EECS 182 Deep Neural Networks
Fall 2025 Anant Sahai and Gireeja Ranade

Discussion 1

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

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

i. What is the gradient of $\mathcal{L}$ w.r.t. θ ?

ii. What is the optimal value of θ ?

iii. Write the gradient descent update.

iv. *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?

(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

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

i. What is the gradient of $\mathcal{L}$ w.r.t. θ ?

ii. What is the optimal value of θ ?

iii. Write the gradient descent update.

iv. Do all points get the same *vote* in the update? Why or why not? (*Hint*: repeat the derivation of the last bullet point in part(a).)

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 on a server, so you should start the training process (ie. run through the *train all layers* cell) then return to the theory part of the discussion while training occurs. The link to run the notebook on Google Colab is https://tinyurl.com/cs182-dis01-code

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

Contributors:

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

