Submission instructions:

- If a problem asks for a numerical answer, you need only provide this answer. There is no need to show your work, unless you would like to.
- Please type up your solutions. We suggest using an online latex editor like www.overleaf.com, though this is not a requirement.
- Upload the PDF file for your homework to gradescope by midnight on Monday, Mar 11.

1. Draw the decision boundary in \( \mathbb{R}^2 \) that corresponds to the prediction rule \( \text{sign}(2x_1 - x_2 - 6) \). Make sure to clearly indicate where this boundary intersects the axes. Show which side of the boundary is classified as positive and which side as negative.

2. The Perceptron algorithm is run on a data set, and converges after performing \( p + q \) updates (i.e. makes \( p + q \) incorrect predictions during training). Of these updates, \( p \) are on data points whose label is \(-1\) and \( q \) are on data points whose label is \(+1\). What is the final value of the parameter \( b \)? (Equivalently, what is the final value of the weight on the bias feature?)

3. An SVM classifier is learned for a data set in \( \mathbb{R}^2 \). It is given by \( w = (3, 4) \) and \( b = -12 \) (or, equivalently, \( w = (3, 4, -12) \) where the last feature is the bias feature).
   
   (a) Draw the decision boundary, making sure to clearly indicate where it intersects the axes.
   
   (b) Draw the left- and right-hand boundaries, also clearly marking where they intersect the axes.
   
   (c) What is the margin of this classifier?
   
   (d) How would the point \((2, 2)\) be classified?
   
   (e) It turns out that the data set has two distinct support vectors of the form \((1, ?)\). What are they?

4. Perceptron algorithm. In this problem, you will code up the Perceptron algorithm and use it to classify the Iris data set.

   (a) Write code for two functions:
      
      - The first function takes as input parameters \( w, b \) of a linear classifier as well as a data point \( x \), and returns the label for that point: \( \text{sign}(w \cdot x + b) \). The label is either \(+1\) or \(-1\).
      
      - The second function takes as input an array of data points and an array of labels (where each label is \(+1\) or \(-1\)), and runs the Perceptron algorithm to learn a linear classifier \( w, b \). The algorithm should begin by randomly permuting the data points.

      In your writeup, give the code for these two functions.

   (b) Load in the Iris data set. You can do this by simply invoking:
from sklearn import datasets
iris = datasets.load_iris()
x = iris.data
y = iris.target

The data has four features and three labels. Restrict it to features 1 and 3 (the second and fourth columns, sepal width and petal width) and to labels 0,1. Recode label 0 as $-1$, since this is what the Perceptron algorithm is expecting. If you are going to use a bias feature, rather than including the bias parameter $b$, add the bias feature to the data.

(c) Now run the Perceptron algorithm on the data. In your writeup, show a plot with the data points (where the two labels have different colors) and the resulting decision boundary. (If you included a bias feature, plot the data and decision boundary in the original two dimensional space.)

5. Consider the following small data set in $\mathbb{R}^2$:

- Points $(1,2), (2,1), (2,3), (3,2)$ have label $-1$.
- Points $(4,5), (5,4), (5,6), (6,5)$ have label $+1$.

Now, suppose (hard margin) SVM is run on this data.

(a) Sketch the resulting decision boundary.
(b) What is the (numerical value of the) margin, exactly?
(c) What are $w$ and $b$, exactly?

6. Support vectors. The picture below shows the decision boundary obtained upon running soft-margin SVM on a small data set of blue squares and red circles.

(a) Copy this figure and mark the support vectors. For each, indicate the approximate value of the corresponding slack variable.
(b) Suppose the factor $C$ in the soft-margin SVM optimization problem were increased. Would you expect the margin to increase or decrease?

7. Support vector machine. Again use the Iris data set, but this time use features 0 and 2, and labels 1,2.

(a) Is this data linearly separable?
(b) Use sklearn.svm.SVC to fit a support vector machine classifier to the data. You will need to invoke the option $\text{kernel}='\text{linear}'$. Try at least 10 different values of the slack parameter $C$. In your writeup, include a table that shows these values of $C$ and for each of them gives the training error and the number of support vectors.
(c) Which value of $C$ do you think is best? For this value, include a plot of the data points and the linear decision boundary.

8. **Projections.** Let $u_1, u_2 \in \mathbb{R}^p$ be two vectors with $\|u_1\| = \|u_2\| = 1$ and $u_1 \cdot u_2 = 0$. Define $U$ to be the matrix whose columns are $u_1$ and $u_2$.

(a) What are the dimensions of each of the following?
- $U$
- $U^T$
- $UU^T$
- $u_1 u_1^T$

(b) What are the differences, if any, between the following four projections?
- $x \mapsto (u_1 \cdot x, u_2 \cdot x)$
- $x \mapsto (u_1 \cdot x)u_1 + (u_2 \cdot x)u_2$
- $x \mapsto U^T x$
- $x \mapsto UU^T x$

9. In this problem, you are going to manually differentiate some simple neural networks. Let $\sigma$ denote the element-wise logistic function. Compute the gradient(s), with respect to each argument, for each of the following functions:

(a) $L(w) = \sigma(w^\top x)$, where $w \in \mathbb{R}^d$
(b) $L(w, a) = \sigma(a \cdot \sigma(w^\top x))$, where $w \in \mathbb{R}^d$ and $a \in \mathbb{R}$
(c) $L(W, u) = \sigma(u^\top \sigma(W^\top x))$, where $u \in \mathbb{R}^k$ and $W \in \mathbb{R}^{d \times k}$

10. Suppose you are given a dataset with four points and two classes. First plot these points for yourself, and convince yourself that any linear classifier will be unable to classify all 4 points correctly. We have drawn you a simple 1 hidden layer MLP, with the 2 inputs $(X_1, X_2)$, 2 hidden nodes $(H_1, H_2)$, 2 biases $(B_1, B_2)$ and one output. All of the parameters are written in the table as well as illustrated on the figure where $W_{ij}$ is the edge weight that connects node $i$ in layer $L$ to node $j$ in layer $L+1$. Assume that the sign activation function (i.e. returning +1 if input is positive, and -1 otherwise) is used at each node, except for the output node. For the output node, assume the logistic function is used to produce $P(y = 1|x)$. Give weights for this MLP that will perfectly classify the training set.
11. In this problem, you will be building neural networks for digit classification using two different approaches. You will again use the MNIST train, validation, and test datasets you have used in past homeworks. Because your neural network will learn its own feature representation, you will be using the raw image vector as input, $x$ (though, you may want to divide each image by its maximum pixel value for numerical stability).

(a) First, implement a simple multilayer perceptron (MLP) with 1 hidden layer and a logistic activation function. Specifically, your network will have 2 layers: a hidden layer of size $k$ and the final output layer. Remember that your input is of size $28 \times 28 = 784$. Plot your training curve and best validation accuracy after experimenting with optimization parameters for subgradient descent, as well as the width of your hidden layer.

(b) Convolutions are a general concept that appear across many different disciplines and not only in your deep learning architectures. They can operate on many types of data including images, audio, text and many other signals. We will see how 2D convolutional neural nets operate for images and try to gain intuition for their various parameterizations.

2D convolutional neural nets are specified by various hyperparameters, a receptive field (filter size, or kernel size), number of filters $K$, stride $S$, amount of zero padding $P$, and type of pooling. We will represent our input data, as well as the hidden layers, as 3D-arrays. We will denote their dimensions by tuples, $W \times H \times D$, of width, height, and depth respectively. Since MNIST images are black-and-white and thus have scalar-valued pixels, the depth of the input image is 1. This means the total input dimensionality is $28 \times 28 \times 1$. Suppose, for now, that MNIST was, in fact, in color (RGB). This means the depth of the input image would be 3. Calculate the dimensionality of the output for the following convolutions applied to a color-valued MNIST input:

i. Convolution Filter size of $2 \times 2$, number of filters 33, stride of 2, padding of 0
ii. Convolution Filter size of $3 \times 3$, number of filters 55, stride of 1, padding of 1
iii. Convolution Filter size of $3 \times 3$, number of filters 77, stride of 0, padding of 1. Followed by a Max Pooling with filter size of $2 \times 2$ and stride 2.

(c) For each question above provide the number of parameters there are if each convolution shares parameters within a depth slice ($W \times H$) but still has different ones across the depth slices ($D$).

(d) Next, lets implement a convolutional neural network to see if we can extract features from the mnist data directly for our classifier. For this start with one convolutional layer and one non-linearity to extract the features. For the convolutional layer use a $5 \times 5$ kernel, with stride 1 and zero-padding of size 2. Use tanh as your non-linear activation.

(e) Finally try experimenting and building deeper (more layers) architectures. You have the option of choosing the conv, pooling, fully connected (FC) layers as well as the activation function and all of the hyper parameters we have discussed this far. To help get started, it maybe easy to use conv/pooling layers that preserve the dimensions or half them. This should help make it easier getting your first deep architectures running without errors.