Problem Set 3

October 22, 2024

Instructions: I encourage you to discuss the problem with each other but write your solution individually (copying is penalized). Problems 1, 2, and 3 should be typed or hand-written and submitted to Moodle. Problem 4 should be submitted (as a link) to Slack. Please let me know if there are typos/mistakes in the problem set. The due date is November 1.

Problem 1: Review (20 points)

Answer the following briefly. For either true of false answers, provide a brief explanation.

1. Backpropagation is a technique used to calculate the loss in a neural network.
2. An neuron, in a neural network, can approximate any logical operation (AND, NOR, NOT, etc.).
3. A Softmax activation is used at the output of a neural network when the value of each prediction in an n-dimensional vector has to be between 0 and 1.
4. A deep network is just a neural network with more neurons in the hidden layer.
5. An autoencoder is an unsupervised neural network architecture because it doesn't require labels.
6. A deep network with 2 hidden layers (each 3 neurons), 1 input and 2 outputs, has 10 fitting parameters. If false provide the correct answer with a brief explanation (consider biases separately).
7. The ReLU activation function is more commonly used than a sigmoid because it doesn't suffer from vanishing gradients and it's faster.
8. L1 and L2 regularization are used in neural networks because they are prone to overfitting.
9. Estimation error increases with the number of neurons in a neural network because it becomes harder for the optimization algorithm to find the optimal model within the hypothesis class.
10. The purpose of a computation graph in deep learning is to make the computation of gradients easier.

Problem 2: Neural Networks (30 points)

Let $X = \{x^{(1)}, \ldots, x^{(m)}\}$ be a dataset of m samples with 2 features, i.e. $x^{(i)} \in \mathbb{R}^2$. The samples are classified into 2 categories with labels $y^{(i)} \in \{0, 1\}$. A scatter plot of the dataset is shown in Figure 1:

The examples in class 1 are marked as $\times$ and examples in class 0 are marked as $\circ$. We want to perform a binary classification using a neural network with the architecture shown in Figure 2

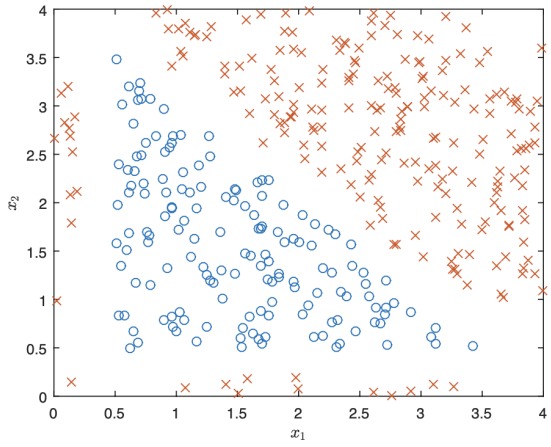

Figure 1: Neural Network Architecture

Denote the two features x_1 and x_2, the three neurons in the hidden layer a_1, a_2, and a_3, and the output neuron as $\hat{y}$. Let the weight from x_i to a_j be $w_{ij}^{(1)}$ for $i \in \{1, 2\}$, $j \in \{1, 2, 3\}$, and the weight from a_j to $\hat{y}$ be $w_j^{(2)}$. Finally, denote the intercept weight (i.e. bias) for a_j as $w_{0j}^{(1)}$, and the intercept weight for $\hat{y}$ as $w_0^{(2)}$. For the loss function, we'll use average squared loss:

$$L(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)^2 \tag{1}$$

where $\hat{y}^{(i)}$ is the result of the output neuron for example i.

1. Suppose we use the sigmoid function as the activation function for a_1, a_2, a_3 and $\hat{y}$. What is the gradient descent update to $w_{12}^{(1)}$, assuming we use a learning rate of η? Your answer should be written in terms of $x^{(i)}$, $\hat{y}^{(i)}$, $y^{(i)}$. and the weights. (Hint: remember that $\sigma'(x) = \sigma(x)(1-\sigma(x))$).

2. Now, suppose instead of using the sigmoid function for the activation function a_1, a_2, a_3, and $\hat{y}$, we instead use the step function $f(x)$, defined as

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2}$$

   What is one set of weights that would allow the neural network to classify this dataset with 100% accuracy? Please specify a value for the weights in the following order and explain your reasoning:

$$w_{01}^{(1)}, w_{11}^{(1)}, w_{21}^{(1)}, w_{02}^{(1)}, w_{12}^{(1)}, w_{22}^{(1)}, w_{03}^{(1)}, w_{13}^{(1)}, w_{23}^{(1)}, w_0^{(2)}, w_1^{(2)}, w_2^{(2)}, w_3^{(2)}$$

   Hint: There are three sides to a triangle, and there are three neurons in the hidden layer.

3. Let the activation functions for a_1, a_2, a_3 be the linear function $f(x) = x$ and the activation for $\hat{y}$ be the same step function as before. Is there a specific set of weights that will make the loss 0? If yes, please explicitly state a value for every weight. If not, please explain your reasoning.

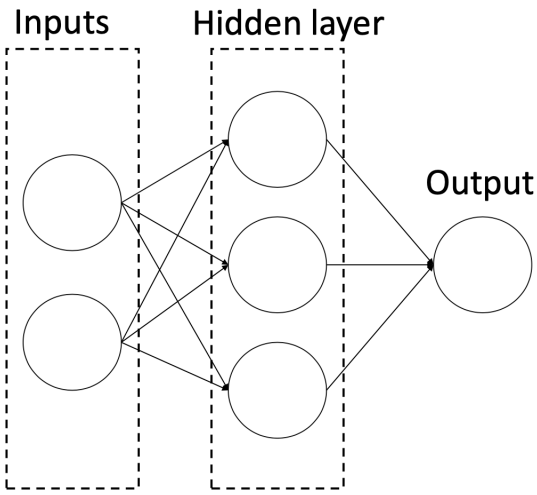

Figure 2: Neural Network Architecture

Problem 3: Neural Network Architectures (30 points)

This is an exploratory problem. Provide brief explanatory answers.

1. Can a fully connected feed-forward neural network be used for predicting the next time step in a time series data? That is, given the time series data for $\{x(t_1), x(t_2), x(t_3), \ldots, x(t_n)\}$, where $x(t) = [x_1, x_2, \ldots, x_d](t) \in \mathbb{R}^d$, can a fully connected network f take in previous time steps $[x(t_i), x(t_{i-1}), \ldots]$ to predict $x(t_{i+1})$? If not, explain why, and if yes, provide more details on the architecture of the neural network (i.e. dimensions of the inputs, outputs, size of weights, etc.).

2. Time series data are often modeled with differential equations of the form

$$\frac{dx}{dt} = \mathbf{g}(x) \tag{3}$$

   where $\mathbf{g}$ can be a nonlinear function in x (e.g. Lorenz system). Discretize the differential equation (i.e. setting $dx/dt \approx (x_i - x_{i-1})/(t_i - t_{i-1})$) and briefly compare the difference between a differential equation model and neural network model.

3. If you were to employ a vanilla recurrent neural network (RNN) to model the time series, what would the inputs and outputs be? Draw a small diagram.

4. Although not common, a convolutional neural network (CNN) can be used for time series data. For example given the 5 previous time steps $[x_i, x_{i-1}, x_{i-2}, x_{i-3}, x_{i-4}]$, one can apply a convolutional operation to the time series before passing it through a feedforward neural network f, as done in the first part, to predict the next time step x_{i+1}. This has applications in anomaly detection, stock price prediction, speech recognition, etc. Describe the structure of a CNN applied to time series data. Specifically, detail:

   - The dimensionality of the input, filters, and feature maps.
   - How the convolutional operation works in this case.

- How the filter size affects the predictions and what patterns the network is expected to learn.
- The steps that follow after the convolution, leading to the prediction of the next time step.

Feel free to visualize the architecture if it helps your explanation.

Problem 4: Deep Learning and GitHub (20 points)

Nowadays, the best way to make your work known and useful to others, is by posting your work on github.com. If you don't have an account on github, create one and learn the basics of pulling and pushing your repository on the platform. This is an open problem whose main purpose to get you familiar with GitHub and a simple neural network implementation.

Find a simple dataset (Kaggle, huggingface, etc.) and apply a feed-forward neural network learning algorithm to it. If you're using a supervised learning approach, look for a dataset that already has input-output pairs. Explain the model, dataset, etc, of your solution in the README.md file (which will be displayed on the homepage of your github). The README.md file should also contain details on how the code should be used. For that, it's preferable if you have a file containing the functions your code uses (e.g. a NeuralNetwork class, a BuildDataset function, etc.) and a Jupyter notebook showing your testcase, so that others who want to use your package, can easily import the functions/modules to be applied to their dataset.

For this problem, don't complicate your solution, although as usual, I encourage you to be creative with it (applying different architectures and discussing the differences if needed). It should be done individually and the link of the code should be posted on the #deep-learning-github channel on slack. Don't forget to make the github page public.