Homework Assignment 2

COGS 181: Neural Networks and Deep Learning

Due: 11:59pm, Thursday, January 26th 2023 (Pacific Time).

Instructions: Answer the questions below, attach your code, and insert figures to create a PDF file; submit your file via Gradescope. You may look up the information on the Internet, but you must write the final homework solutions by yourself.

Late Policy: 5% of the total points will be deducted on the first day past due. Every 10% of the total points will be deducted for every extra day past due.

System Setup: You can install Anaconda to setup the Jupyter Notebook environment. Most packages have been already installed in Anaconda. If some package is not installed, you can use `pip` to install the missing package, that is, just type `pip install PACKAGE_NAME` in the terminal.

Grade: ____ out of 100 points

1 (20 points) Conceptual Questions

1. Is the following statement true or false?

   For inputs of two-dimensional features $\mathbf{x} = (x_1, x_2)$, a perceptron classifier cannot solve the XOR problem (Figure 1), but a logistic regression classifier can.

   [True]

   [False]

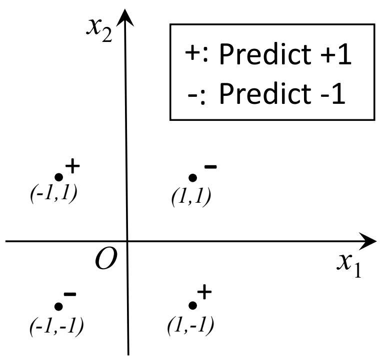

Figure 1: The XOR problem

2. Choose **all** the valid linear classifiers from the options below:

   A. Perceptron.
   B. Logistic regression classifier.
   C. Support vector machine (linear kernel).
   D. Support vector machine (RBF kernel).
   E. Multi-layer neural networks classifier.

3. Choose **all** the valid descriptions about **gradient descent** from the options below:

   A. The global minimum can always be reached by using gradient descent.
   B. Every gradient descent iteration can always decrease the value of loss function even when the gradient of the objective function is zero.
   C. When the learning rate is very large, some iterations of gradient descent may not decrease the value of loss function.
   D. With different initial weights, the gradient descent algorithm may lead to different local minimum.
   E. None of the above is valid.

4. Choose **all** the valid descriptions about the **stochastic gradient descent** algorithm from the options below:

   A. Stochastic gradient decent (SGD) always find the global optimal solution.
   B. Stochastic gradient decent (SGD) is typically more computationally efficient than standard gradient decent (GD).
   C. Stochastic gradient decent (SGD) is typically more memory demanding than standard gradient decent (GD).
   D. Stochastic gradient decent (SGD) can be used to avoid bad local opitmals.
   E. Stochastic gradient decent (SGD) uses a fixed learning rate.

5. In this question, we will apply perceptron learning algorithm to solve a binary classification problem: We need to predict a binary label $y \in \{-1, +1\}$ for a feature vector $\mathbf{x} = [x_0, x_1]^\top$. The decision rule of the perceptron model is defined as:

$$f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1, & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 0, \\ -1, & \text{otherwise.} \end{cases} \tag{1}$$

   where $\mathbf{w} = [w_0, w_1]^\top$ is the weight vector, and b is the bias scalar. Given a training dataset $S_{\text{training}} = \{(\mathbf{x}_i, y_i)\}, i = 1, \ldots, n\}$, the outline of the perceptron algorithm is shown as below:

   - Initialize weight $\mathbf{w}$ and bias b.
   - If not all data points in S_{training} are correctly predicted:
     - Obtain the prediction for each feature vector in S_{training}.
     - If the prediction is wrong, then update the parameters $\mathbf{w}$ and b.
   - Otherwise return current weight $\mathbf{w}$ and bias b.

   In this problem, you will implement a perceptron algorithm. Suppose `X` is the matrix of all feature vectors $\mathbf{x}_i$ and `Y` is the matrix of all labels y_i in the training set S_{training}. Besides, assume a function `calc_error(X, Y, W, b)` is given, which computes the error from the feature matrix `X`, the label matrix `Y`, the weight `W` and the bias `b`.

Please reorder the following lines of Python code to complete your perceptron algorithm. Fill the indexes (a-g) into the blanks.

```
def perceptron_algorithm(X, Y):
    # Initialization.
    W   = np.zeros(2)
    b   = 0.0
    lam = 1
    # Main algorithm.
    while calc_error(X, Y, W, b) > 0:
        for xi, yi in zip(X, Y):
            __________________________________________________
                ______________________________________________
            else:
                ______________________________________________
            __________________________________________________
                ______________________________________________
            else:
                ______________________________________________
                ______________________________________________
    return W, b
```

(a) `W = W + lam * (yi - yi_pred) * xi`

(b) `yi_pred = -1`

(c) `b = b + lam * (yi - yi_pred)`

(d) `yi_pred = +1`

(e) `if W.T.dot(x) + b >= 0:`

(f) `continue`

(g) `if yi_pred == yi:`

2 (10 points) Perceptron Update Rule

We are given a training set $S = \{(\mathbf{x}_i, y_i), i = 1, 2, ..., n\}$, where $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \{-1, +1\}$. Define the objective function for sample i when training the perceptron as:

$$\mathcal{L}_i(\mathbf{w}, b) = \max(0, -y_i(\mathbf{w}^T\mathbf{x}_i + b)).$$

For a chosen sample point i, let $target_i = y_i$ and $output_i = sign(\mathbf{w}^T\mathbf{x}_i + b)$. Show your proof that:

1. $\frac{\mathcal{L}_i(\mathbf{w},b)}{\partial \mathbf{w}} = -\frac{1}{2}(target_i - output_i)\mathbf{x}_i$

2. $\frac{\mathcal{L}_i(\mathbf{w},b)}{\partial b} = -\frac{1}{2}(target_i - output_i)$

3 (30 points) Perceptron

3.1 (20 points) Perceptron Algorithm - Deterministic Iteration

In this section, we will apply perceptron learning algorithm to solve the same binary classification problem as logistic regression above: We need to predict a binary label $y \in \{-1, 1\}$ for a feature vector $\mathbf{x} = [x_0, x_1]^\top$. The decision rule of the perceptron model is defined as:

$$f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} 1, & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 0, \\ -1, & \text{otherwise.} \end{cases} \tag{2}$$

where $\mathbf{w} = [w_0, w_1]^\top$ is the weight vector, and b is the bias scalar. Given a training dataset $S_{\text{training}} = \{(\mathbf{x}_i, y_i)\}, i = 1, \ldots, n\}$, we define the training error e_{training} as:

$$e_{\text{training}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\big(y_i \neq f(\mathbf{x}_i; \mathbf{w}, b)\big) \tag{3}$$

and the test error e_{test} on the test set S_{test} can be defined in the same way. In the perceptron algorithm, we aim to directly minimize the training error e_{training} in order to obtain the optimal parameters $\mathbf{w}^*, b^*$. If we represent data points in training set S_{training} as matrices $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T$ and $Y = [y_1, y_2, \ldots, y_n]^T$, the perceptron algorithm is shown as below:

Algorithm 1 Perceptron Learning Algorithm

Data: Training set S_{training}, which contains feature vectors X and labels Y;
Initialize parameters $\mathbf{w}$ and b; pick a constant $\lambda \in (0, 1]$, which is similar to the step size in the standard gradient descent algorithm ($\lambda = 1$ by default).

```
while not every data point is correctly classified do
    for each feature vector x_i and its label y_i in the training set S_training do
        compute the model prediction f(x_i; w, b);
        if y_i = f(x_i; w, b) then
            continue;
        else
            update the parameters w and b:
              w ⇐ w + λ(y_i − f(x_i; w, b))x_i
              b ⇐ b + λ(y_i − f(x_i; w, b))
        end
    end
end
```

Please fill out the missing blanks in `perceptron.ipynb`. Follow the instructions in the skeleton code and report:

1. Your code.
2. Equation of decision boundary corresponding to the optimal $\mathbf{w}^*$ and b^*.
3. Plot of training set along with decision boundary.
4. Plot of test set along with decision boundary.
5. Training error and test error.
6. Training error curve.

Hint:

1. Multiplication inside `f_perceptron` is matrix-vector(or vector-vector) multiplication. You can use `dot`.
2. Inside the `update rule`, and $y_i - f(\mathbf{x}_i; \mathbf{w}, b)$ are scalar while $\mathbf{x}_i$ is a vector. You should consider an element-wise multiplication using `*`.

3.2 (10 points) Perceptron Algorithm - Random Sample

In this section, we want to execute perceptron algorithm in a slight different way. In previous section, perceptron algorithm visits each feature vector $\mathbf{x}_i$ and its label y_i in the training set S_{training} in a deterministic order (Line 5 for each feature ...). Now, we want to random sample next $\mathbf{x}_i$ and its label y_i from the training set S_{training} with replacement. The new algorithm is shown as below:

Algorithm 2 Perceptron Learning Algorithm

Data: Training set S_{training}, which contains feature vectors X and labels Y;
Initialize parameters $\mathbf{w}$ and b; pick a constant $\lambda \in (0, 1]$, which is similar to the step size in the standard gradient descent algorithm ($\lambda = 1$ by default).
while *not every data point is correctly classified* **do**
  for *i in range(0,k)* **do**
    random sample each feature vector $\mathbf{x}_i$ and its label y_i from the training set S_{training}
    compute the model prediction $f(\mathbf{x}_i; \mathbf{w}, b)$;
    if $y_i = f(\mathbf{x}_i; \mathbf{w}, b)$ **then**
      continue;
    else
      update the parameters $\mathbf{w}$ and $\mathbf{b}$:
       $\mathbf{w} \Leftarrow \mathbf{w} + \lambda(y_i - f(\mathbf{x}_i; \mathbf{w}, b))\mathbf{x}_i$
       $\mathbf{b} \Leftarrow \mathbf{b} + \lambda(y_i - f(\mathbf{x}_i; \mathbf{w}, b))$
    end
  end
end

Please set k = training data and follow the instructions in the skeleton code and report:

1. Your code.

2. Run your code for 3 times, and draw 3 training error curves and the curve with deterministic training(Section 3.1). Compare and write your observation between the two sampling methods.

Hint: You can randomly sample the indices of training data from 0 to k-1 and then use the index to get corresponding sample each time. `random.choices(pop,n)` would return a n sized list of elements chosen from the input list pop with replacement.

4 (20 points) Logistic Regression

Assume in a binary classification problem, we need to predict a binary label $y \in \{-1, +1\}$ for a feature vector $\mathbf{x} = [x_0, x_1]^\top$. In logistic regression, we can reformulate the binary classification problem in a probabilistic framework: We aim to model the distribution of classes given the input feature vector $\mathbf{x}$. Specifically, we can express the conditional probability $p(y|\mathbf{x})$ parameterized by $(\mathbf{w}, b)$ using a logistic function. Assume the probability of the positive prediction $p(y = +1|\mathbf{x})$ is represented as:

$$p(y = +1|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}} \tag{4}$$

4.1 (10 points) Basic Formulation

1. Please derive the formulation of $p(y = -1|\mathbf{x})$.

2. Please show that $p(y|\mathbf{x}) = \dfrac{1}{1 + e^{-y(\mathbf{w}^T\mathbf{x}+b)}}$.

4.2 (10 points) Derive Gradient

Given a training dataset $S_{\text{training}} = \{(\mathbf{x}_i, y_i)\}, i = 1, \ldots, n\}$, we wish to optimize the negative log-likelihood loss $\mathcal{L}(\mathbf{w}, b)$ of the logistic regression model defined above:

$$\mathcal{L}(\mathbf{w}, b) = -\sum_{i=1}^{n} \ln p_i \tag{5}$$

where $p_i = p(y_i|\mathbf{x}_i)$. The optimal weight vector $\mathbf{w}$ and bias b are used to build the logistic regression model:

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b) \tag{6}$$

In this problem, we attempt to obtain the optimal parameters $\mathbf{w}^*$ and b^* by using a standard gradient descent algorithm.

(a) Please show that $\dfrac{\partial \mathcal{L}(\mathbf{w}, b)}{\partial \mathbf{w}} = -\displaystyle\sum_{i=1}^{n}(1 - p_i)y_i\mathbf{x}_i$.

(b) Please show that $\dfrac{\partial \mathcal{L}(\mathbf{w}, b)}{\partial b} = -\displaystyle\sum_{i=1}^{n}(1 - p_i)y_i$.

5 (20 points) Circuit Diagram I

The computation of the backpropagation can be nicely visualized with a circuit diagram. Let's consider a complicated expression which involves multiple composite functions, and the function is defined as

$$f(x, y, z) = (x + y)z. \tag{7}$$

We visualize the function with a circuit diagram, which helps us intuitively understand the backpropagation. You can review it from *Fei-Fei Li*, *Andrej Karpathy* and *Justin Johnson*'s Stanford CS231n course. Link: http://cs231n.github.io/optimization-2/ The visualization is shown in Fig. 2.

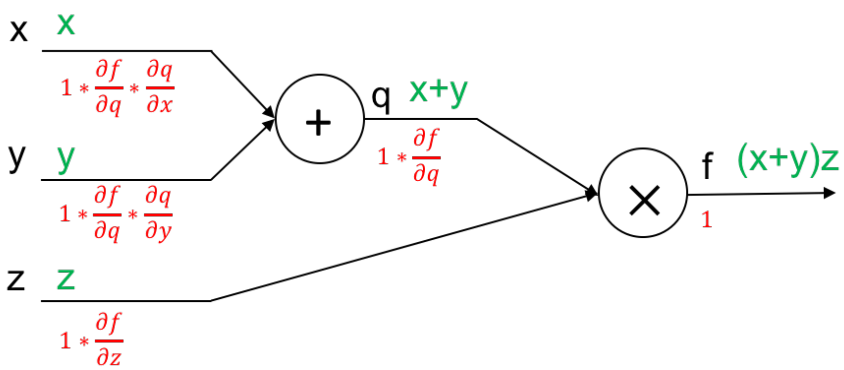

Figure 2: Circuit Diagram of the function $f(x, y, z) = (x + y)z$.

Particularly, this expression can be broken down into two expressions: $q = x + y$ and $f = qz$. According to **Chain Rule**, the derivative/gradient of f with respect to its inputs x, y, and z can be calculated easily with the intermediate variable q. Although we don't really care about the gradient of f with respect to q, but the gradient $\frac{\partial f}{\partial q}$ simplifies the calculation of the gradient.

Let's take a look into the circuit diagram. The letters in black represent the variables, which are input variables x, y, z, intermediate variable q and output f. The expressions in green represent the current values of the variables respectively, and the expressions in red represent the gradient of f with respect to the particular variables.

The circuit diagram visualizes how the gradient of f with respect to each variable is calculated. Let's follow the circuit diagram to perform a forward calculation and a backward calculation by hand.

In Fig. 3, suppose we have three inputs, which are $x = -2$, $y = 5$ and $z = -4$, and the circuit diagram represents the function $f(x, y, z) = (x + y)z$. We follow the steps below to calculate the value and the gradient with respect to each variable in the diagram.

1. $q = x + y = 3$

2. $f = qz = -12$

3. $\dfrac{\partial f}{\partial q} = \dfrac{\partial (qz)}{\partial q} = z = -4$

4. $\dfrac{\partial f}{\partial z} = \dfrac{\partial (qz)}{\partial z} = q = 3$

5. $\dfrac{\partial f}{\partial x} = \dfrac{\partial f}{\partial q}\dfrac{\partial q}{\partial x} = -4 \times 1 = -4$

6. $\dfrac{\partial f}{\partial y} = \dfrac{\partial f}{\partial q}\dfrac{\partial q}{\partial y} = -4 \times 1 = -4$

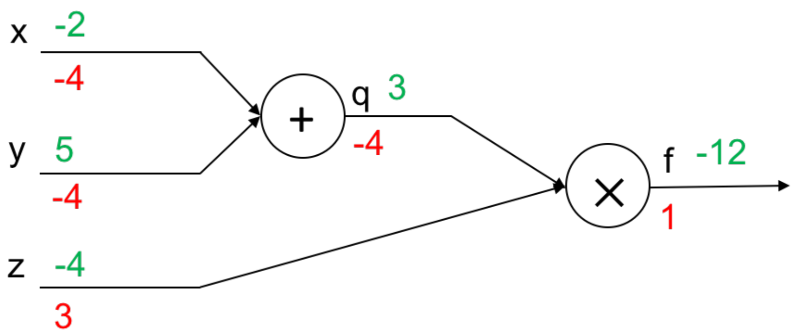

Figure 3: The real-valued circuit diagram of the function $f(x, y, z) = (x + y)z$.

Your tasks:
The circuit diagram for the function defined as below is provided in Fig. 4:

$$f(x, y, z, w) = \ln[\max(x, y) \times (z + w)], \tag{8}$$

where $\max(x, y)$ takes the maximum value of x and y as output.

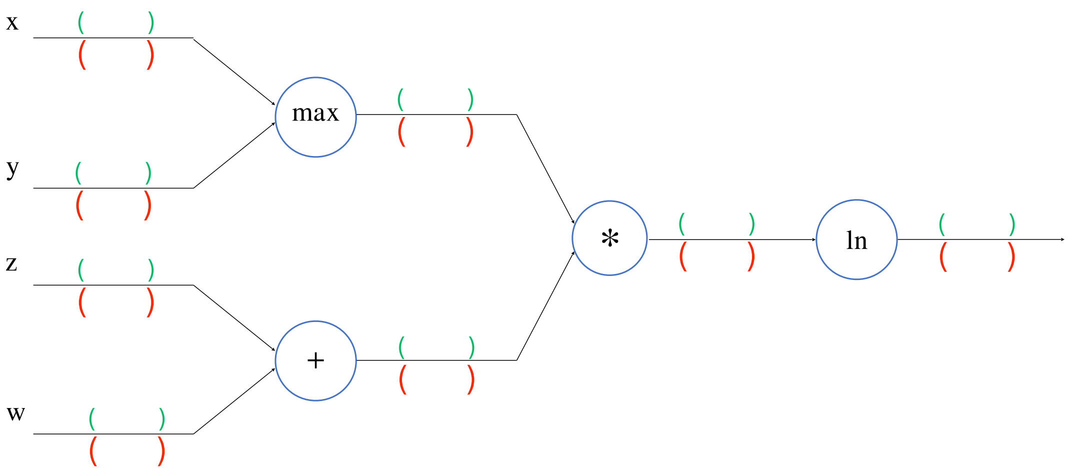

Figure 4: Circuit Diagram of the function $f(x, y, z, w)$. We use $*$ to denote the multiplication of the two scalar values.

Suppose the value for each input is $x = 2.5$, $y = 1.5$, $z = 1$ and $w = 3$.

1. (**10 points**) Perform the forward computation of the function f, and fill-in the circuit diagram in Fig. 4 with your calculated values.

2. (**10 points**) Perform the backward computation of the function f, and fill-in the circuit diagram in Fig. 4 with your calculated gradients.