

# Solutions

10-601 Machine Learning  
Fall 2024  
Exam 3 Practice Problems  
Updated: December 2, 2024  
Time Limit: N/A

---

Name:  
AndrewID:

## Instructions:

- Fill in your name and Andrew ID above. Be sure to write neatly, or you may not receive credit for your exam.
  - Clearly mark your answers in the allocated space **on the front of each page**. If needed, use the back of a page for scratch space, but you will not get credit for anything written on the back of a page. If you have made a mistake, cross out the invalid parts of your solution, and circle the ones which should be graded.
  - No electronic devices may be used during the exam.
  - Please write all answers in pen.
  - You have N/A to complete the exam. Good luck!
-

## Instructions for Specific Problem Types

For “Select One” questions, please fill in the appropriate bubble completely:

**Select One:** Who taught this course?

- Henry Chai
- Marie Curie
- Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

**Select One:** Who taught this course?

- Henry Chai
- Marie Curie
- Noam Chomsky

For “Select all that apply” questions, please fill in all appropriate squares completely:

**Select all that apply:** Which are scientists?

- Stephen Hawking
- Albert Einstein
- Isaac Newton
- I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

**Select all that apply:** Which are scientists?

- Stephen Hawking
- Albert Einstein
- Isaac Newton
- I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

**Fill in the blank:** What is the course number?

10-601

10-~~7~~601

# 1 CNNs and RNNs

1. Let's begin by considering some of the high-level components of a CNN kernel along with the basic motivation.

- i. What is a kernel?

---

---

---

The collection of weights/filters that you will pass through an image, along with hyperparameters such as padding, stride, and filter size/quantity.

- ii. Why do we need stride, and what benefits/tradeoffs might different values of stride have on the output?

---

---

---

---

---

Stride is what allows the filter to actually pass through the CNN (defines at what pace the filter will move across the rows/columns). Larger values of stride can allow you to reduce the output dimensionality which could combat overfitting along with reducing computational power. The downside however is that you lose more and more information with larger values of stride, which could limit the upside of your model's accuracy.

- iii. What functionality does padding add to the kernel? Why might we want to use it?

---

---

---

---

---

Padding essentially makes sure that the output shape is the same as the input shape, and allows every pixel to be included in the convolution: if our filter is about to slide off of the original image padding allows the filter/image to still line up correctly. Furthermore, padding helps filters focus on the corner pixels just as much as middle pixels by making the filter pass through the corners multiple times as opposed to just once.

2. Consider the following image, filter, and output shape, which you have seen in prior

homework in this course.

$$x = \begin{array}{|c|c|c|c|c|c|} \hline 1 & 0 & -2 & 3 & 4 & 1 \\ \hline 2 & 9 & 5 & 6 & 0 & -1 \\ \hline 0 & -3 & 1 & 3 & 4 & 4 \\ \hline 6 & 5 & 2 & 0 & 6 & 8 \\ \hline -5 & 4 & -3 & 1 & 3 & -2 \\ \hline 4 & 1 & 2 & 8 & 9 & 7 \\ \hline \end{array}$$

$$F = \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

$$Y = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline e & f & g & h \\ \hline i & j & k & l \\ \hline m & n & o & p \\ \hline \end{array}$$

The shape of this particular  $Y$  is that of an kernel using no padding and a stride of 1.

- i. Suppose we decide that, instead of having our output shape be  $(4, 4)$ , we want a slightly smaller,  $(3, 3)$  image as output for the kernel. In order for this to happen, what is the smallest combination of stride and padding that would work?

$$s=2, p=1$$

- ii. Let's make this a bit more general. Suppose our original image of shape  $(a, a)$ , and we want the shape of our final image to be of shape  $(b, b)$ , where  $b \leq a$ . Furthermore the shape of the filter is  $(k, k)$  and the stride length is  $s$ . Express  $b$  in terms of all defined variables.

$$b \approx \lfloor ((a + 2p - k)/s) + 1 \rfloor$$

- iii. Finally, suppose that we want to add another convolutional layer after this forward pass from part i) - in particular we want to *output a single-celled,  $(1, 1)$ , binary (e.g.  $\{-1, 1\}$ ) label*. Describe a function that could allow you to do this. **Hint:** consider some of the basic classification methods we've used in the past!

---



---



---

answer: make it a perceptron (3x3 filter with 0 stride  $\rightarrow$  a single number, take the sign of this number as a prediction); other correct answers may be accepted such as logistic regression

3. What's the difference between upsampling and downsampling in CNNs? What are the

appropriate scenarios to use them?

---



---



---



---



---



---

Downsampling: the goal is to *reduce* the output dimensionality. Some common methods include pooling functions (e.g. maxpooling, mean pooling). These are appropriate for larger images where you want to reduce the compute time/mitigate overfitting by making the input to future convolutions less complex.

Upsampling: the goal is to *increase* the output dimensionality. Upsampling is generally used when you want the output of a convolution to be larger e.g. match the input image in dimensionality. For instance you might be assigning a label to each pixel of the output and matching it with the input.

4. We finally discuss aspects of parameter sharing.

- i. Consider the setup from 2ii), with the same variables representing the same quantities. How many parameters do we learn, both with and without parameter sharing?

$k^2, k^2b^2$

- ii. Suppose that this CNN is being used for a prediction task in a subject you have prior knowledge in - for instance, suppose you are being asked to classify the image into classes of cars, and you know that each image contains a side view of the car with the front of the car facing right, with each car being roughly the same size. In this scenario, would parameter sharing be appropriate, disregarding computational constraints? Why or why not?

---



---



---

Here it is more appropriate not to share parameters. Because we know that all filters placed on the same area within a given image will be looking for the same thing (e.g. bumpers, rear windows, front wheels, etc.), having one set of parameters try to identify each of these features is not necessary when we can have a set of filters, each responsible identifying a different segment/part of the car.

5. **Select all that apply:** It is more appropriate to use RNN's over CNN's in these settings:
- Speech recognition
  - Facial recognition
  - Music composition
  - Autocorrect system
  - All of the above

A, C, D

6. **True or False:** RNN's are helpful in analyzing time series data. Explain your reasoning in 1-2 sentences.
- True
  - False

---

---

---

True. RNNs can handle sequential information, and are also able to incorporate information from previous time steps into the current time step. In addition, RNNs can process inputs of variable lengths, which is helpful for time series data.

## 2 Transformers

1. **True or False:** Masking in transformers is used to enhance the model's focus on specific parts of the input sequence during training.
  - True
  - False

False. Masking in transformers, especially in the context of sequence-to-sequence tasks, is primarily used to prevent the model from accessing future tokens in the sequence during training, not to enhance focus on specific parts of the input. This is crucial for maintaining the causality principle in language modeling tasks.

2. **Short answer:** Explain why padding and truncation are important in training transformer models. Provide a scenario where padding is necessary and another where truncation is required, explaining the implications of each in the context of a natural language processing task.

---

---

---

---

---

---

**Padding:** Transformer models work with batches of data. For efficient computation, all sequences in a batch need to be of the same length. However, natural language data typically consists of sentences or sequences of varying lengths. Padding addresses this issue by adding extra tokens (usually zero values or a special token like [PAD]) to shorter sequences to match the length of the longest sequence in the batch.

Scenario: Consider a batch of sentences for sentiment analysis where each sentence has a different length. For instance, "I love this" (3 words) and "This product is absolutely fantastic!" (5 words) are shorter than "I had an amazing experience with the customer service team" (9 words). To process these sentences in a single batch, padding would be used to extend the shorter sentences to the length of the longest one.

Implications: Padding ensures that all input sequences in a batch can be processed together, which is essential for the parallel processing capabilities of GPUs. However, excessive padding can lead to inefficiencies and might even impact model performance if the model starts learning from the padding rather than the actual content.

**Truncation:** Truncation is used to limit the length of the input sequences so that they do not exceed a predefined maximum length. Truncating longer sequences helps manage computational resources and memory usage.

Scenario: In a language modeling task, you might encounter exceptionally long para-

graphs or documents. If the maximum sequence length for the model is set to 512 tokens, but a document contains 1000 tokens, truncation will be used to cut the document down to the first 512 tokens.

Implications: Truncation helps in controlling the computational load and memory usage, which is critical for training large models like transformers efficiently. However, it may lead to the loss of potentially important information at the end of the sequences that are truncated. This trade-off needs to be managed carefully, often by segmenting long texts into smaller, meaningful chunks that can be processed without significant loss of context or meaning.

3. In the context of sequence learning with classical RNN architectures, what is the primary challenge associated with forgetting, and why does it occur?
- Forgetting occurs due to the network's inability to store information permanently, making it difficult to recall specific details from long sequences.
  - The vanishing gradient problem leads to forgetting, as it causes the gradients to become very small, reducing the network's ability to learn dependencies between distant sequence elements.
  - Forgetting is caused by the network's overemphasis on recent inputs, which overshadows earlier inputs in long sequences.
  - The issue arises from the network's fixed-size memory, which is insufficient for storing all the information from long sequences.
  - None of the above

B only. The vanishing gradient problem leads to forgetting, as it causes the gradients to become very small, reducing the network's ability to learn dependencies between distant sequence elements.

4. **Short answer:** What is the major problem that is addressed by layer normalization?

---

---

---

---

---

---

**Internal Covariate Shift:** Internal covariate shift refers to the change in the distribution of network activations due to the updates in the network parameters during training. This shift can make the training process slower and less stable, as each layer needs to continuously adapt to new distributions.

5. **Short answer:** Highlight in 1-2 sentences the importance of multihead attention over



single head attention.

---

---

---

---

---

---

With multiple attention heads, the model can focus on different aspects of the input, enhancing its ability to understand complex dependencies in tasks like natural language processing. Each head will pay attention to a distinct input element individually, the model does a better job of capturing positional detail. Since it can also be parallelized it also boosts computational efficiency which makes transformers more versatile.

6. **Short answer:** A pre-trained language model is to be deployed for two tasks: medical report analysis and sentiment classification of a product review. For medical report analysis, the user will ask the model follow-up questions about a report it is prompted with. For sentiment classification, the model is prompted with a review for a product and must classify the review as either {positive, negative or neutral}. For each task, decide whether fine-tuning or in-context learning would be more appropriate and justify your choice with specific reasons related to the nature of each task.

---

---

---

For medical report analysis, fine-tuning would be more appropriate due to the specialized and technical nature of medical reports, requiring the model to adapt to specific medical terminologies and concepts. In contrast, for sentiment classification, in-context learning might be better suited as it allows the model to leverage its pre-trained knowledge of language and style. The model doesn't need extensive specialized knowledge to know if a review is positive or negative, it should already know what language is associated with these concepts.

### 3 Reinforcement Learning

#### 3.1 Markov Decision Process

**Environment Setup** (may contain spoilers for Shrek 1)

Lord Farquaad is hoping to evict all fairytale creatures from his kingdom of Duloc, and has one final ogre to evict: Shrek. Unfortunately all his previous attempts to catch the crafty ogre have fallen short, and he turns to you, with your knowledge of Markov Decision Processes (MDP's) to help him catch Shrek once and for all.

Consider the following MDP environment where the agent is Lord Farquaad:

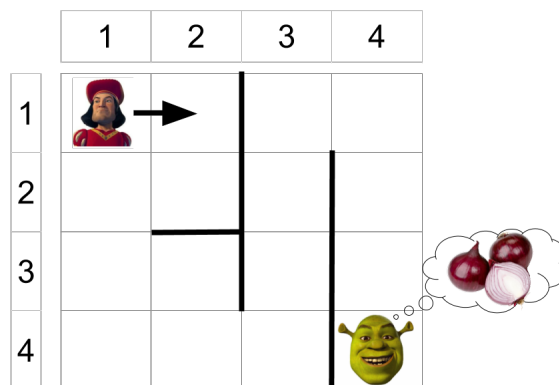


Figure 1: Kingdom of Duloc, circa 2001

Here's how we will define this MDP:

- **$S$  (state space):** a set of states the agent can be in. In this case, the agent (Farquaad) can be in any location  $(row, col)$  and also in any orientation  $\in \{N, E, S, W\}$ . Therefore, state is represented by a three-tuple  $(row, col, dir)$ , and  $S =$  all possible of such tuples. Farquaad's start state is  $(1, 1, E)$ .
- **$A$  (action space):** a set of actions that the agent can take. Here, we will have just three actions: turn right, turn left, and move forward (turning does not change  $row$  or  $col$ , just  $dir$ ). So our action space is  $\{R, L, M\}$ . Note that Farquaad is debilitatingly short, so he cannot travel through (or over) the walls. Moving forward when facing a wall results in no change in state (but counts as an action).
- **$R(s, a)$  (reward function):** In this scenario, Farquaad gets a reward of 5 by moving into the swamp (the cell containing Shrek), and a reward of 0 otherwise.
- **$p(s'|s, a)$  (transition probabilities):** We'll use a deterministic environment, so this will be 1 if  $s'$  is reachable from  $s$  and by taking  $a$ , and 0 if not.

1. What are  $|S|$  and  $|A|$  (size of state space and size of action space)?

$$|S| = 4 \text{ rows} \times 4 \text{ columns} \times 4 \text{ orientations} = 64$$

$$|A| = |\{R, L, M\}| = 3$$

2. Why is it called a "Markov" decision process? (Hint: what is the assumption made with  $p$ ?)

$p(s'|s, a)$  assumes that  $s'$  is determined only by  $s$  and  $a$  (and not any other previous states or actions).

3. What are the following transition probabilities?

$$p((1, 1, N)|(1, 1, N), M) =$$

$$p((1, 1, N)|(1, 1, E), L) =$$

$$p((2, 1, S)|(1, 1, S), M) =$$

$$p((2, 1, E)|(1, 1, S), M) =$$

$$p((1, 1, N)|(1, 1, N), M) = 1$$

$$p((1, 1, N)|(1, 1, E), L) = 1$$

$$p((2, 1, S)|(1, 1, S), M) = 1$$

$$p((2, 1, E)|(1, 1, S), M) = 0$$

4. Given a start position of  $(1, 1, E)$  and a discount factor of  $\gamma = 0.5$ , what is the expected discounted future reward from  $a = R$ ? For  $a = L$ ? (Fix  $\gamma = 0.5$  for following problems).

For  $a = R$  we get  $R_R = 5 * (\frac{1}{2})^{16}$  (it takes 17 moves for Farquaad to get to Shrek, starting with  $R, M, M, M, L...$ )

For  $a = L$ , this is a bad move, and we need another move to get back to our original orientation, from which we can go with our optimal policy. So the reward here is:

$$R_L = (\frac{1}{2})^2 * R_R = 5 * (\frac{1}{2})^{18}$$

5. What is the optimal action from each state, given that orientation is fixed at  $E$ ? (if there are multiple options, choose any)

R	R	M	R
R	R	L	R
M	R	L	R
M	M	L	-

(some have multiple options, I just chose one of the possible ones)

6. Farquaad's chief strategist (Vector from Despicable Me) suggests that having  $\gamma = 0.9$  will result in a different set of optimal policies. Is he right? Why or why not?

Vector is wrong. While the reward quantity will be different, the set of optimal policies does not change. (it is now  $5 * (\frac{9}{10})^{16}$ ) (one can only assume that Lord Farquaad and Vector would be in kahoots: both are extremely nefarious!)

7. Vector then suggests the following setup:  $R(s, a) = 0$  when moving into the swamp, and  $R(s, a) = -1$  otherwise. Will this result in a different set of optimal policies? Why or why not?

It will not. While the reward quantity will be different, the set of optimal policies does not change. (Farquaad will still try to minimize the number of steps he takes in order to reach Shrek)

8. Vector now suggests the following setup:  $R(s, a) = 5$  when moving into the swamp, and  $R(s, a) = 0$  otherwise, but with  $\gamma = 1$ . Could this result in a different optimal policy? Why or why not?

This will change the policy, but not in Lord Farquaad's favor. He will no longer be incentivized to reach Shrek quickly (since  $\gamma = 1$ ). The optimal reward from each state is the same (5) and therefore each action from each state is also optimal. Vector really should have taken 10-301/601...

9. Surprise! Elsa from Frozen suddenly shows up. Vector hypnotizes her and forces her to use her powers to turn the ground into ice. The environment is now stochastic: since the ground is now slippery, when choosing the action  $M$ , with a 0.2 chance, Farquaad will slip and move two squares instead of one. What is the expected future-discounted rewards from  $s = (2, 4, S)$ ?

Recall that  $R_{exp} = \max_a E[R(s, a) + \gamma R_{s'}]$

(notation might be different than in the notes, but conceptually, our reward is the best expected reward we can get from taking any action  $a$  from our current state  $s$ .)

In this case, our best action is obviously to move forward. So we get

$R_{exp} = (\text{expected value of going two steps}) + (\text{expected value of going one step})$

$$E[2_{steps}] = p((4, 4, S)|(2, 4, S), M) \times R((4, 4, S), (2, 4, S), M) = 0.2 \times 5 = 1$$

$$E[1_{step}] = p((4, 3, S)|(2, 4, S), M) \times (R((4, 3, S), (2, 4, S), M) + \gamma R_{(4,3,S)})$$

where  $R_{(4,3,S)}$  is the expected reward from  $(4, 3, S)$ . Since the best reward from here is obtained by choosing  $a = M$ , and we always end up at Shrek, we get

$$E[1_{step}] = 0.8 \times (0 + \gamma \times 5) = 0.8 \times 0.5 \times 5 = 2$$

giving us a total expected reward of  $R_{exp} = 1 + 2 = 3$

(I will be very disappointed if this is not the plot of Shrek 5)

### 3.2 Value and Policy Iteration

1. **Select all that apply:** Which of the following environment characteristics would increase the computational complexity per iteration for a value iteration algorithm? Choose all that apply:
- Large Action Space
  - A Stochastic Transition Function
  - Large State Space

- Unknown Reward Function
- None of the Above

A and C (state space and action space). The computational complexity for value iteration per iteration is  $O(|A||S|^2)$

B is NOT correct. The time complexity is  $O(|A||S|^2)$  for both stochastic and deterministic transition (review the lecture slides).

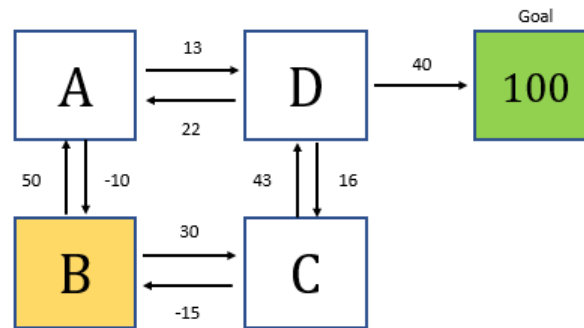
2. **Select all that apply:** Which of the following environment characteristics would increase the computational complexity per iteration for a policy iteration algorithm? Choose all that apply:

- Large Action Space
- A Stochastic Transition Function
- Large State Space
- Unknown Reward Function
- None of the Above

A and C again. The computational complexity for policy iteration per iteration is  $O(|A||S|^2 + |S|^3)$

Again, B is NOT correct.

3. In the image below is a representation of the game that you are about to play. There are 5 states: A, B, C, D, and the goal state. The goal state, when reached, gives 100 points as reward (that is, you can assume  $R(D, \text{right}) = 140$ ). In addition to the goal's points, you also get points by moving to different states. The amount of points you get are shown next to the arrows. You start at state B. To figure out the best policy, you use asynchronous value iteration with a decay ( $\gamma$ ) of 0.9. You should initialize the value of each state to 0.



- (i) When you first start playing the game, what action would you take (up, down, left, right) at state B?

Up

- (ii) What is the total reward at state B at this time?

50 (immediate reward of 50, and future reward (value at state A) starts at 0)

- (iii) Let's say you keep playing until your total values for each state has converged. What action would you take at state B?

C

- (iv) What is the total reward at state B at this time?

182.1 (30 from the immediate action, and  $43 * 0.9 + (100 + 40) * 0.9^2 = 152.1$  from the future reward (value at state C))

4. **Select one:** Let  $V_k(s)$  indicate the value of state  $s$  at iteration  $k$  in (synchronous) value iteration. What is the relationship between  $V_{k+1}(s)$  and  $\sum_{s' \in S} P(s'|s, a)[R(s, a, s') + \gamma V_k(s')]$ , for any  $a \in A$ ? Indicate the most restrictive relationship that applies. For example, if  $x < y$  always holds, use  $<$  instead of  $\leq$ . Selecting  $?$  means it's not possible to assign any true relationship. Assume  $R(s, a, s') \geq 0 \forall s, s' \in S, a \in A$ .

$$V_{k+1}(s) \square \sum_{s'} P(s'|s, a)[R(s, a, s') + \gamma V_k(s')]$$

- =  
 <  
 >  
 ≤  
 ≥  
 ?

E

### 3.3 Q-Learning

1. For the following true/false, circle one answer and provide a one-sentence explanation:
- (i) One advantage that Q-learning has over Value and Policy iteration is that it can account for non-deterministic policies.
- Circle one:**    True    False
- False.** All three methods can account for non-deterministic policies
- (ii) You can apply Value or Policy iteration to any problem that Q-learning can be applied to.
- Circle one:**    True    False
- False.** Unlike the others, Q-learning doesn't need to know the transition probabilities ( $p(s' | s, a)$ ), or the reward function ( $r(s,a)$ ) to train. This is its biggest advantage.
- (iii) Q-learning is guaranteed to converge to the true value  $Q^*$  for a greedy policy.
- Circle one:**    True    False
- False.** Q-learning converges only if every state will be explored infinitely. Thus, purely exploiting policies (e.g. greedy policies) will not necessarily converge to  $Q^*$ , but rather to a local optimum.
2. For the following parts of this problem, recall that the update rule for Q-learning is:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \left( q(\mathbf{s}, a; \mathbf{w}) - (r + \gamma \max_{a'} q(\mathbf{s}', a'; \mathbf{w})) \right) \nabla_{\mathbf{w}} q(\mathbf{s}, a; \mathbf{w})$$



- (i) From the update rule, let's look at the specific term  $X = (r + \gamma \max_{a'} q(s', a'; \mathbf{w}))$ . Describe in English what is the role of X in the weight update.

Estimate of true total return ( $Q^*(s,a)$ ). This may get multiple answers, so grade accordingly

- (ii) Is this update rule synchronous or asynchronous?

Asynchronous

- (iii) A common adaptation to Q-learning is to incorporate rewards from more time steps into the term X. Thus, our normal term  $r_t + \gamma \max_{a_{t+1}} q(s_{t+1}, a_{t+1}; w)$  would become  $r_t + \gamma * r_{t+1} + \gamma^2 \max_{a_{t+2}} q(s_{t+2}, a_{t+2} : \mathbf{w})$ . What are the advantages of using more rewards in this estimation?

Incorporating rewards from multiple time steps allows for a more "realistic" estimate of the true total reward, since a larger percentage of it is from real experience. It can help with stabilizing the training procedure, while still allowing training at each time step (bootstrapping). This type of method is called N-Step Temporal Difference Learning.

3. **Select one:** Let  $Q(s, a)$  indicate the estimated Q-value of state-action pair  $(s, a) \in |S| \times |A|$  at some point during Q-learning. Suppose you receive reward  $r$  after taking action  $a$  at state  $s$  and arrive at state  $s'$ . Before updating the Q values based on this experience, what is the relationship between  $Q(s, a)$  and  $r + \gamma \max_{a' \in A} Q(s', a')$ ? Indicate the most restrictive relationship that applies. For example, if  $x < y$  always holds, use  $<$  instead of  $\leq$ . Selecting ? means it's not possible to assign any true relationship.

$$Q(s, a) \square r + \gamma \max_{a'} Q(s', a')$$

- =
- <
- >
- $\leq$
- $\geq$
- ?

F

4. During standard (not deep) Q-learning, you get reward  $r$  after taking action *North* from state *A* and arriving at state *B*. You compute the sample  $r + \gamma Q(B, \textit{South})$ , where  $\textit{South} = \arg \max_a Q(B, a)$ .

Which of the following Q-values are updated during this step? (Select all that apply)

- $Q(A, \textit{North})$

- Q(A, South)
- Q(B, North)
- Q(B, South)
- None of the above

**A**

5. In general, for Q-Learning (standard/tabular Q-learning, not approximate Q-learning) to converge to the optimal Q-values, which of the following are true?

**True or False:** It is necessary that every state-action pair is visited infinitely often.

- True
- False

**True or False:** It is necessary that the discount  $\gamma$  is less than 0.5.

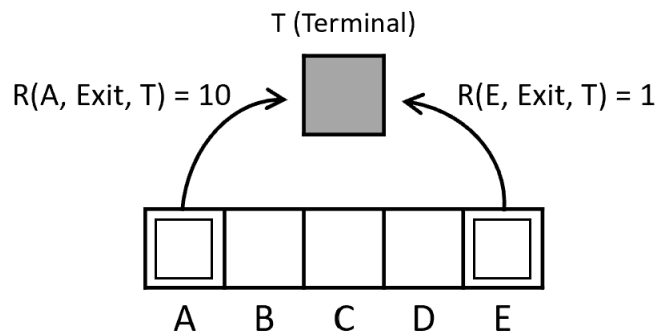
- True
- False

**True or False:** It is necessary that actions get chosen according to  $\arg \max_a Q(s, a)$ .

- True
- False

(1) **True:** In order to ensure convergence in general for Q learning, this has to be true. In practice, we generally care about the policy, which converges well before the values do, so it is not necessary to run it infinitely often. (2) **False:** The discount factor must be greater than 0 and less than 1, not 0.5. (3) **False:** This would actually do rather poorly, because it is purely exploiting based on the Q-values learned thus far, and not exploring other states to try and find a better policy.

6. Consider training a robot to navigate the following grid-based MDP environment.



- There are six states, A, B, C, D, E, and a terminal state T.
- Actions from states B, C, and D are Left and Right.

- The only action from states A and E is Exit, which leads deterministically to the terminal state

The reward function is as follows:

- $R(A, Exit, T) = 10$
- $R(E, Exit, T) = 1$
- The reward for any other tuple  $(s, a, s')$  equals -1

Assume the discount factor is 1. When taking action Left, with probability 0.8, the robot will successfully move one space to the left, and with probability 0.2, the robot will move one space in the opposite direction. When taking action Right, with probability 0.8, the robot will successfully move one space to the right, and with probability 0.2, the robot will move one space in the opposite direction. Run synchronous value iteration on this environment for two iterations. Begin by initializing the value of all states to zero.

Write the value of each state after the first ( $k = 1$ ) and the second ( $k = 2$ ) iterations. Write your values as a comma-separated list of 6 numerical expressions in the alphabetical order of the states, specifically  $V(A), V(B), V(C), V(D), V(E), V(T)$ . Each of the six entries may be a number or an expression that evaluates to a number. Do not include any max operations in your response.

$V_1(A), V_1(B), V_1(C), V_1(D), V_1(E), V_1(T)$  (Values for 6 states):

10, -1, -1, -1, 1, 0

$V_2(A), V_2(B), V_2(C), V_2(D), V_2(E), V_2(T)$  (values for 6 states):

10, 6.8, -2, -0.4, 1, 0

What is the resulting policy after this second iteration? Write your answer as a comma-separated list of three actions representing the policy for states, B, C, and D, in that order. Actions may be Left or Right.

$\pi(B), \pi(C), \pi(D)$  based on  $V_2$  :

Left, Left, Right

## 4 Ensemble Methods

### 4.1 AdaBoost

- In the AdaBoost algorithm, if the final hypothesis makes no mistakes on the training data, which of the following is correct?

Select all that apply:

- Additional rounds of training can help reduce the errors made on unseen data.
- Additional rounds of training have no impact on unseen data.
- The individual weak learners also make zero error on the training data.
- Additional rounds of training always leads to worse performance on unseen data.

A. AdaBoost is empirically robust to overfitting and the testing error usually continues to reduce with more rounds of training.

- True or False:** In AdaBoost weights of the misclassified examples go up by the same multiplicative factor.

True

False

True, follows from the update equation.

Round	$D_t(A)$	$D_t(B)$	$D_t(C)$	$D_t(D)$	$D_t(E)$	$D_t(F)$
1	?	?	$\frac{1}{6}$	?	?	?
2	?	?	?	?	?	?
...						
219	?	?	?	?	?	?
220	$\frac{1}{14}$	$\frac{1}{14}$	$\frac{7}{14}$	$\frac{1}{14}$	$\frac{2}{14}$	$\frac{2}{14}$
221	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{7}{20}$	$\frac{1}{20}$	$\frac{1}{4}$	$\frac{1}{10}$
...						
3017	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{16}$	0
...						
8888	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{1}{8}$	$\frac{2}{8}$	$\frac{3}{8}$	$\frac{1}{8}$

- In the last semester, someone used AdaBoost to train some data and recorded all the weights throughout iterations but some entries in the table are not recognizable. Clever as you are, you decide to employ your knowledge of Adaboost to determine some of the missing information.

Below, you can see part of table that was used in the problem set. There are columns for the Round # and for the weights of the six training points (A, B, C, D, E, and F) at the start of each round. Some of the entries, marked with “?”, are impossible for you to read.

In the following problems, you may assume that non-consecutive rows are independent of each other, and that a classifier with error less than  $\frac{1}{2}$  was chosen at each step.

- (a) The weak classifier chosen in Round 1 correctly classified training points A, B, C, and E but misclassified training points D and F. What should the updated weights have been in the following round, Round 2? Please complete the form below.

Round	$D_2(A)$	$D_2(B)$	$D_2(C)$	$D_2(D)$	$D_2(E)$	$D_2(F)$
2						

$\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{1}{8}, \frac{1}{4}$

- (b) During Round 219, which of the training points (A, B, C, D, E, F) must have been misclassified, in order to produce the updated weights shown at the start of Round 220? List all the points that were misclassified. If none were misclassified, write ‘None’. If it can’t be decided, write ‘Not Sure’ instead.

Not sure

- (c) You observe that the weights in round 3017 or 8888 (or both) cannot possibly be right. Which one is incorrect? Why? Please explain in one or two short sentences.
- Round 3017 is incorrect.  
 Round 8888 is incorrect.  
 Both rounds 3017 and 8888 are incorrect.

C. 3017: weight cannot be 0; 8888: sum of weights should be 1.

4. What condition must a weak learner satisfy in order for boosting to work?

**Short answer:**

The weak learner must classify above chance performance.

5. After an iteration of training, AdaBoost more heavily weights which data points to train the next weak learner? (Provide an intuitive answer with no math symbols.)

**Short answer:**

The data points that are incorrectly classified by weak learners trained in previous iterations are more heavily weighted.

6. **Extra credit** Do you think that a deep neural network is nothing but a case of boosting? Why or why not? Impress us.

**Answer:**

Both viewpoints can be argued. One may view passing a linear combination through a nonlinear function as a weak learner (e.g., logistic regression), and that the deep neural network corrects for errors made by these weak learners in deeper layers. Then again, every layer of the deep neural network is optimized in a global fashion (i.e., all weights are updated simultaneously) to improve performance, which could possibly capture dependencies which boosting could not.

Almost all coherent answers should be accepted, with full points to those who strongly argue their position with ML ideas.

## 4.2 Random Forests

1. Consider a random forest ensemble consisting of 5 decision trees DT1, DT2 ... DT5 that has been trained on a dataset consisting of 7 samples. Each tree has been trained on a random subset of the dataset. The following table represents the predictions of each tree on its out-of-bag samples.

- (a) What is the OOB error of the above random forest classifier?

OOB is the average error of the table which is 0.5.

- (b) In the above random forest classifier, which Decision tree(s) will be given the

Tree	Sample Number	Prediction	Actual
DT1	6	No	Yes
DT1	7	No	Yes
DT2	2	No	No
DT3	1	No	No
DT3	2	Yes	No
DT3	4	Yes	Yes
DT4	2	Yes	No
DT4	7	No	Yes
DT5	3	Yes	Yes
DT5	5	No	No

highest weight in inference? If there are multiple trees, mention them all

DT1, DT2, DT3, DT4, DT5. Random Forests do unweighted sums of the individual tree predictions

- (c) To reduce the error of each individual decision tree, Neural uses all the features to train each tree. How would this impact the generalisation error of the random forest?
- The generalisation error would decrease as each tree has lower generalisation error
  - The generalisation error would increase as each tree has insufficient training data
  - The generalisation error would increase as the trees are highly correlated

The generalisation error would increase as the trees are highly correlated



## 5 Recommender Systems

1. Applied to the Netflix Prize problem, which of the following methods does NOT always require side information about the users and the movies?

Select all that apply:

- Neighborhood methods
- Content filtering
- Latent factor methods
- Collaborative filtering
- None of the above

ACD

2. Select all that apply:

- Using matrix factorization, we can embed both users and items in the same space
- Using matrix factorization, we can embed either solely users or solely items in the same space, as we cannot combine different types of data
- In a rating matrix of users by books that we are trying to fill up, the best-known solution is to fill the empty values with 0s and apply PCA, allowing the dimensionality reduction to make up for this lack of data
- Alternating minimization allows us to minimize over two variables
- Alternating minimization avoids the issue of getting stuck in local minima
- If the data is multidimensional, then overfitting is extremely rare
- Nearest neighbor methods in recommender systems are restricted to using euclidian distance for their distance metric
- None of the above

AD

Filling empty values with 0s is not ideal since we are assuming data values that are not necessarily true. Thus, we cannot apply PCA when there is missing values.

Alternating minimization can still get stuck at a local minimum.

Both euclidian distance and cosine similarity are valid metrics.

3. Your friend Duncan wants to build a recommender system for his new website DuncTube, where users can like and dislike videos that are posted there. In order to build his system using collaborative filtering, he decides to use Non-Negative Matrix Factorization. What is an issue with Duncan's approach, and what could he change about the website *or* the algorithm in order to fix it?



Since Duncan's website incorporates negative responses directly, NNMF can't be used to model these sorts of responses (since NNMF enforces that both the original and the factored matrices are all non-negative). To fix this, Duncan would either have to remove the dislike option from his website, OR use a different matrix factorization algorithm like SVD.

4. You and your friends want to build a movie recommendation system based on collaborative filtering. There are three websites (A, B and C) that you decide to extract users rating from. On website A, the rating scale is from 1 to 5. On website B, the rating scale is from 1 to 10. On website C, the rating scale is from 1 to 100. Assume you will have enough information to identify users and movies on one website with users and movies on another website. Would you be able to build a recommendation system? And briefly explain how would you do it?



Yes. We would be able to do it. First, Normalize the ratings score within certain range. (E.g. re-scale each dataset ratings to a 0-1 range). After that, combine users ratings of the three websites by matching movies and users. With users rating, we could conduct Matrix Factorization to predict the missing ratings for users. (Or Neighborhood method)

5. What is the difference between collaborative filtering and content filtering?



Content filtering assumes access to side information about items and content filtering does not.

## 6 K-Means

1. For **True or False** questions, circle your answer and justify it; for **QA** questions, write down your answer.

(i) For a particular dataset and a particular  $k$ ,  $k$ -means always produce the same result, if the initialized centers are the same. Assume there is no tie when assigning the clusters.

True

False

**Justify your answer:**

---

True. Every time you are computing the completely same distances, so the result is the same.

(ii)  $k$ -means can always converge to the global optimum.

True

False

**Justify your answer:**

---

False. It depends on the initialization. Random initialization could possibly lead to a local optimum.

(iii)  $k$ -means is not sensitive to outliers.

True

False

**Justify your answer:**

---

False.  $k$ -means is quite sensitive to outliers, since it computes the cluster center based on the mean value of all data points in this cluster.

(iv)  $k$  in  $k$ -nearest neighbors and  $k$ -means have the same meaning.

True

False

**Justify your answer:**

---

False. In knn,  $k$  is the number of data points we need to look at when classifying a data point. In  $k$ -means,  $k$  is the number of clusters.

- (v) What's the biggest difference between  $k$ -nearest neighbors and  $k$ -means?

**Write your answer in one sentence:**

---

knn is a supervised algorithm, while  $k$ -means is unsupervised.

- (vi) In  $k$ -means, the cost always drops after one update step.

- True  
 False

True.

- (vii)  $k$ -means is more likely to pick the wrong centers when number of clusters  $k$  increases.

- True  
 False

True.

- (viii) Recall the  $k$ -means++ algorithm from lecture. Here we provide the generalized version of  $k$ -means++:

- Choose  $\mathbf{c}_1$  at random.
- For  $j = 2, \dots, K$ 
  - Pick  $\mathbf{c}_j$  among  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}$  according to the distribution

$$P(\mathbf{c}_j = \mathbf{x}^{(i)}) \propto \min_{j' < j} \|\mathbf{x}^{(i)} - \mathbf{c}_{j'}\|^\alpha$$

The lecture version uses  $\alpha = 2$ .

- True  
 False

True.

- (ix) When  $\alpha$  in  $k$ -means++ becomes 0, it means random sampling.

- True  
 False

True.

2. In k-means, random initialization could possibly lead to a local optimum with very bad performance. To alleviate this issue, instead of initializing all of the centers completely randomly, we decide to use a smarter initialization method. This leads us to k-means++.

The only difference between k-means and k-means++ is the initialization strategy, and all of the other parts are the same. The basic idea of k-means++ is that instead of simply choosing the centers to be random points, we sample the initial centers iteratively, each time putting higher probability on points that are far from any existing center. Formally, the algorithm proceeds as follows.

**Given:** Data set  $x^{(i)}, i = 1, \dots, N$

**Initialize:**

$$\mu^{(1)} \sim \text{Uniform}(\{x^{(i)}\}_{i=1}^N)$$

For  $j = 2, \dots, k$

Computing probabilities of selecting each point

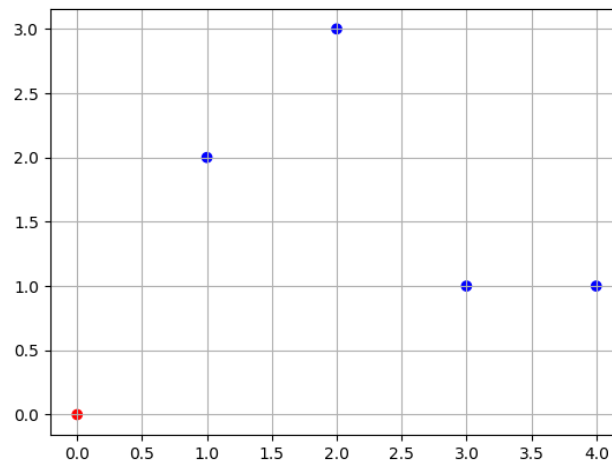
$$p_i = \frac{\min_{j' < j} \|\mu^{(j')} - x^{(i)}\|_2^2}{\sum_{i'=1}^N \min_{j' < j} \|\mu^{(j')} - x^{(i')}\|_2^2}$$

Select next center given the appropriate probabilities

$$\mu^{(j)} \sim \text{Categorical}(\{x^{(i)}\}_{i=1}^N, \mathbf{p}_{1:N})$$

Note: n is the number of data points, k is the number of clusters. For cluster 1's center, you just randomly choose one data point. For the following centers, every time you initialize a new center, you will first compute the distance between a data point and the center closest to this data point. After computing the distances for all data points, perform a normalization and you will get the probability. Use this probability to sample for a new center.

Now assume we have 5 data points (n=5): (0, 0), (1, 2), (2, 3), (3, 1), (4, 1). The number of clusters is 3 (k=3). The center of cluster 1 is randomly chosen as (0, 0). These data points are shown in the figure below.



- (i) What is the probability of every data point being chosen as the center for cluster 2? (The answer should contain 5 probabilities, each for every data point)

(0, 0): 0  
(1, 2): 0.111  
(2, 3): 0.289  
(3, 1): 0.222  
(4, 1): 0.378

- (ii) Which data point is mostly likely chosen as the center for cluster 2?

(4, 1) is mostly likely chosen.

- (iii) Assume the center for cluster 2 is chosen to be the most likely one as you computed in the previous question. Now what is the probability of every data point being chosen as the center for cluster 3? (The answer should contain 5 probabilities, each for every data point)

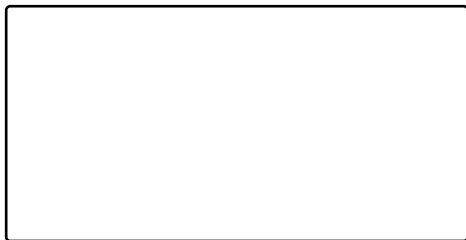
(0, 0): 0  
(1, 2): 0.357  
(2, 3): 0.571  
(3, 1): 0.071  
(4, 1): 0

- (iv) Which data point is mostly likely chosen as the center for cluster 3?



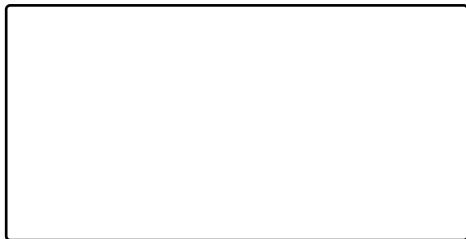
(2, 3) is mostly likely chosen.

- (v) Assume the center for cluster 3 is also chosen to be the most likely one as you computed in the previous question. Now we finish the initialization for all 3 centers. List the data points that are classified into cluster 1, 2, 3 respectively.



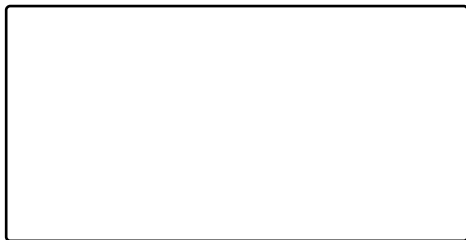
cluster 1: (0, 0)  
cluster 2: (1, 2), (2, 3)  
cluster 3: (3, 1), (4, 1)

- (vi) Based on the above clustering result, what's the new center for every cluster?



center for cluster 1: (0, 0)  
center for cluster 2: (1.5, 2.5)  
center for cluster 3: (3.5, 1)

- (vii) According to the result of (ii) and (iv), explain how does k-means++ alleviate the local optimum issue due to initialization?



k-means++ tends to initialize new cluster centers with the data points that are far

away from the existing centers, to make sure all of the initial cluster centers stay away from each other.

3. Consider a dataset with seven points  $\{x_1, \dots, x_7\}$ . Given below are the distances between all pairs of points.

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$
$x_1$	0	5	3	1	6	2	3
$x_2$	5	0	4	6	1	7	8
$x_3$	3	4	0	4	3	5	6
$x_4$	1	6	4	0	7	1	2
$x_5$	6	1	3	7	0	8	9
$x_6$	2	7	5	1	8	0	1
$x_7$	3	8	6	2	9	1	0

Assume that  $k = 2$ , and the cluster centers are initialized to  $x_3$  and  $x_6$ . Which of the following shows the two clusters formed at the end of the first iteration of  $k$ -means? Circle the correct option.

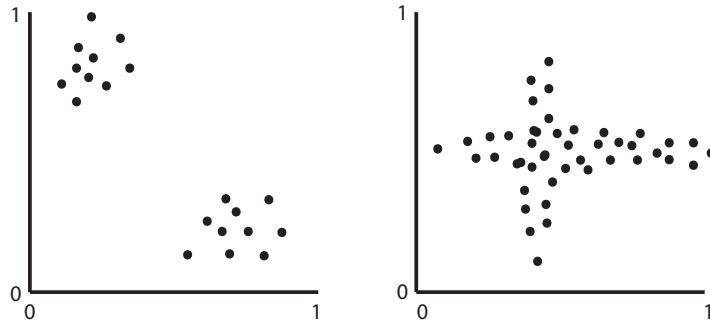
- $\{x_1, x_2, x_3, x_4\}, \{x_5, x_6, x_7\}$
- $\{x_2, x_3, x_5\}, \{x_1, x_4, x_6, x_7\}$
- $\{x_1, x_2, x_3, x_5\}, \{x_4, x_6, x_7\}$
- $\{x_2, x_3, x_4, x_7\}, \{x_1, x_5, x_6\}$

**Solution:** (b).

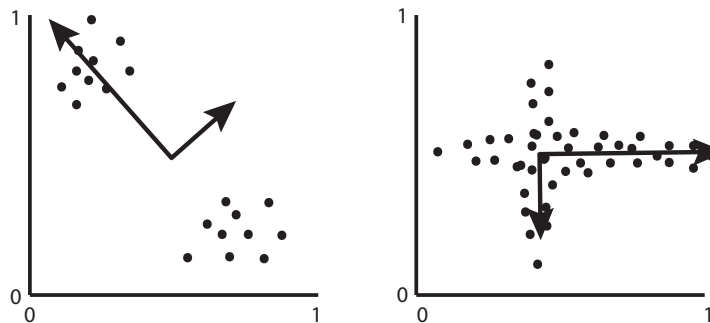


## 7 Principal Component Analysis

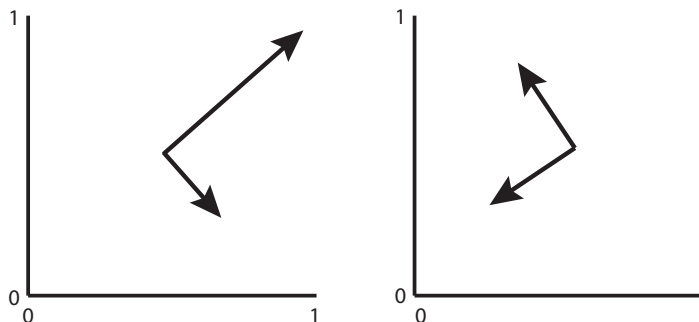
1. (i) Consider the following two plots of data. Draw arrows from the mean of the data to denote the direction and relative magnitudes of the principal components.



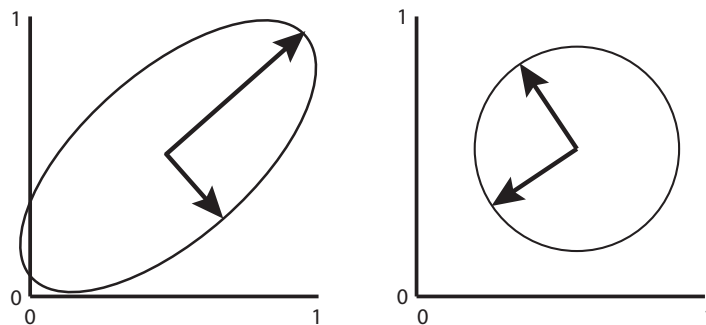
**Solution:**



- (ii) Now consider the following two plots, where we have drawn only the principal components. Draw the data ellipse or place data points that could yield the given principal components for each plot. Note that for the right hand plot, the principal components are of equal magnitude.



**Solution:**



2. Circle one answer and explain.

In the following two questions, assume that using PCA we factorize  $X \in \mathbb{R}^{n \times m}$  as  $Z^T U \approx X$ , for  $Z \in \mathbb{R}^{m \times n}$  and  $U \in \mathbb{R}^{m \times m}$ , where the rows of  $X$  contain the data points, the rows of  $U$  are the prototypes/principal components, and  $Z^T U = \hat{X}$ .

- (i) Removing the last row of  $U$  and  $Z$  will still result in an approximation of  $X$ , but this will never be a better approximation than  $\hat{X}$ .

**Circle one:**      True      False

True. As we are removing a principal component of the data when we remove any row from  $U$  and  $Z$ , we take the variance attributed to that principal component with it. Since variance is always nonnegative, removing some of the variance preserved by a given principal component will increase the reconstruction error of the original data (recall that maximizing the variance preserved is equivalent to minimizing the reconstruction error).

- (ii)  $\hat{X} \hat{X}^T = Z^T Z$ .

**Circle one:**      True      False

True.  $\hat{X} \hat{X}^T = Z^T U (Z^T U)^T = Z^T U U^T Z = Z^T Z$ . Recall that the rows of  $U$  are eigenvectors, meaning  $U^T U$  has non-zero entries in the main diagonal only. Further, the principal components themselves are unit vectors, meaning the dot product of any eigenvector in  $U$  with itself is one. Then the main diagonal of  $U^T U$  is all ones. Thus,  $U^T U$  is the identity matrix.

- (iii) The goal of PCA is to interpret the underlying structure of the data in terms of the principal components that are best at predicting the output variable.

**Circle one:**      True      False

False. The goal of PCA is to produce an underlying structure to the data that preserves the largest amount of variance (or synonymously minimizes the reconstruction error). While performing PCA, the output variable is never provided.

- (iv) The output of PCA is a new representation of the data that is always of lower dimensionality than the original feature representation.

**Circle one:**      True      False

False. PCA can produce a representation that is up to the same number of dimensions as the original feature representation.