psi : x \mapsto D_\psi(x) \in (0, 1)$, with parameters ψ, takes an image x and estimates the probability of the image being real.

Generator and Discriminator play a 2-player zero-sum game to compute:

$$\min_\theta \max_\psi \Big(\mathbf{E}_{x \sim p_{\text{data}}} \big[\log D_\psi(x) \big] + \mathbf{E}_{z \sim p_{\text{model}}} \big[\log \big(1 - D_\psi(G_\theta(z)) \big) \big] \Big)$$

Discriminator tries to maximize the bracketed expression,
Generator tries to minimize it.

13

For the Generative Adversarial Networks discussed in this course, the game between the Generator and Discriminator:

- a. is always zero-sum
- b. can be either zero-sum or not, but the non-zero-sum version produces better images
- c. is never zero-sum
- d. can be either zero-sum or not, but the zero-sum version produces better images

Ex 48

Oscillation and Mode Collapse

- Due to the coevolutionary dynamics, GANs can sometimes oscillate or get stuck in a mediocre stable state.
  - *oscillation*: GAN trains for a long time, generating a variety of images, but quality fails to improve.
  - *mode collapse*: Generator produces only a small subset of the desired range of images, or converges to a single image (with minor variations).
- Methods for avoiding mode collapse:
  - Conditioning Augmentation
  - Minibatch Features (Fitness Sharing)
  - Unrolled GANs

25 UNSW

Briefly describe two ways in which a GAN may fail to converge.

Ex 49

Part B: Question 21 (4 marks)

Consider a convolutional neural network which takes as input a 65×77 color image (i.e. with three channels R, G, B). The first convolutional layer has 18 filters that are 5-by-5, with stride 3 and no zero-padding.

Compute the number of:

i. (1 mark) weights per filter in this layer (including bias) :

ii. (1 mark) neurons in this layer :

iii. (1 mark) connections into the neurons in this layer :

iv. (1 mark) independent parameters in this layer :

10

Ex 50

Question **19**

Answer saved

Marked out of 3.60

Flag question

Consider a convolutional neural network which takes as input a 65-by-77 color image (i.e. with three channels R, G, B). The first convolutional layer has 18 filters that are 3-by-3, with stride 2 and no zero-padding.

Compute the number of:

* weights per neuron in this layer (including bias): 28
* neurons in this layer: 21888
* connections into the neurons in this layer: 612864
* independent parameters in this layer: 504