CS 5350/6350: Machine Learning Fall 2023

Homework 5

Handed out: 21 Nov, 2022
Due date: 11:59pm, 8 Dec, 2022

- You are welcome to talk to other members of the class about the homework. I am more concerned that you understand the underlying concepts. However, you should write down your own solution. Please keep the class collaboration policy in mind.
- Feel free to discuss the homework with the instructor or the TAs.
- Your written solutions should be brief and clear. You do not need to include original problem descriptions in your solutions. You need to show your work, not just the final answer, but you do *not* need to write it in gory detail. Your assignment should be **no more than 20 pages**. Every extra page will cost a point.
- Handwritten solutions will not be accepted.
- *Your code should run on the CADE machines.* **You should include a shell script, `run.sh`, that will execute your code in the CADE environment. Your code should produce similar output to what you include in your report.**

  You are responsible for ensuring that the grader can execute the code using only the included script. If you are using an esoteric programming language, you should make sure that its runtime is available on CADE.
- Please do not hand in binary files! We will *not* grade binary submissions.
- The homework is due by **midnight of the due date**. Please submit the homework on Canvas.

1 Paper Problems [40 points]

1. [5 points] (Warm up) Suppose we have a composite function, $z = \sigma(y_1^2 + y_2 y_3)$, where $y_1 = 3x$, $y_2 = e^{-x}$, $y_3 = \sin(x)$, and $\sigma(\cdot)$ is the sigmoid activation function . Please use the chain rule to derive $\frac{\partial z}{\partial x}$ and compute the derivative at $x = 0$.

   We know that we first must apply the chain rule, whereby $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial u} * \frac{\partial u}{\partial x}$

   We know that $\sigma'(u) = (\sigma(u))(1 - \sigma(u))$

   We know that $\frac{\partial u}{\partial x} = \frac{\partial}{\partial x}(y_1^2 + y_2 y_3) = (2 * 3x * 3 + e^{-x} * (-e^{-x}) + sin(x) * cos(x))$

   The derivative at x = 0 would make this equation: $\frac{\partial z}{\partial x} = (\sigma(u))(1 - \sigma(u) * (18(0) + 1 + sin(0)cos(0)))$

   $= \frac{1}{2} * (1 + 0)$

   $= \frac{1}{2}$

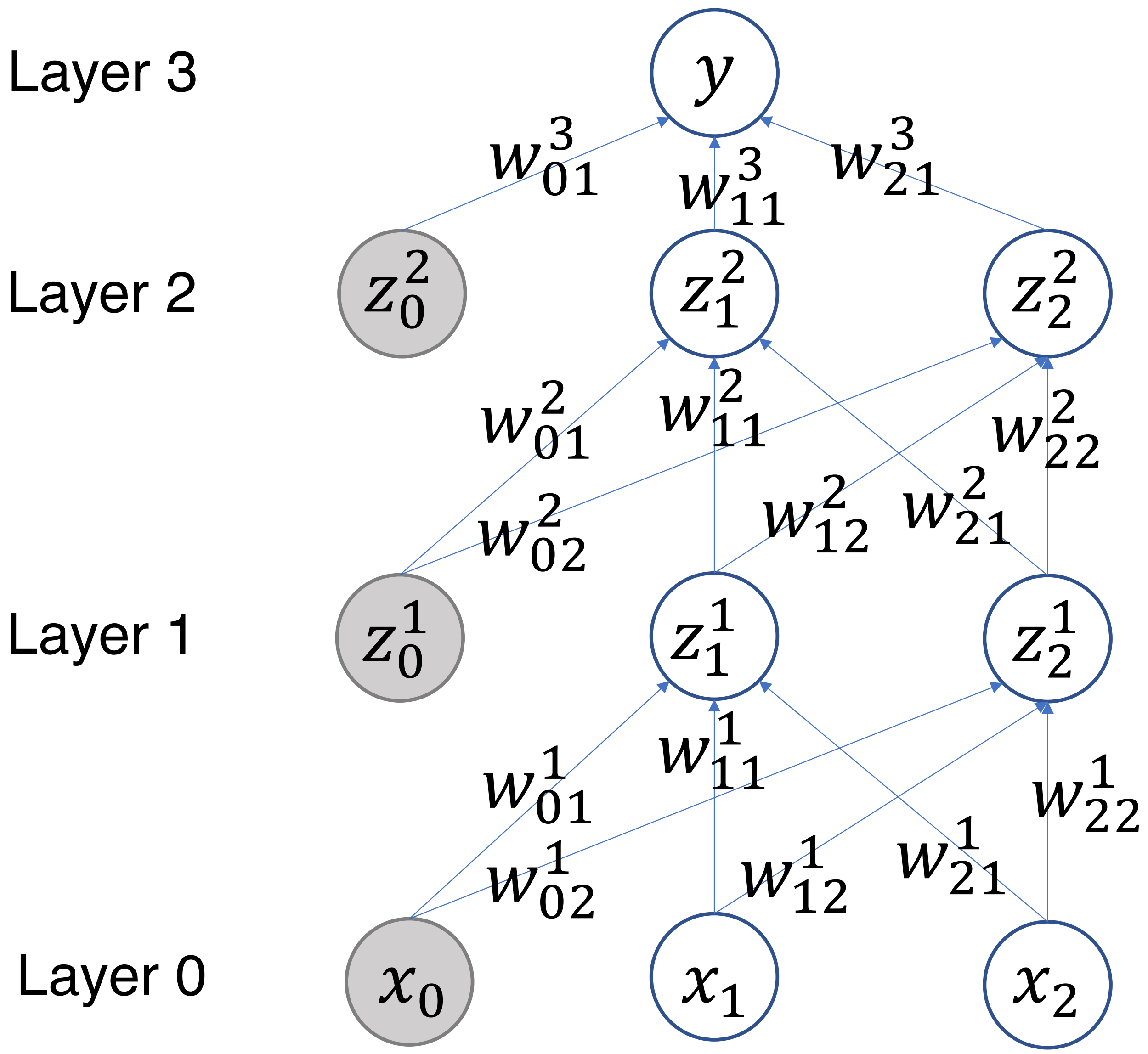

Figure 1: A three layer artificial neural network.

Layer	weigth	value
1	w^1_{01}	-1
1	w^1_{02}	1
1	w^1_{11}	-2
1	w^1_{12}	2
1	w^1_{21}	-3
1	w^1_{22}	3
2	w^2_{01}	-1
2	w^2_{02}	1
2	w^2_{11}	-2
2	w^2_{12}	2
2	w^2_{21}	-3
2	w^2_{22}	3
3	w^3_{01}	-1
3	w^3_{11}	2
3	w^3_{21}	-1.5

Table 1: Weight values.

2. [5 points] Suppose we have a three-layered feed-forward neural network in hand. The architecture and the weights are defined in Figure 1. We use the sigmoid activation function. Note that the shaded variables are the constant feature 1, i.e., $x_0 = z^1_0 = z^2_0 = 1$. As we discussed in the class, they are used to account for the bias parameters. We have the values of all the edge weights in Table 1. Now, given a new input example $\mathbf{x} = [1, 1, 1]$. Please use the forward pass to compute the output y. Please list every step in your computation, namely, how you calculate the variable value in each hidden unit, and how you combine the variables in one layer to compute each variable in the next layer. Please be aware of the subtle difference in computing the variable value in the last layer (we emphasized it in the class).

   The forward pass of the figure 1 neural network requires calculating the values of each neuron in each layer based on the weights between layers.

   For example, the value of $Z^1_1 = \sigma(w^1_{01}x_0 + w^1_{11}x_1 + w^1_{21}x_2)$, where the bias term $x_0 = 1$.

   Following this pattern and starting with x = [1,1,1], we can calculate that

   $Z^1_1 = \sigma(-1 * 1 + -2 * 1 + -3 * 1) = 0.00669$

   $Z^1_2 = \sigma(1 * 1 + 1 * 1 + 3 * 1) = 0.993$

   $Z^1_0 = 1$

   We have calculated the values of layer 1. We follow the same pattern from layer 0 to layer 1, using $Z_0 = [1, 0.00669, 0.993]$:

   $Z^2_1 = \sigma(-1 * 1 + -2 * 0.00669 + -3 * 0.993) = 0.0181$

   $Z^2_2 = \sigma(1 * 1 + 2 * 0.00669 + 3 * 0.993) = 0.981$

   $Z^2_0 = 1$

Finally, we calculate the output y using the layer 2 values of [1, 0.0181, 0.981] using this formula, without the sigmoid:

$y = w_{01}^3 * z_0^2 + w_{11}^3 * z_1^2 + w_{21}^3 * z_2^2$

$y = -1 * 1 + 2 * 0.0181 + -1.5 * 0.981 = -2.4353$

3. [20 points] Suppose we have a training example where the input vector is $\mathbf{x} = [1, 1, 1]$ and the label $y^* = 1$. We use a square loss for the prediction,

$$L(y, y^*) = \frac{1}{2}(y - y^*)^2.$$

To make the prediction, we will use the 3 layer neural network shown in Figure 1, with the sigmoid activation function. Given the weights specified in Table 1, please use the back propagation (BP) algorithm to compute the derivative of the loss L over all the weights, $\{\frac{\partial L}{\partial w_{ij}^m}\}$. Please list every step of your BP calculation. In each step, you should show how you compute and cache the new (partial) derivatives from the previous ones, and then how to calculate the partial derivative over the weights accordingly.

We use a top down approach to calculating the derivative of the loss function with respect to each of the weights.

For the **3rd** layer of weights, we know that

$\frac{\partial L}{\partial w_{01}^3} = \frac{\partial L}{\partial y} * \frac{\partial y}{\partial w_{01}^3}$

To solve this, we know that

$\frac{\partial L}{\partial y} = y - y^* = -2.4353 - 1 = -3.4353$ and $\frac{\partial y}{\partial w_{01}^3} = z_0^2 = 1$.

From here, we can repeat this formula for each weight on the third layer, to find:

$\frac{\partial L}{\partial w_{01}^3} = -3.4353 * (1) = -3.4253,$

$\frac{\partial L}{\partial w_{11}^3} = -3.4353 * (0.0181) = -0.06217893,$

$\frac{\partial L}{\partial w_{21}^3} = -3.4353 * (0.981) = -3.3700293$

On the **2nd** layer of weights, we know that

$\frac{\partial L}{\partial w_{01}^2} = \frac{\partial L}{\partial y} * \frac{\partial y}{\partial z_1^2} * \frac{\partial z_1^2}{\partial w_{01}^2}$

To solve this, we know that $\frac{\partial y}{\partial z_1^2} = w_{11}^3$ and $\frac{\partial z_1^2}{\partial w_{01}^2} = z_0^1$

From here, we can repeat this formula for each weight on the second layer, to find:

$\frac{\partial L}{\partial w_{01}^2} = -3.4353 * (2) * (-1) = 6.8706,$

$\frac{\partial L}{\partial w_{11}^2} = -3.4353 * (2) * (0.00669) = -0.045964314,$

$\frac{\partial L}{\partial w_{21}^2} = -3.4353 * (2) * (0.993) = -6.8225058$

$\frac{\partial L}{\partial w_{01}^2} = -3.4353 * (-1.5) * (-1) = -5.15295,$

$\frac{\partial L}{\partial w_{11}^2} = -3.4353 * (-1.5) * (0.00669) = 0.0344732355,$

$\frac{\partial L}{\partial w_{21}^2} = -3.4353 * (-1.5) * (0.993) = 5.11687935$

On the **1st** layer of weights, we know that

$\frac{\partial L}{\partial w_{01}^1} = \frac{\partial L}{\partial z_1^1} * \frac{\partial z_1^1}{\partial w_{01}^1}$

To solve this, we know that

$\frac{\partial L}{\partial z_1^1} = \frac{\partial L}{\partial z_1^2} * \frac{\partial z_1^2}{\partial z_1^1} + \frac{\partial L}{\partial z_2^2} * \frac{\partial z_2^2}{\partial z_1^1}$

and

$\frac{\partial z_1^1}{\partial w_{01}^1} = x_0$

From here, we can repeat this formula for each weight on the second layer, to find:

$\frac{\partial L}{\partial w_{01}^1} = ((-6.8706) * (-2) + (-5.1529 * 2)) * (1) = 3.4354,$

$\frac{\partial L}{\partial w_{11}^1} = ((-6.8706) * (-2) + (-5.1529 * 2)) * (-2) = -6.8708,$

$\frac{\partial L}{\partial w_{21}^1} = ((-6.8706) * (-2) + (-5.1529 * 2)) * (-3) = -10.3062$

$\frac{\partial L}{\partial w_{02}^1} = ((-6.8706) * (-2) + (-5.1529 * (-3))) * (1) = 29.1999,$

$\frac{\partial L}{\partial w_{12}^1} = ((-6.8706) * (-2) + (-5.1529 * (-3))) * (2) = 58.3998,$

$\frac{\partial L}{\partial w_{22}^1} = ((-6.8706) * (-2) + (-5.1529 * (-3))) * (3) = 87.5997$

This was all done without the sigmoid function derivatives.

4. [10 points] Suppose we have the training dataset shown in Table 2. We want to learn a logistic regression model. We initialize all the model parameters with 0. We assume each parameter (i.e., feature weights $\{w_1, w_2, w_3\}$ and the bias w_0) comes from a standard Gaussian prior distribution,

$$p(w_i) = \mathcal{N}(w_i|0,1) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}w_i^2) \ \ (0 \le i \le 3).$$

- [7 points] We want to obtain the maximum a posteriori (MAP) estimation. Please write down the objective function, namely, the log joint probability, and derive the gradient of the objective function.

  The MAP estimation is maximizing the posterior probability of the model parameters given the data. The log joint probability is

  log p(w, y — X) = log p(y — X, w) + log p(w)

  where w is the model parameters, y is the target values, x is the feature matrix, and p(w) is the prior probability of the model parameters.

  The gradient of the objective function is

$$\nabla_{\mathbf{w}} \log p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \begin{pmatrix} -\frac{1}{2}w_0 + \sum_{i=1}^n (\sigma(z_i) - y_i)x_{i0} \\ -\frac{1}{2}w_1 + \sum_{i=1}^n (\sigma(z_i) - y_i)x_{i1} \\ -\frac{1}{2}w_2 + \sum_{i=1}^n (\sigma(z_i) - y_i)x_{i2} \\ -\frac{1}{2}w_3 + \sum_{i=1}^n (\sigma(z_i) - y_i)x_{i3} \end{pmatrix}$$

- [3 points] We set the learning rates for the first three steps to $\{0.01, 0.005, 0.0025\}$. Please list the stochastic gradients of the objective w.r.t the model parameters for the first three steps, when using the stochastic gradient descent algorithm.

x_1	x_2	x_3	y
0.5	-1	0.3	1
-1	-2	-2	-1
1.5	0.2	-2.5	1

Table 2: Dataset

2 Practice [62 points + 60 bonus]

1. [2 Points] Update your machine learning library. Please check in your implementation of SVM algorithms. Remember last time you created the folders "SVM". You can commit your code into the corresponding folders now. Please also supplement README.md with concise descriptions about how to use your code to run these algorithms (how to call the command, set the parameters, etc). Please create new folders "Neural Networks" and "Logistic Regression" in the same level as these folders. *After the completion of the homework this time, please check in your implementation accordingly.*

2. [58 points] Now let us implement a three-layer artificial neural network for classification. We will use the dataset, "bank-note.zip" in Canvas. The features and labels are listed in the file "classification/data-desc.txt". The training data are stored in the file "classification/train.csv", consisting of 872 examples. The test data are stored in "classification/test.csv", and comprise of 500 examples. In both the training and test datasets, feature values and labels are separated by commas. The architecture of the neural network resembles Figure 1, but we allow an arbitrary number of units in hidden layers (Layer 1 and 2). So please ensure your implementation has such flexibility. We will use the sigmoid activation function.

   (a) [25 points] Please implement the back-propagation algorithm to compute the gradient with respect to all the edge weights given one training example. For debugging, you can use the paper problem 3 and verify if your algorithm returns the same derivatives as you manually did.

   I implemented the back-propagation algorithm to compute the gradient with respect to all edge weights.

   (b) [17 points] Implement the stochastic gradient descent algorithm to learn the neural netowrk from the training data. Use the schedule of learning rate: $\gamma_t = \frac{\gamma_0}{1+\frac{\gamma_0}{d}t}$. Initialize the edge weights with random numbers generated from the standard Gaussian distribution. We restrict the width, i.e., the number of nodes, of each hidden layer (i.e., Layer 1 & 2) to be identical. Vary the width from

$\{5, 10, 25, 50, 100\}$. Please tune γ_0 and d to ensure convergence. Use the curve of the objective function (along with the number of updates) to diagnosis the convergence. Don't forget to shuffle the training examples at the start of each epoch. Report the training and test error for each setting of the width.

(c) [10 points]. Now initialize all the weights with 0, and run your training algorithm again. What is your training and test error? What do you observe and conclude?

(d) [6 points]. As compared with the performance of SVM (and the logistic regression you chose to implement it; see Problem 3), what do you conclude (empirically) about the neural network?

(e) [**Bonus**] [30 points] Please use PyTorch (or TensorFlow if you want) to fulfill the neural network training and prediction. Please try two activation functions, "tanh" and "RELU". For "tanh", please use the "Xavier' initialization; and for "RELU", please use the "he" initialization. You can implement these initializations by yourselves or use PyTorch (or TensorFlow) library. Vary the depth from $\{3, 5, 9\}$ and width from $\{5, 10, 25, 50, 100\}$. Pleas use the Adam optimizer for training. The default settings of Adam should be sufficient (*e.g.,* initial learning rate is set to 10^{-3}). Report the training and test error with each (depth, width) combination. What do you observe and conclude? Note that, we won't provide any link or manual for you to work on this bonus problem. It is YOUR JOB to search the documentation, find code snippets, test, and debug with PyTorch (or TensorFlow) to ensure the correct usage. This is what all machine learning practitioners do in practice.

3. [**Bonus**] [30 points] We will implement the logistic regression model with stochastic gradient descent. We will use the dataset "bank-note.zip" in Canvas. Set the maximum number of epochs T to 100. Don't forget to shuffle the training examples at the start of each epoch. Use the curve of the objective function (along with the number of updates) to diagnosis the convergence. We initialize all the model parameters with 0.

(a) [10 points] We will first obtain the MAP estimation. In order for that, we assume each model parameter comes from a Gaussian prior distribution,

$$p(w_i) = \mathcal{N}(w_i|0, v) = \frac{1}{\sqrt{2\pi v}} \exp(-\frac{1}{2v} w_i^2)$$

where v is the variance. From the paper problem 4, you should be able to write down the objective function and derive the gradient. Try the prior variance v from $\{0.01, 0.1, 0.5, 1, 3, 5, 10, 100\}$. Use the schedule of learning rate: $\gamma_t = \frac{\gamma_0}{1+\frac{\gamma_0}{d}t}$. Please tune γ_0 and d to ensure convergence. For each setting of variance, report your training and test error.

(b) [5 points] We will then obtain the maximum likelihood (ML) estimation. That is, we do not assume any prior over the model parameters, and just maximize the logistic likelihood of the data. Use the same learning rate schedule. Tune γ_0 and d to ensure convergence. For each setting of variance, report your training and test error.

(c) [3 points] How is the training and test performance of the MAP estimation compared with the ML estimation? What can you conclude? What do you think of v, as compared to the hyperparameter C in SVM?

4. [2 Points] After the completion, please upload the implementation to your Github repository immediately. How do you like your own machine learning library? *Although it is still light weighted, it is the proof of your great efforts and achievement in this class! It is an excellent start of your journey to machine learning. Wish you further success in your future endeavours!*

   I like my own machine learning library. It contains some of the fundamentals of machine learning inside, and I believe it is relatively well commented so I can come back, understand it better, and improve upon it. This library should help me well on my final project.