Name: ___________________________________ NetID: _____________

S&DS 365 / 665

Intermediate Machine Learning

Final Exam

Saturday, May 7, 2022

Complete all of the problems. You have 2.5 hours (150 minutes) to complete the exam.

The exam is closed book, computer, phone, etc. You are allowed one double-sided $8\frac{1}{2} \times 11$ sheet of paper with hand-written notes.

1. ***Multinomial choice*** (10 points)

For each of the following questions, circle the *single best* answer.

1.1. Suppose that we fit a Gaussian process regression model using training data X and Y with noise level σ^2, and we use the model to for a new set X'. Let that $\mathbb{K}_{ij} = K(X_i, X_j)$ and $\mathbb{K}'_{ij} = K(X'_i, X_j)$. Then the posterior mean of the regression function $\widehat{m}(X')$ is given by

(a) $\mathbb{K}(\mathbb{K} + \sigma^2 I)^{-1} Y$
(b) $\mathbb{K}(\mathbb{K}' + \sigma^2 I)^{-1} Y$
$\boxed{\text{(c)}}$ $\mathbb{K}'(\mathbb{K} + \sigma^2 I)^{-1} Y$
(d) None of the above

1.2. Consider the "toy" lasso problem

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{2}(Y - \beta)^2 + |\beta|$$

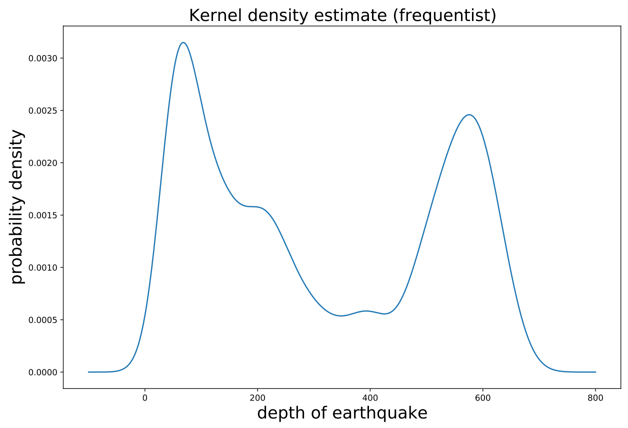
where Y is a random variable and β is a scalar. If $Y = -2$ the solution is

(a) $\widehat{\beta} = 0$
$\boxed{\text{(b)}}$ $\widehat{\beta} = -1$
(c) $\widehat{\beta} = -3$
(d) $\widehat{\beta} = 1$

1.3. Suppose that we have a kernel regression technique in one dimension with bandwidth parameter h for which the squared bias scales as $O(h^3)$ and the variance scales as $O\left(\frac{1}{nh}\right)$ as $h \to 0$ with $nh \to \infty$, for a sample of size n, under certain assumptions. What is the fastest rate at which the risk (expected squared error) will decrease with sample size for this technique?

(a) $O(n^{-1/3})$
(b) $O(n^{-1/4})$
$\boxed{\text{(c)}}$ $O(n^{-3/4})$
(d) $O(n^{-1})$

1.4. Recall that the kernel density estimate for the Fiji earthquake data looks like:

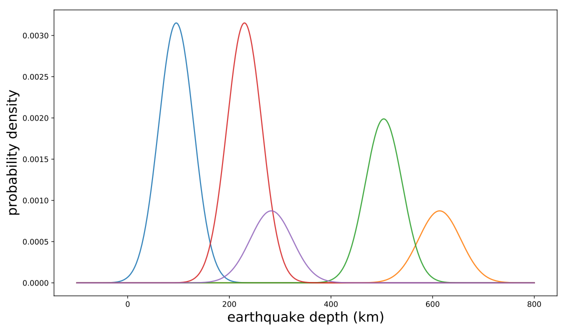


Our nonparametric Bayes model is a Dirichlet process mixture. Specifically, we take $X_i \mid \theta_i \sim N(\theta_i, \sigma^2)$ with $\sigma = 30$ fixed, and use a Dirichlet process prior $DP(\alpha, F_0)$ for the means θ_i, where $F_0 = N(\mu_0, \tau_0)$. We set $\mu_0 = 300$ and $\tau_0 = 100$.

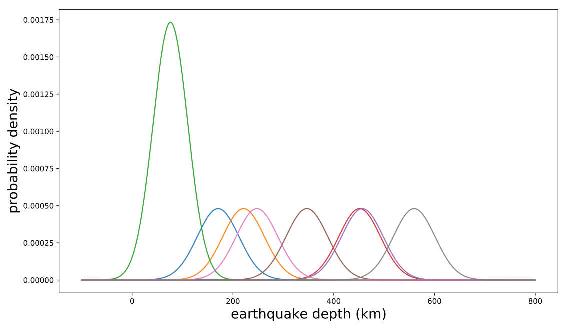
Gibbs sampling was separately run for all six combinations of $\alpha \in \{1, 10, 100\}$ and sample sizes $n \in \{10, 100\}$, where n earthquakes were randomly sampled from the data. The plots below show the posterior Gaussian distribution for each occupied "table" (cluster) at one stage of the Gibbs sampling algorithm.

Which of the plots was most likely generated with $\alpha = 10$ and $n = 100$?

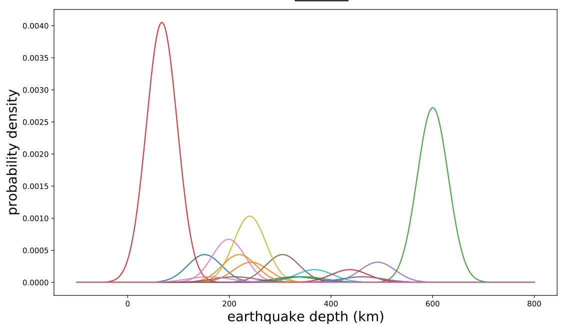
(a)



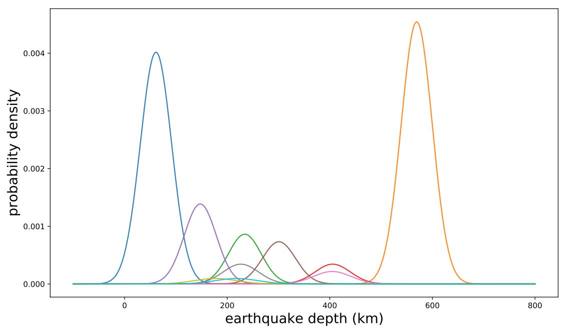
(b)



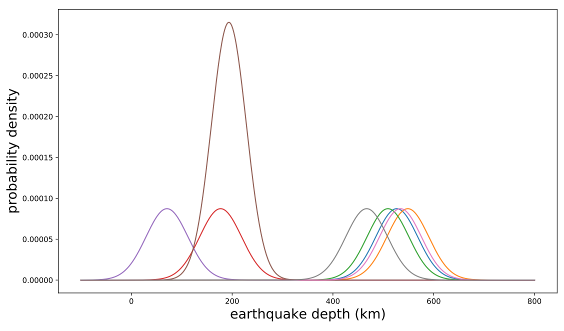
$\boxed{\text{(c)}}$



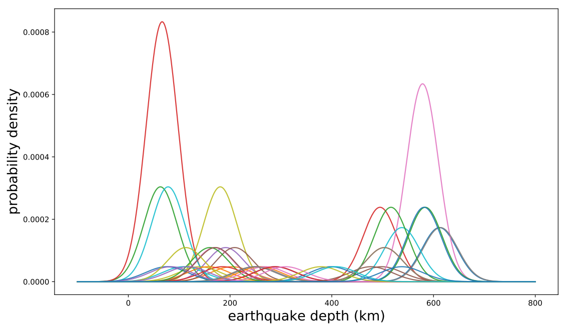
(d)



(e)



(f)



1.5. Suppose F is drawn from a Dirichlet process $DP(\alpha, F_0)$ where $\alpha = \frac{1}{3}$ and F_0 is a discrete uniform distribution on the 6 values $\{1, 2, 3, 4, 5, 6\}$, with equal probability $\frac{1}{6}$ for each (a fair 6-sided die).

We draw three values $X_1 = 1$, $X_2 = 3$, $X_3 = 1$ from F. What is the probability that the next sample is $X_4 = 1$?

(a) $1/60$

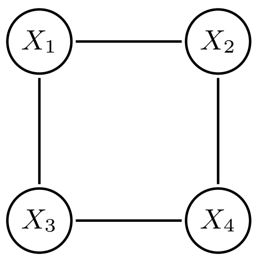
(b) $1/10$

[✓ (c)] $37/60$

(d) $31/60$

(e) None of the above

1.6 Consider a graph neural network for binary classification of inputs $x = (x_1, x_2, x_3, x_4)^T$ on the graph below:

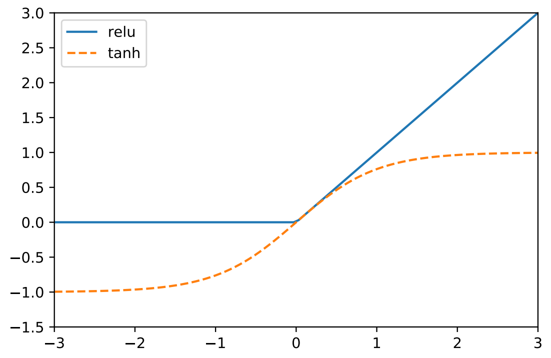


The graph neural network defines a classifier with discriminant function

$$\log\left(\frac{p(Y=1\,|\,x)}{p(Y=0\,|\,x)}\right) = \beta^T h(x)$$

with $h(x) = \varphi(Lx)$ where φ is an activation function and L is the graph Laplacian (with edge weights 1), and $\beta = (1, -1, 1, -1)^T$.

Recall the relu and tanh activation functions look like this, with $\tanh(-x) = -\tanh(x)$:

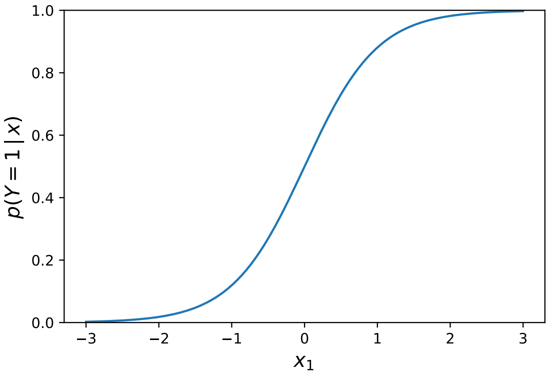


[✓ TRUE] FALSE (1) If φ is relu and $x = (1, -1, 1, -1)$ then $p(Y = 1\,|\,x) > \frac{1}{2}$.

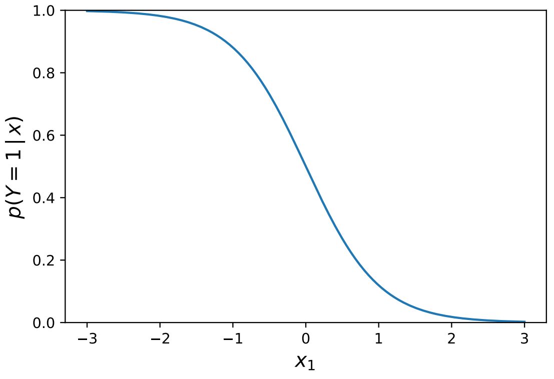
TRUE [✓ FALSE] (2) If φ is tanh and $x = (1, 2, 3, 4)$ then $p(Y = 1\,|\,x) > \frac{1}{2}$.

(3) If φ is relu and $x = (x_1, 0, 0, 0)^T$, which of the following is a plot of the probability $p(Y = 1\,|\,x)$ as a function of x_1?

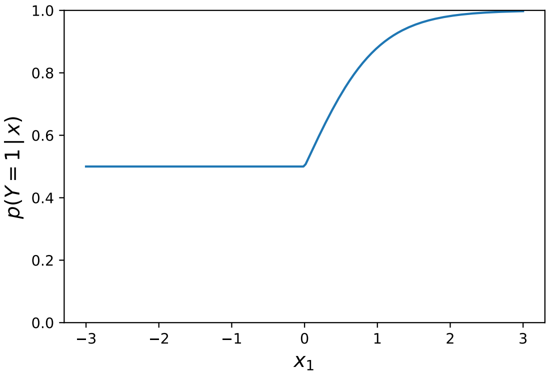
(a)



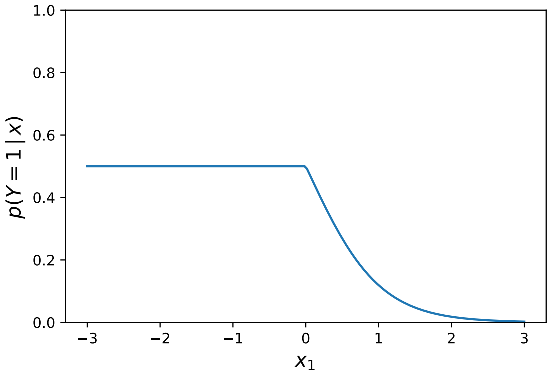
(b)



$\boxed{\text{(c)}}$



(d)



1.7 The following questions concern recurrent neural networks (RNNs) and gated recurrent units (GRUs).

[✓ TRUE] FALSE (1) GRUs are a practical solution to the "vanishing gradient problem" in vanilla RNNs that prevents the states from remembering important information in the past.

[✓ TRUE] FALSE (2) RNNs can be used as language models to predict the next word or sequence in a corpus of text.

TRUE [✓ FALSE] (3) GRUs cannot properly be used as language models because the memory cells violate the Markov property.

[✓ TRUE] FALSE (4) GRUs can be trained with backpropagation.

TRUE [✓ FALSE] (5) In vanilla RNNs the states are deterministic functions of the past input sequence; GRUs introduce a stochastic memory component.

2. ***Convolutional neural networks*** (10 points)

   During the class we've often used convolutional neural networks (CNNs). The following familiar tensorflow code constructs a CNN to classify 64×64 color images, using two convolutional layers followed by a dense layer.

```
from tensorflow.keras import layers, models

model = models.Sequential()
model.add(layers.Conv2D(10, (5, 5), input_shape=(64, 64, 3)))
model.add(layers.MaxPooling2D((4, 4)))
model.add(layers.Conv2D(20, (6, 6)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(rate=.5))
model.add(layers.Flatten())
model.add(layers.Dense(2))
```

Indicate the shape of the output tensor for each layer, together with the number of trainable parameters, by filling in the two missing fields for each of the rows below. For partial credit if an answer is wrong, you may show your work below the table. No calculators.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 60, 60, 10)        760
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 10)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 20)        7220
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 20)          0
_________________________________________________________________
dropout (Dropout)            (None, 5, 5, 20)          0
_________________________________________________________________
flatten (Flatten)            (None, 500)               0
_________________________________________________________________
dense (Dense)                (None, 2)                 1002
=================================================================
```

3. ***Short answer*** (10 points)

   The following two subproblems ask you to explain important concepts associated with two topics in the course.

   (a) ***Actor-critic reinforcement learning***. (1) What is an actor-critic approach to reinforcement learning? (2) How is it different from Q-learning? (3) How might neural networks be used in an actor-critic approach?

   (1) Actor-critic approaches are on-policy RL methods that simultaneously estimate a value function (the critic) and a policy (the actor)

   (2) Q-learning is off-policy in that only the Q-function is learned, not an explicit policy.

   (3) A neural network can be used to take states as inputs and output both the value and action probabilities. The TD errors drive the learning; see the lecture slides for details.

(b) ***Attention***. (1) Describe the high level idea behind attention in sequence-to-sequence models. (2) What is the main problem that attention mechanisms are designed to address? (3) Give an example of how attention is used.

(1) Attention weights the different states in the input sequence differently depending on which output sequence term is being generated.

(2) They are meant to address the problem that without attention all information about the input sequence is put into a fixed state vector.

(3) In class we discussed examples in machine translation. When, for example, the verb in the target sentence is generated, the attention mechanism will point to the verb in the input sequence. See the slides for more details.

4. ***The graphical lasso*** (10 points)

(a) Explain the purpose of the graphical lasso algorithm.

To estimate the undirected graph of a Gaussian graphical model, which equivalent to the sparsity pattern of the precision matrix.

(b) Give the objective function that the graphical lasso uses. Give a brief derivation of the objective function, stating the assumptions that are made.

The objective function is

$$\mathcal{O}(\Omega) = \text{trace}(S\Omega) - \log|\Omega| + \lambda \sum_{j \neq k} |\Omega_{jk}|$$

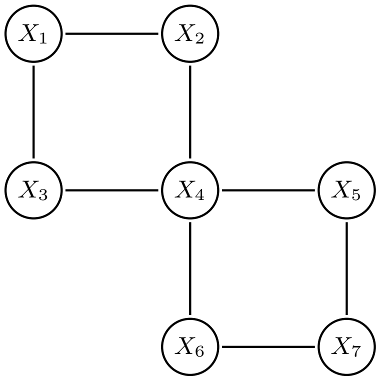
which is an ℓ_1 penalty combined with $-\frac{1}{n}\sum_i \log p(x_i)$ using

$$-\log p(x) = \frac{1}{2}\log|\Sigma| + \frac{1}{2}\text{trace}(\Omega x x^T) = -\frac{1}{2}\log|\Omega| + \frac{1}{2}\text{trace}(\Omega x x^T)$$

Suppose that the graphical lasso estimates the following matrix for seven variables $X_1, \ldots, X_7$:

$$\begin{array}{c} \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 \end{array} \\ \begin{bmatrix} 3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 3 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 5 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 3 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 3 \end{bmatrix} \end{array}$$

(c) Draw the corresponding graph.



(d) Based on this graph, find a minimal set of variables S for which

$$X_1 \perp\!\!\!\perp X_7 \quad | \ S$$

$$X_1 \perp\!\!\!\perp X_7 \quad | \ X_4$$

5. ***Coding: Implement a prediction function*** (10 points)

   In this problem you are asked to implement a prediction function in Python. Try to make your code as syntactically correct as possible; partial credit will be given when it is not.

   A simple implementation of a 2-layer neural network using only `numpy` has a function `train_2_layer_network` that starts and ends like this:

```
def train_2_layer_network(X, y, H=100):
    # initialize parameters randomly
    # H = size of hidden layer
    # K = number of classes
    # D = input dimension

    K = len(set(y))
    D = X.shape[1]
    W = np.random.randn(H, D)
    b = np.zeros((H, 1))
    beta = np.random.randn(H,K)
    beta0 = np.zeros((K,1)) # intercepts for the last layer

    # some hyperparameters
    step_size = 1e-1

    # batch gradient descent loop
    num_examples = X.shape[0]
    for i in range(20000):

      # Implementation of backpropagation would follow here
      # It's skipped for brevity ... You don't need to write it!
      ...

      # perform a parameter update
      W += -step_size * dW
      b += -step_size * db
      beta += -step_size * dbeta
      beta0 += -step_size * dbeta0

  return W, b, beta, beta0
```

Suppose that we have a call to this function:

```
W, b, beta, beta0 = train_2_layer_network(X, y)
```

(a) Write a function `predict(X, W, b, beta, beta0)` that takes a matrix of inputs `X` and makes predictions for the class labels. Your function should return an array of length `X.shape[0]`, the number of rows of `X`. You should assume that the rectified linear activation was used during training.

```
def predict(X, W, b, beta, beta0):
  hidden_layer = np.maximum(0, np.dot(X, W.T) + b.T)
  scores = np.dot(hidden_layer, beta) + beta0.T
  return np.argmax(scores, axis=1)
```

```
def predict(X, W, b, beta, beta0):
  hidden_layer = np.maximum(0, np.dot(W, X.T) + b)
  scores = np.dot(beta.T, hidden_layer) + beta0
  return np.argmax(scores, axis=0)
```

(b) Now write code that takes test data `X` and `y`, computes the predicted labels for X by calling the function from (a), and then prints out the error rate of the predictions. The error rate should be expressed as a percentage between 0% and 100%.

```
yhat = predict(X, W, b, beta, beta0)
error_rate = np.mean(yhat != y)
print('Error rate: %.2f%%' % (error_rate*100))
```

6. ***Reinforcement learning*** (10 points)

The following three subproblems are on the topic of reinforcement learning.

(1) Give the Bellman equations for the value function and the Q-function for a deterministic environment. Show that a Q-function that satisfies the Bellman equations defines a value function that also satisfies Bellman's equations.

$$\begin{aligned} v_*(s) &= \max_a \{\text{reward}(s,a) + \gamma v_*(\text{next}(s,a))\} \\ Q_*(s,a) &= \text{reward}(s,a) + \gamma \max_{a'} Q_*(\text{next}(s,a),a') \end{aligned}$$

Suppose $Q_*(s,a)$ satisfies the Bellman equation and define $v_*(s) = \max_a Q_*(s,a)$ then

$$\begin{aligned} v_*(s) &= \max_a Q_*(s,a) \\ &= \max_a \left\{\text{reward}(s,a) + \gamma \max_{a'} Q_*(\text{next}(s,a),a')\right\} \\ &= \max_a \left\{\text{reward}(s,a) + \gamma v_*(\text{next}(s,a))\right\} \end{aligned}$$

(2) Suppose that an environment has three states, $s = 0$, $s = 1$, and $s = 2$, and an agent can take two actions $a = 0$ and $a = 1$. The environment is deterministic and has the following `reward` and `next` functions:

$$\texttt{next}(s,a): \begin{array}{c|cc} & a{=}0 & a{=}1 \\ \hline s{=}0 & 1 & 1 \\ s{=}1 & 2 & 2 \\ s{=}2 & 0 & 0 \end{array} \qquad \texttt{reward}(s,a): \begin{array}{c|cc} & a{=}0 & a{=}1 \\ \hline s{=}0 & 0 & 1 \\ s{=}1 & 1 & 0 \\ s{=}2 & 0 & 1 \end{array}$$

Give an optimal Q-function $Q^*(s,a)$ using discount $\gamma = \frac{2}{3}$. Write your answer in the following table. Show your work below for partial credit.

$$Q^*(s,a): \begin{array}{c|cc} & a{=}0 & a{=}1 \\ \hline s{=}0 & 2 & 3 \\ s{=}1 & 3 & 2 \\ s{=}2 & 2 & 3 \end{array}$$

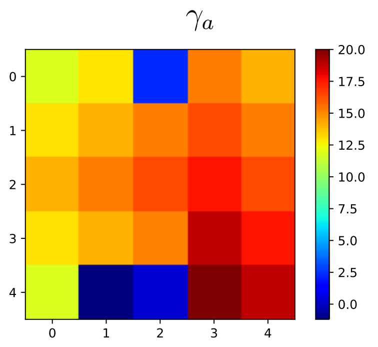
Recall the Taxi problem discussed in class. A taxi navigates a 5×5 grid, picking up passengers and delivering them to their desired destinations. There is a default per-step reward of -1, and a reward of $+20$ for delivering the passenger. An illegal pickup or drop-off action has a reward of -10.

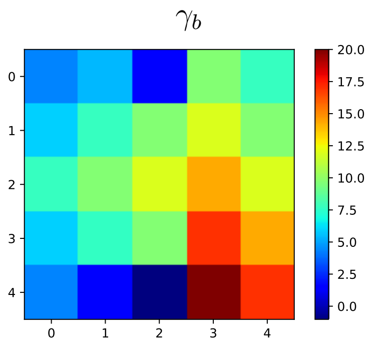
The state can be represented as a tuple

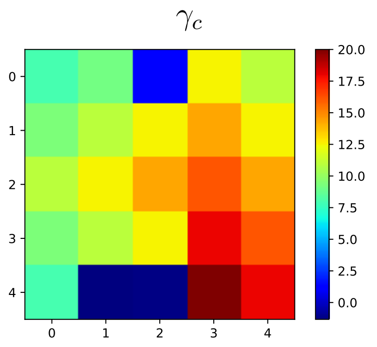
$$\texttt{(row, col, passenger\_location, destination).}$$

The passenger is either waiting to be picked up at one of four locations, or is in the taxi being driven (hopefully) to their desired destination. So, for a fixed passenger location and destination, the value function $v(\texttt{row}, \texttt{col})$ assigns a value to each of the $25 = 5 \times 5$ grid points.

The following three figures show the value function where the passenger is either waiting to be picked up at a specific location, or is in the taxi. Each plot corresponds to running the Q-learning algorithm for a *different* discount factor γ.







(3) Order the discount factors. For example, write $\gamma_c < \gamma_b < \gamma_a$ if γ_a was the largest and γ_c was the smallest discount factor. Briefly explain your answer.

$$\gamma_b < \gamma_c < \gamma_a$$

In the taxi problem, the only positive reward is at the end of an episode when the passenger is dropped off at the correct location. A higher discount factor γ means more weight is put on this future reward and thus leads to higher values for the destination or the surrounding pixels.

Extra work space

Extra work space

Thank you for being part of IML this semester!
Have a wonderful summer!