EECS 182 Deep Neural Networks
Fall 2022 Discussion Worksheet
Discussion 1

This discussion serves as an introduction to neural networks. You will examine a simple example of gradient descent, and build intuition for the ReLU nonlinearity through visualizations in the Jupyter [notebook]. This notebook takes a long time to do the initial network training! You should begin the training process then return to the theory part of this discussion while you wait.

1. Gradient Descent Mechanics

Gradient descent is the primary algorithm to search optimal parameters for our models. Typically, we want to solve optimization problems stated as

$$\min_{\theta \in \Theta} \mathcal{L}(f_\theta, \mathcal{D})$$

where $\mathcal{L}$ are differentiable functions. In this example, we look at a simple supervised learning problem where given a dataset $\mathcal{D} = \{(x_i, y_i)\}_i^N$, we want to find the optimal parameters θ that minimizes some loss. We consider different models for learning the mapping from input to output, and examine the behavior of gradient descent for each model.

(a) The simplest parameteric model entails learning a single-parameter constant function, where we set $\hat{y}_i = \theta$. We wish to find

$$\hat{\theta}_{\text{const}} = \min_{\theta \in \mathbb{R}} \mathcal{L}(f_\theta, \mathcal{D}) = \min_{\theta \in \mathbb{R}} \frac{1}{N} \sum_{i=1}^{N} (y_i - \theta)^2$$

- What is the gradient of $\mathcal{L}$ w.r.t θ?
- What is the optimal value of θ?
- Write the gradient descent update.
- *Stochastic Gradient Descent (SGD)* is an alternative optimization algorithm, where instead of using all N samples, we use single sample per optimization step to update the model. What is the contribution of each data-point to the full gradient update?

Solution:

- Taking the gradient of $\mathcal{L}$ w.r.t θ we have

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{2}{N} \sum_{i=1}^{N} (\theta - y_i)$$

- From the first-order optimality condition, at θ^* we have $\frac{\partial \mathcal{L}}{\partial \theta} = 0$. Using this, we get

$$\frac{2}{N} \sum_{i=1}^{N} (\theta^* - y_i) = 0$$

$$\implies \theta^* = \frac{1}{N} \sum_{i=1}^{N} y_i$$

- The gradient descent update follows the rule

$$\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}}{\partial \theta}$$

where α is the step-size. Plugging in the gradient, we have

$$\theta \leftarrow \theta - \frac{2\alpha}{N} \sum_{i=1}^{N} (\theta - y_i)$$

- Using SGD, we update the model by uniformly sampling a single data-point per iteration, with the following update rule

$$\theta \leftarrow \theta - 2\alpha(\theta - y_i)$$

Note that the term $(y_i - \theta)$ in this example measures the error of the model on the i-th data-point. The full gradient can be written as

$$\nabla_\theta \mathcal{L} = \frac{2}{N}(\theta - y_1) + \frac{2}{N}(\theta - y_2) + \frac{2}{N}(\theta - y_3) + \cdots + \frac{2}{N}(\theta - y_N)$$

which suggests that each data-point contributes equally to the full gradient. In other words, if there are two points with the same *error*, their contribution to the full gradient is the same (irrespective of x_i)

(b) Instead of constant functions, we now consider a single-parameter linear model $\hat{y}(x_i) = \theta x_i$, where we search for θ such that

$$\hat{\theta} = \min_{\theta \in \mathbb{R}} \frac{1}{N} \sum_{i=1}^{N} (y_i - \theta x_i)^2$$

- Repeat questions from part (a).
- Do all points get the same *vote* in the update? Why?

Solution:

- Taking the gradient of $\mathcal{L}$ w.r.t θ we have

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{2}{N} \sum_{i=1}^{N} (\theta x_i - y_i) x_i$$

- From the first-order optimality condition, at θ^* we have $\frac{\partial \mathcal{L}}{\partial \theta} = 0$. Using this, we get

$$\frac{2}{N} \sum_{i=1}^{N} (y_i - \theta x_i) x_i = 0$$

$$\implies \theta^* = \frac{\sum_{i=1}^{N} x_i y_i}{\sum_{i=1}^{N} x_i^2}$$

- The gradient descent update follows the rule

$$\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}}{\partial \theta}$$

where α is the step-size. Plugging in the gradient, we have

$$\theta \leftarrow \theta - \frac{2\alpha}{N}\sum_{i=1}^{N}(\theta x_i - y_i)x_i$$

- Using SGD, we update the model by uniformly sampling a single data-point per iteration, with the following update rule

$$\theta \leftarrow \theta - 2\alpha(\theta x_i - y_i)x_i$$

The full gradient can be written as

$$\nabla_\theta \mathcal{L} = \frac{2}{N}(\theta x_1 - y_1)x_1 + \frac{2}{N}(\theta x_2 - y_2)x_2 + \frac{2}{N}(\theta x_3 - y_3)x_3 + \cdots + \frac{2}{N}(\theta x_N - y_N)x_N$$

As before the term $(y_i - \theta x_i)$ in this example measures the error of the model on the i-th data-point. However, the contribution of each data-point to the full gradient is now proportional to x_i. In other words, if there are two points with the same *error*, their contribution to the full gradient is proportional to magnitude of x_i.

2. ReLU SGD Visualization

Work through the notebook to explore how a simple network with ReLU non-linearities adapts to model a function with SGD updates. training the networks takes 5-10 minutes depending on whether you run locally or the server, so you should start the training process (run through the train all layers cell) then return to the theory part of the discussion while training occurs.

As you walk through the notebook, pay attention to how the slopes and elbows of the ReLU functions change during training and how they impact the shape of the final learned function.

Contributors:

- Kumar Krishna Agrawal.
- Anant Sahai.
- Ashwin Pananjady.
- Josh Sanz.
- Yichao Zhou.

