CPU9444_Final4_yann

Autor: Yann

latest: Aug 2024

Reference:

University of Melbourne - COMP30024 Lecture slides

USYD - COMP5329 Lecture slides

UNSW - COMP9414 Lecture slides

UNSW - COMP9814 Lecture slides

Reinforcement Learning

智能体的目标是最大化总奖励，而不仅仅是单步的即时奖励。这意味着智能体需要考虑长期的效益，而不仅仅是眼前的利益。可能有的时候当前或局部的利益很大，但之后的收益很小，这种情况称为局部最优解（local maximum）。可以将其想象成一个吸引智能体的陷阱。强化学习可以看作是无监督学习，因为它是通过环境对智能体的奖励作为监督信号来进行学习的。智能体通过探索和利用的平衡，不断调整其策略，最终找到最优策略，以最大化长期奖励。

- **Policy:** 强化学习的目的是学习最优的策略（policy）。策略是状态到动作的映射，表示在每个状态下智能体应采取的动作。策略可以是确定性策略 $\pi(s) = a$ 或随机性策略 $\pi(a|s)$。策略指导智能体在某个状态下应该采取什么行动。
- **Reward Function:** 奖励函数定义了智能体在执行动作后从环境中获得的即时反馈。奖励函数 R(s, a) 用于鼓励智能体采取对目标有利的行为。通过这种方式，智能体能够学到对不同状态的价值评估。
- **Value Function:** 价值函数用于估计在给定策略下，从某个状态开始或采取某个动作的长期奖励期望值。状态值函数 $V^{\pi}(s)$ 和动作值函数 $Q^{\pi}(s, a)$ 。

Discount Factor

智能体的目标是最大化总奖励，而不是单步的即时奖励。这意味着智能体需要考虑长期的效益，而不仅仅是眼前的利益。为了实现这一目标，我们引入了折扣因子 γ，它在累积奖励计算中起到关键作用。帮助智能体平衡当前和未来的奖励，避免只关注即时奖励而忽视长期目标：

- 离散情况（Discrete Case），总奖励 G_t 可以表示为各个时间步奖励的和：

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

这里假设任务在时间 T 时终止。

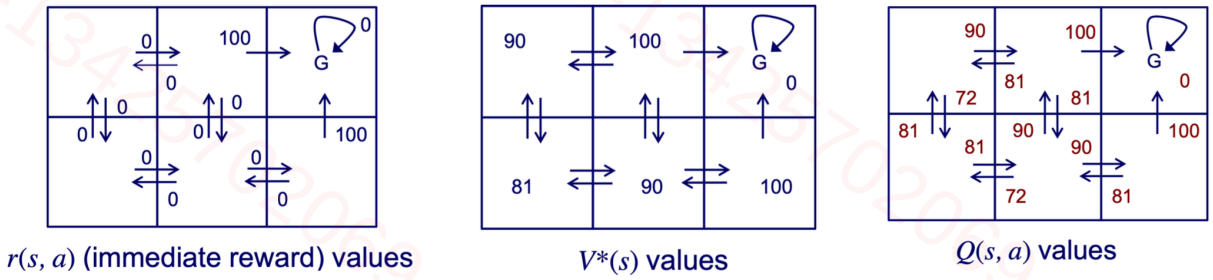
- 连续情况（Continuous Case），对于没有明确终止状态的连续任务，我们引入折扣因子 γ 来计算总奖励 G_t：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

折扣因子 γ 是一个介于 0 和 1 之间的数（$0 \leq \gamma < 1$）。它决定了未来奖励的现值。具体来说：

- 如果 $\gamma = 0$，智能体只关注即时奖励而忽视未来的奖励，表现为“近视”。
- 如果γ 接近 1，智能体则更加重视未来的奖励，表现为“有远见”。

Value Function



UNSW COMP9414 Lecture slides - Week3

- 状态值函数（State Value Function）：

$$V^{\pi}(s) = \mathbb{E}\pi \left[\sum t = 0^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s \right]$$

其中，γ 是折扣因子，表示未来奖励的当前价值。

- 动作值函数（Action Value Function）：

$$Q^{\pi}(s, a) = \mathbb{E}\pi \left[\sum t = 0^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a \right]$$

动作值函数表示在状态 s 采取动作 a 后获得的长期奖励期望值。

智能体通过学习策略 π 来最大化其期望的累积奖励。具体来说，智能体希望找到一个最优策略 π^*，*使得在每个状态* s *下，价值函数* $V^{\pi^*}(s)$ 最大。最优策略满足：

$$\pi^* = \arg\max_{\pi} V^{\pi}(s) \quad \forall s \in S$$

通过最大化期望累积奖励，智能体能够学会在不同状态下采取最优的行动，从而在长期内获得最大的总奖励。

Exercise

Ex 1

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 2

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 3

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 4

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 5

What does the **Bellman Equation** describe?

A. The relationship between state values and rewards

B. The process of updating policy probabilities

C. The recursive relationship for the value function in dynamic programming

D. The optimal policy derivation method

Answer: C

Ex 6

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 7

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 8

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 9

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 10

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 11

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:

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

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:

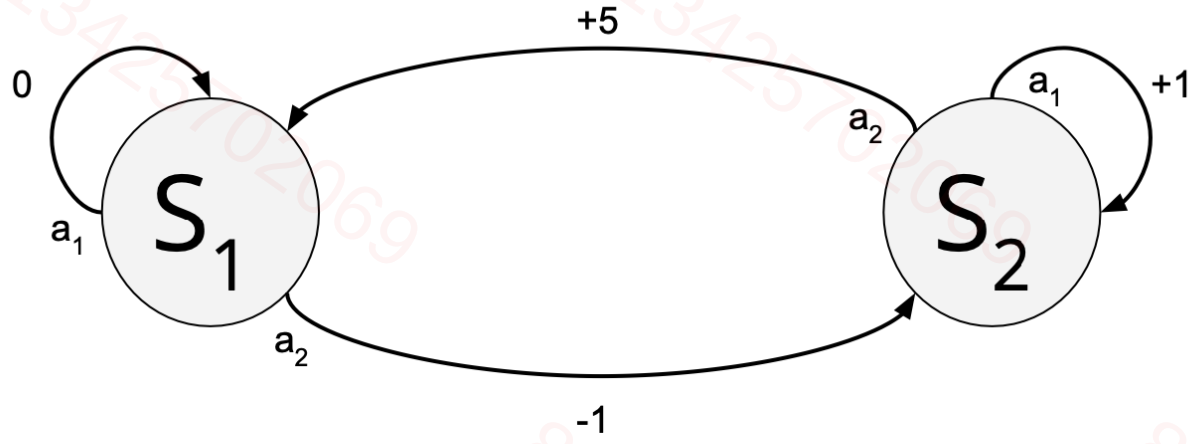


Figure 1: Draw of the world transitions.

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

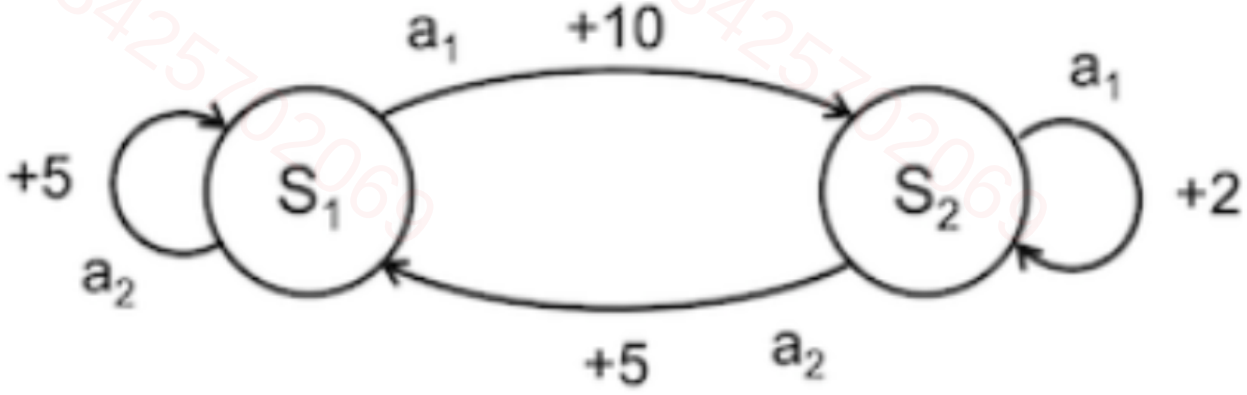
$V^*(S_1) = -1 + \gamma V^*(S_2)$
$V^*(S_2) = 5 + \gamma V^*(S_1)$
So $V^*(S_1) = -1 + 5\gamma + \gamma^2 V^*(S_1)$
$V^*(S_1) - \gamma^2 V^*(S_1) = -1 + 5\gamma$
$(1 - \gamma^2)V^*(S_1) = -1 + 5\gamma$
i.e. $V^*(S_1) = (-1 + 5\gamma)/(1 - \gamma^2) = 3.5/0.19 = 18.42$
And $V^*(S_2) = 5 + \gamma V^*(S_1)$
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 12



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

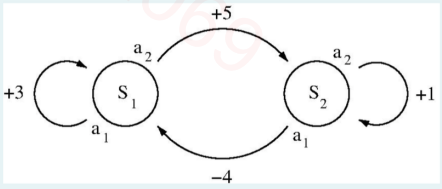
b. $V^*(S_1) = 50, V^*(S_2) = 50$

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

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

Ex 13

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 14

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.

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

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 15

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:

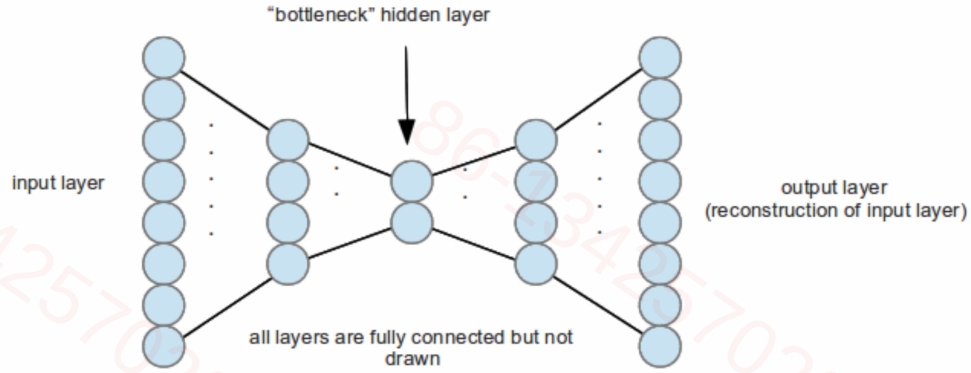
- a. Back Propagation Through Time
- b. Experience Replay
- c. Cross Entropy Minimization
- d. Increased Momentum

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 16

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)

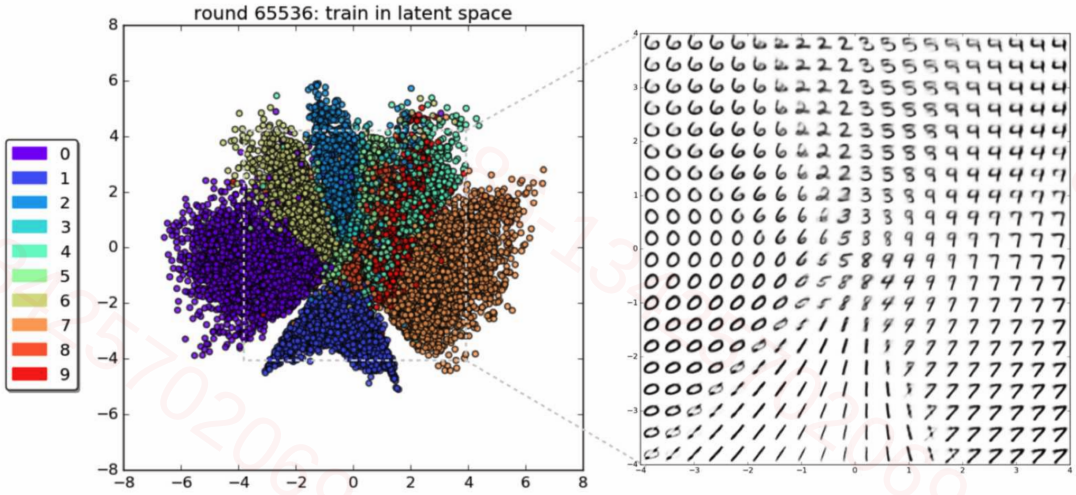
Instead of producing a single z for each $x^{(i)}$, the encoder (with parameters ϕ) can be made to produce a mean $\mu_{z|x^{(i)}}$ and standard deviation $\sigma_{z|x^{(i)}}$
This defines a conditional (Gaussian) probability distribution $q_\phi(z|x^{(i)})$
We then train the system to maximize

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

- 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 17

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 18

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 19

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 21

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 22

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