6.036: Final Exam, Spring 2019

Do not tear exam booklet apart!

- This is a closed book exam. Two sheets of paper (8 1/2 in. by 11 in.) of notes, front and back, are permitted. Calculators are not permitted.
- The problems are not necessarily in any order of difficulty.
- Record all your answers in the places provided. If you run out of room for an answer, continue on a blank page and mark it clearly.
- If a question seems vague or under-specified to you, make an assumption, write it down, and solve the problem given your assumption.
- If you absolutely *have* to ask a question, come to the front.
- **Write your name on every page.**

Name: ______________________ Athena ID: ______________________
(username)

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	8	8	12	10	16	14	14	8	10	100
Score:										

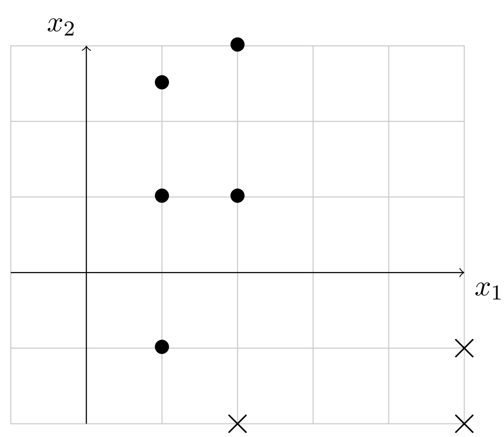
Name: ______________________

Nearest Neighbors

1. (8 points) Consider the following 2D dataset:

x	y
$(1,-1)$	$+1$
$(1,1)$	$+1$
$(1,2.5)$	$+1$
$(2,-2)$	-1
$(2,1)$	$+1$
$(2,3)$	$+1$
$(5,-1)$	-1
$(5,-2)$	-1

The dataset is plotted below, with positively labeled points as solid points (•) and negatively labeled points as X marks (×):



Break ties in distance by choosing the point with smaller x_1 coordinate, and if still tied, by smaller x_2 coordinate.

(a) Compute the leave-one-out cross validation accuracy (i.e., average 8-fold cross validation accuracy) of the 1-nearest-neighbor learning algorithm on this dataset.

(b) Compute the leave-one-out cross validation accuracy of the 3-nearest-neighbor learning algorithm on this dataset.

(c) In the case of the 1-nearest-neighbor learning algorithm, is it possible to strictly increase the leave-one-out cross validation accuracy on this dataset by changing the label of a single point in the original dataset? If so, give such a point.

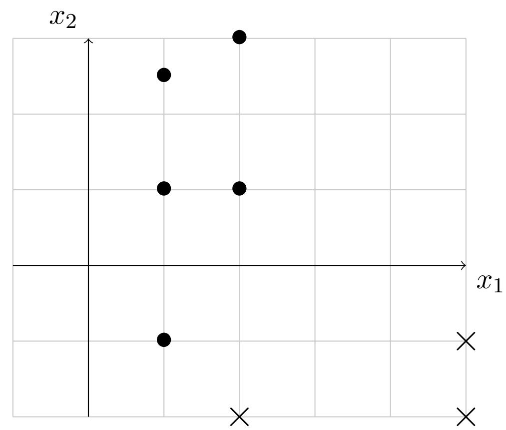
(d) How about in the case of the 3-nearest neighbor algorithm? If so, give such a point.

Name: ______________________________

Decision Trees

2. (8 points) Consider the following 2D dataset (same as that used in the previous question). Positively labeled points are solid points (•) and negatively labeled points are X marks (×).

x	y
$(1, -1)$	+1
$(1, 1)$	+1
$(1, 2.5)$	+1
$(2, -2)$	-1
$(2, 1)$	+1
$(2, 3)$	+1
$(5, -1)$	-1
$(5, -2)$	-1



We will construct a tree using the algorithm discussed in the lecture notes, i.e., a greedy algorithm that recursively minimizes weighted average entropy. Recall that the weighted average entropy of a split into subsets A and B is:

$$(\text{fraction of points in } A) \cdot H(R^A_{j,s}) + (\text{fraction of points in } B) \cdot H(R^B_{j,s})$$

where the entropy $H(R_m)$ of data in a region R_m is given by

$$H(R_m) = -\sum_k \hat{P}_{mk} \log_2 \hat{P}_{mk}.$$

The $\hat{P}_{mk}$ is the *empirical probability*, which is in this case the fraction of items in region m that are of class k.

Some facts that might be useful to you:

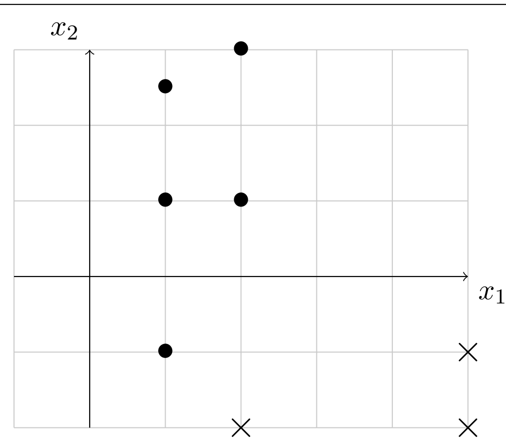
$$H(0) = 0$$
$$H(3/5) = 0.97$$
$$H(3/8) = 0.95$$
$$H(3/4) = 0.81$$
$$H(5/6) = 0.65$$
$$H(1) = 0$$

Name: ____________________

(a) Draw the decision tree that would be constructed by our tree algorithm for this dataset. Clearly label the test in each node, which case (yes or no) each branch corresponds to, and the prediction that will be made at each leaf. Assume there is no pruning and that the algorithm runs until each leaf has only members of a single class.

(b) Draw the decision tree boundaries represented by your decision tree on the data plot figure below.

x	y
$(1,-1)$	+1
$(1,1)$	+1
$(1,2.5)$	+1
$(2,-2)$	-1
$(2,1)$	+1
$(2,3)$	+1
$(5,-1)$	-1
$(5,-2)$	-1



(c) What class does your decision tree (above) predict for the new point: (1, -2)?

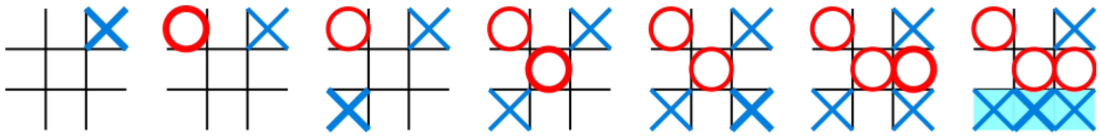
(d) Decision trees built using our greedy algorithm are a good choice of classifiers for images.
◯ T ◯ F Explain.

(e) For decision trees built using our greedy algorithm, standardizing feature values is important.
◯ T ◯ F Explain.

(f) A disadvantage of using decision trees for classification is that they can only be used to classify data having two classes.
◯ T ◯ F Explain.

Tic-Tac-Toe Revised

3. (12 points) Tic-tac-toe is a paper-and-pencil game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks sequentially in a horizontal, vertical, or diagonal row wins the game. The following example game is won by the first player, X:



In this question, we'll consider a "solitaire" version of tic-tac-toe, in which we assume:

- We are the X player;
- The O player is a fixed (but possibly stochastic) algorithm;
- The initial state of the board is empty, and X has the first move;
- We can select any of the nine squares on our turn;
- We don't know the strategy of the O player or the reward function used by O.

We place an X in an empty square, then an O appears in some other square, and then it's our turn to play again. We receive a +1 reward for getting three X's in a row, reward -1 if there are three O's in a row, and reward 0 otherwise. If we select a square that already has an X or an O in it, nothing changes and it's still our turn.

(a) We can model this problem as a Markov decision process in several different ways. Here are some possible choices for the state space.

- Jody suggests letting the state space be all possible 3 x 3 grids in which each square contains one of the following: a space, an O, and an X.
- Dana suggests using all possible 3 x 3 grids in which each square contains one of the three options (a space, an O, and an X), and there is an equal number of O's and X's.
- Chris suggests using all 3 x 3 tic-tac-toe game grids which appear in games where the players both employ optimal strategies.

i. Is Dana's suggestion better or worse for tabular Q learning than Jody's? Explain your answer.

ii. Is Chris' suggestion better or worse for tabular Q learning than Jody's? Explain your answer.

(b) Many states of the game are effectively the same due to symmetry.

i. Draw a pair of such states which are the same due to symmetry:

ii. Jordan suggests using a state-space that includes one state that stands for each set of board games that are equivalent due to symmetry. Would this be better or worse for learning than Jody's representation? Explain your answer.

(c) What is the action space of the MDP with Dana's state space definition?

(d) You get to sit and watch an expert player (who always makes optimal moves) play this game for a long time, and you observe the sequence of state-action pairs that occur in many games. Which of the following machine-learning problem formulations is most appropriate, for you to learn how to play the game? For the item you select, provide the specified additional information (where not "none").

1. supervised regression (describe the loss function)
2. supervised classification (describe the loss function)
3. reinforcement learning of a policy (none)
4. reinforcement learning of a value function (none)

Explain your answer.

(e) You get to interact with an implementation of this game for many game instances, selecting your actions, observing the results and rewards. Which of the following machine-learning problem formulations is most appropriate, for you to learn how to play the game? For the item you select, provide the specified additional information (where not "none").

1. supervised regression (describe the loss function)
2. supervised classification (describe the loss function)
3. reinforcement learning of a policy (none)
4. reinforcement learning of a value function (none)

Explain your answer.

(f) Barney wants to solve a tic-tac-toe problem that is exactly the same as the above game (i.e., three in a row/column/diagonal wins), except that it is played on a 100 x 100 grid.

i. Is it better for Barney to use tabular Q learning or neural-net Q learning? Explain.

ii. Suppose Barney were to use neural-net Q learning; would it help for him to start with a convolutional layer? If your answer is yes, describe four 3 x 3 convolutional filters that would be particularly helpful for this problem.

(g) Suppose you apply Q-learning to the 3x3 tic-tac-toe problem, and your actions always select an unfilled square. Bert suggests that it is okay to let the discount factor be 1. Is that true? Explain why or why not.

CNN Backpropagation

4. (10 points) Conne von Lucien has many pictures from her trip to Flatland and wants to determine which ones have her in the image. All of the pictures are arrays of size 4x1, with array values of either 0 or 1. Conne looks like the vector [1, 0, 1] in one dimension, so if a picture contains the pattern [1, 0, 1] anywhere inside it, it should be classified as a positive example, otherwise as a negative example.

   Fortunately, you learned about CNNs and have helped Conne by designing the following network architecture with three layers:

   1. A convolutional layer with one filter W that is size 3x1, and stride 1, and a single bias w_0 (where the output pixel corresponds to the input pixel that the filter is centered on). Input values of 0 should be assumed beyond the boundaries of the input.
   2. A max-pooling layer P with size 2x1 and stride 2.
   3. A fully connected layer $\sigma(\cdot)$ with a single output unit having a sigmoidal activation function.

   (a) What is the shape of the output of each layer?

   i. Layer 1:

   ii. Layer 2:

   iii. Layer 3:

   (b) What loss function is most appropriate here, especially if you want your neural network package to be useful with few modifications, to other Flatland visitors (who may appear as longer vectors)?

   ◯ A. NLL loss

   ◯ B. Hinge loss

   ◯ C. Quadratic loss

(c) We can express the loss function as $L(\sigma(P), y)$ where P is the output from the max pooling layer of the CNN and y is the true label for the input. Given $\frac{dL}{dP}$, derive the update rule for w_1 if the filter is composed of $W = [w_1, w_2, w_3]^T$ with bias w_0, and step size is η.

(d) Given $\frac{dL}{dP}$, provide the update rule for w_0, the bias to the filter.

(e) Conne decides to use the neural network code as written by a 6.036 student for the 6.036 homework (and that actually was a correct implementation) to train her CNN using SGD. The `sgd` procedure may be called multiple times from elsewhere (e.g., to implement multiple epochs of SGD). Conne thinks she has a better `sgd` python procedure than that given in the package; her code is:

```
1  def sgd(nn, X, Y, iters=100, lrate=0.005):
2      D, N = X.shape
3      sum_loss = 0
4      for k in range(iters):
5          Xt = X[:, k:k+1]
6          Yt = Y[:, k:k+1]
7          Ypred = nn.forward(Xt)
8          sum_loss += nn.loss.forward(Ypred, Yt)
9          err = nn.loss.backward()
10         nn.backward(err)
11         nn.sgd_step(lrate)
```

Here, `nn` is an instance of the `Sequential` class implementing the CNN. She knows from the unit tests that the `nn` routines function properly. In particular, `nn.forward` properly computes the predicted outputs `Ypred` from input data `Xt`, `nn.loss.forward` also properly computes the forward loss, `nn.loss.backward` properly computes the backward loss, `nn.backward` properly computes the backward gradients, and `nn.sgd_step` properly applies an SGD update step with the specified learning rate `lrate`. And the N sets of dimension D input data `X`, and labels `Y` are known to be correct.

However, Conne's procedure consistently gives poor results (and occasionally throws errors), compared with the 6.036 student's correct SGD routine, when run with identical arguments.

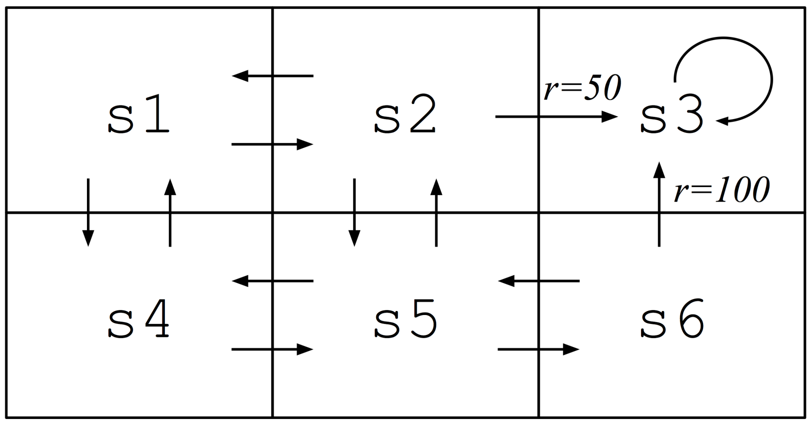
Why? Specify the line(s) which have errors, and describe how the code should be improved to do as well as the correct implementation of the 6.036 student:

Lines with errors (give linenumbers):

Improved code lines:

Robots

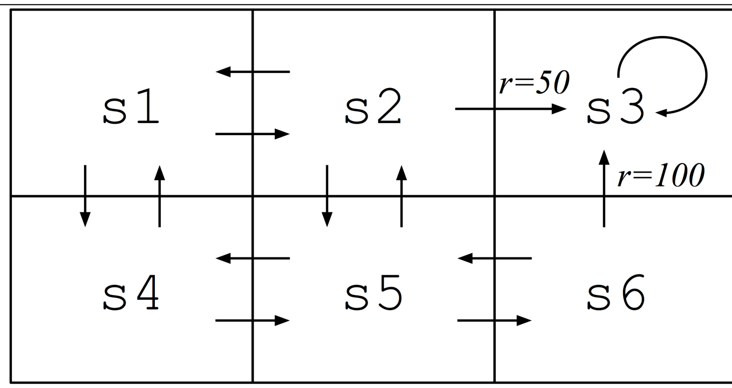
5. (16 points) Consider the following deterministic Markov Decision Process (MDP), describing a simple robot grid world. Notice that the values of the immediate rewards r for **two** transitions are written next to them; the other transitions, with no value written next to them, have an immediate reward of $r = 0$. **Assume the discount factor γ is 0.8.**



(a) For states $s \in \{s6, s5, s2\}$, write the value for $V_{\pi^*}(s)$, the discounted infinite horizon value of state s using an optimal policy π^*. It is fine to write a numerical expression—you don't have to evaluate it—but it shouldn't contain any variables.

i. $V_{\pi^*}(s6) =$
ii. $V_{\pi^*}(s5) =$
iii. $V_{\pi^*}(s2) =$

(b) For each state in the state diagram below, circle exactly one outgoing arrow, indicating an optimal action $\pi^*(s)$ to take from that state. If there is a tie, it is fine to select any action with optimal value.



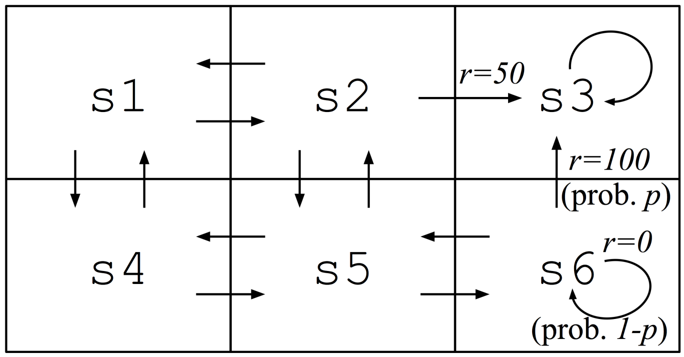
(c) Give a value for γ (constrained by $0 < \gamma < 1$) that results in a different optimal policy, and describe the resulting policy by indicating which $\pi^*(s)$ values (i.e., which policy actions) change.

New value for γ:

Changed policy action:

(d) Is it possible to change the immediate reward for each state in such a way that V_{π^*} changes but the optimal policy π^* remains unchanged? If yes, provide a new reward function, and explain how the resulting V_{π^*} changes but π^* does not. Otherwise, explain in at most two sentences why this is impossible.

When winter comes, snow also appears on one path in the grid world, making exactly one of the actions non-deterministic. The resulting MDP is shown below. Specifically, the change is that now the result of the action "go north" from state `s6` results in one of two outcomes. With probability p, the robot succeeds in transitioning to state `s3` and receives immediate reward 100. However, with probability $(1-p)$ it slips on the ice, and remains in state `s6` with 0 immediate reward. **Assume again that the discount factor** $\gamma = 0.8$.



(e) Assume $p = 0.75$. For each of the states $s \in \{s2, s5, s6\}$, write the value for $V_{\pi^*}(s)$. It is fine to write a numerical expression, but it shouldn't contain any variables.

i. $V_{\pi^*}(s6) =$

ii. $V_{\pi^*}(s5) =$

iii. $V_{\pi^*}(s2) =$

(f) How bad does the ice have to get before the robot will prefer to completely avoid the ice? Let us answer the question by giving a value for p for which the optimal policy chooses actions that completely avoid the ice, i.e., choosing the action "go left" over "go up" when the robot is in the state `s6`. Approach this in four parts. The answer to each of the first three parts can be a numerical expression; the answer to the last part can be an expression involving numbers and p.

i. What is the value V of going right in state `s2`?

ii. What is the value V of going up in state `s5`, if you're going to go right in state `s2`?

iii. What is the value V of going left in state `s6`, if you're going to go up in state `s5` and right in state `s2`?

iv. Under what condition on p is it better to go left in state `s6` (then up in state `s5` and right in state `s2`) than it is to go up in state `s6`?

Recommender System

6. (14 points) After taking 6.036, Bob decides to train a recommender system to predict what ratings different customers will give to different movies. Currently, he knows of three really popular movies, and he knows of two potential customers who have ranked some of these movies. The data matrix currently looks like: $Y = [[2, ?, 3], [4, 2, ?]]$ where, as in class, rows correspond to customers and columns correspond to movies, and ? indicates a missing or unknown ranking. He decides to find a low rank factorization of Y using the alternating least squares algorithm implemented in class. Assume for this question that offsets are set to 0.

   (a) Bob starts out by trying a rank 1 factorization of Y as UV^T. He initializes $U = [1, 2]^T$. Assume there is no regularization. In the first iteration of alternating least squares, we will find the best V given the current U. What is the objective function $J(V)$ in terms of V? Write it in terms of V_1, V_2, V_3 and specific numerical values from Y.

   (b) What is the optimal value of V?

   (c) What is the associated overall training error?

   (d) Bob is happy about what he has accomplished, until he realizes that there are a bunch of movies and users that he still needs to add to his database! He sees that his database will slowly grow over time, and that it will be time-consuming to train a completely new model every single time he updates his database. If Bob has an $m \times n$ data matrix which he wants to find a rank k factorization for, his analysis indicates that the worst-case run-time (in terms of number of expensive multiplications) of performing alternating least squares for t iterations (where each iteration updates both U and V) will be $O(k^2mnt)$.

Instead, Bob comes upon the following idea: whenever he gets information about a new movie, he adds an extra row to V but does not alter the existing entries of U or V. He then finds the values of the entries in that extra row that minimize the objective function (with no regularization). He performs a similar procedure when he gets a new user, but instead adds an extra row to U. Working from the dataset in the first part, say Bob receives a new movie to which his first user has given the rating 4. What is the updated value of V?

(e) With this updated V, what rating does Bob predict that the second user will give this movie?

(f) Bob continues using this update scheme whenever he adds new movies and users. Does the order in which Bob receives new information affect the final values of U and V that he learns? If your answer is yes, explain in detail why they are different. If your answer is no, explain in detail why they are the same.

(g) Let us say that Bob modifies this procedure, so that he still adds new movies and users in this way, but after every 100 new additions, he retrains U and V from scratch using alternating least squares. Would you expect that this method would make better predictions than if we just used Bob's original procedure? Explain.

(h) After having added a few thousand users and movies to his database, Bob wants to try analyzing the user and movie vectors that he has learned, in order to see whether he can interpret what is causing customers to like certain movies over others. However, some of the numbers in U and V have a very high magnitude, which may lead to problems with numerical precision. How might Bob adjust his training process to fix the problem of high magnitude numbers in U and V?

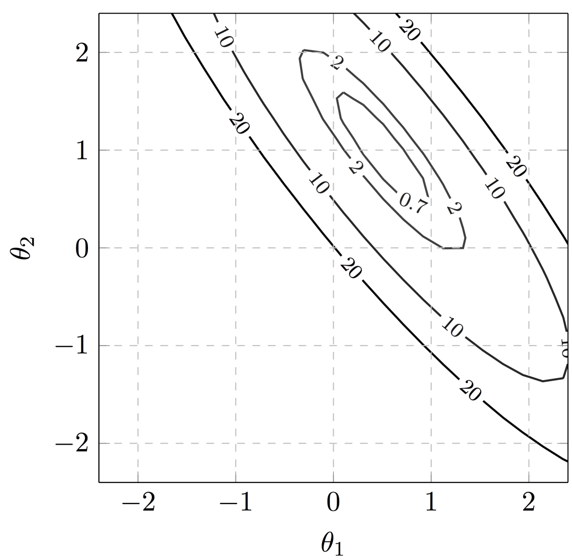
Linear Regression with Regularization

7. (14 points) In this problem, we consider using linear regression with a regularization term. Assume a dataset of n samples $\{(x^{(i)}, y^{(i)})\}$ with $x^{(i)} \in \mathbb{R}^2$ and output values $y^{(i)} \in \mathbb{R}$. Recall that the ridge regression objective is defined as follows:

$$J_{ridge}(\theta) = J_{data}(\theta) + J_{reg}(\theta) = \frac{1}{n}\sum_{i=1}^{n}(\theta^T x^{(i)} - y^{(i)})^2 + \lambda||\theta||^2$$

where $\theta = [\theta_1, \theta_2]$ and λ is the regularization trade-off parameter.

Chris would like to solve the problem of computing θ that minimizes the ridge regression objective. He will employ graphical methods to obtain the solution. When plotting *just* the data error term, $J_{data}(\theta)$, as a function of θ_1 and θ_2, the following set of isocontour lines (curves connecting sets of θ_1, θ_2 for which the objective value is constant) is obtained, for his dataset:



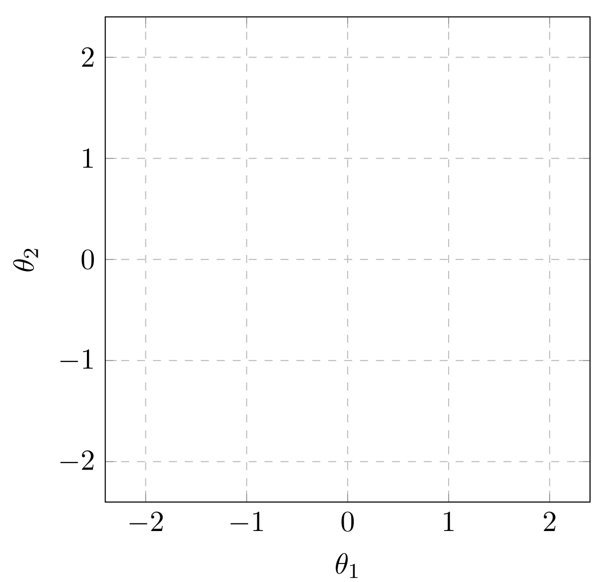
(a) What is the optimum solution θ^* when you minimize only the data error term, $J_{data}(\theta)$, i.e., for $\lambda = 0$? Give an approximate value, for Chris's data.

(b) In general, is the data error term $J_{data}(\theta^*)$ guaranteed to be zero for the optimal value of θ, for the case when $\lambda = 0$? Explain.

(c) Recall that $\nabla J_{data}(\theta)$ is a vector in 2D. In general, at any parameter vector θ, describe the geometric relationship between $\nabla J_{data}(\theta)$ and the isocontour line of the data error term $J_{data}(\theta)$ that passes through θ.

(d) What is $\nabla J_{data}(\theta^*)$ at the optimum θ^*, when $\lambda = 0$?

(e) Now we consider regularization. Sketch the isocontour lines for just the regularization term, $J_{reg}(\theta)$. Clearly label the contour line corresponding to the values of θ for which this term has value 1, when $\lambda = 1$.



(f) What is the effect of the regularization trade-off parameter λ on the **shape** and **value** of the isocontour lines of the regularization term $J_{reg}(\theta)$?

(g) Now consider the gradient of the regularization term $\nabla J_{reg}(\theta)$. Towards what specific point does the $-\nabla J_{reg}(\theta)$ vector point to?

(h) If λ is *very* large, what is the θ^* that minimizes $J_{ridge}(\theta^*)$? What approximate numerical value does $J_{data}(\theta^*)$ have for Chris's data?

(i) Given a general optimal solution θ^* for $J_{ridge}(\theta)$ for a given (finite) λ, what is the algebraic relationship between $\nabla J_{data}(\theta^*)$ and $\nabla J_{reg}(\theta^*)$?

Name: ____________________

Training Neural Networks with Regularization

8. (8 points) In this problem we will investigate regularization for neural networks.

Kim constructs a fully connected neural network with L=2 layers using mean squared error (MSE) loss and ReLU activation functions for the hidden layer, and a linear activation for the output layer. The network is trained with a gradient descent algorithm on a data set of n points $\{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$.

Recall that the update rule for weights W^1 can be specified in terms of step size η and the gradient of the loss function with respect to weights W^1. This gradient can be expressed in terms of the activations A^l, weights W^l, pre-activations Z^l, and partials $\frac{\partial L}{\partial A^2}$, $\frac{\partial A^l}{\partial Z^l}$, for l = 1,2:

$$W^1 := W^1 - \eta \sum_{i=1}^{n} \frac{\partial L(h(x^{(i)}; W), y^{(i)})}{\partial W^1},$$

where $h(\cdot)$ is the input-output mapping implemented by the entire neural network, and

$$\frac{\partial L}{\partial W^1} = \frac{\partial Z^1}{\partial W^1} \cdot \frac{\partial A^1}{\partial Z^1} \cdot W^2 \cdot \frac{\partial A^2}{\partial Z^2} \cdot \frac{\partial L}{\partial A^2}.$$

(a) Derive a new update rule for weights W^1 which also penalizes the sum of squared values of all individual weights in the network:

$$L^{new} = L(h(x^{(i)}; W), y^{(i)}) + \lambda ||W||^2$$

where λ denotes the regularization trade-off parameter. You can express the new update rule as follows:

$$W^1 := \alpha W^1 - \eta \sum_{i=1}^{n} \frac{\partial L(h(x^{(i)}; W), y^{(i)})}{\partial W^1}$$

where $L(\cdot)$ represents the previous prediction error loss.

What is the value of α in terms of λ and η?

(b) Explain how this new update rule helps the neural network reduce overfitting to the data.

(c) Given that we are training a neural network with gradient descent, what happens when we increase the regularization trade-off parameter λ too much, while holding the step size η fixed?

Lost in Translation

9. (10 points) We want to make an RNN to translate English to Martian. We have a training set of pairs $(e^{(i)}, m^{(i)})$, where $e^{(i)}$ is a sequence of length $J^{(i)}$ of English words and $m^{(i)}$ is a sequence of length $K^{(i)}$ of Martian words. The sequences, even within a pair, do not need to be of the same length, i.e., $J^{(i)}$ need not equal $K^{(i)}$. We are considering two different strategies for turning this into a transduction or sequence-to-sequence learning problem for an RNN.

Method 1: Construct a training-sequence pair (x, y) from an example (e, m) by letting

$$x = (e_1, e_2, \ldots, e_L, \mathit{stop})$$
$$y = (m_1, m_2, \ldots, m_L, \mathit{stop})$$

In Method 1, we assume that if the original e and m had different numbers of words, then the shorter sentence is padded with enough time-wasting words ("ummm" for English, "grlork" for Martian) so that they now have equal length, L. Any needed padding words are inserted at the end of $e^{(i)}$, and at the start of $m^{(i)}$.

Method 2: Construct a training-sequence pair (x, y) from an example (e, m) by letting

$$x = (e_1, e_2, \ldots, e_J, \mathit{stop}, \mathit{blank}, \ldots, \mathit{blank})$$
$$y = (\mathit{blank}, \ldots, \mathit{blank}, m_1, m_2, \ldots, m_K, \mathit{stop})$$

In Method 2, blanks are inserted at the end of e and start of m such that the length of x and y are now both $J + K + 1$.

(a) Assume an element-wise loss function $L_{elt}(p, y)$ on predicted versus true Martian words. What is an appropriate sequence loss function for **Method 1**? Assume that the predicted sequence p has the same length as the target sequence y.

(b) Assume an element-wise loss function $L_{elt}(p, y)$ on predicted versus true Martian words. What is an appropriate sequence loss function for **Method 2**? Assume the predicted sequence p has the same length as the target sequence y.

(c) Which method is likely to need a higher dimensional state? Explain why.

(d) Which method is better if English and Martian have very different word order? Explain why.

(e) Martian linguist Grlymp thinks it is also important to pad the original English and Martian sentences with time-wasting word to be of the same length for Method 2 (i.e., so that $J = K$), but English linguist Chome Nimsky disagrees. Who is correct, and why?

Name: ______________________________

Work space