

EECS 182 Deep Neural Networks

Fall 2026 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?

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

$$\begin{aligned} \frac{2}{N} \sum_{i=1}^N (\theta^* - y_i) &= 0 \\ \implies \theta^* &= \frac{1}{N} \sum_{i=1}^N y_i \end{aligned}$$

- 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

$$\theta \leftarrow \theta - \frac{2\alpha}{N} (\theta - y_i)$$

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.

- (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

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

- What is the gradient of $\mathcal{L}$ w.r.t. θ ?
- What is the optimal value of θ ?
- Write the gradient descent update.
- 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).)

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

$$\begin{aligned} \frac{2}{N} \sum_{i=1}^N (y_i - \theta x_i) x_i &= 0 \\ \implies \theta^* &= \frac{\sum_{i=1}^N x_i y_i}{\sum_{i=1}^N x_i^2} \end{aligned}$$

- 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

$$\theta \leftarrow \theta - \frac{2\alpha}{N} (\theta x_i - y_i) x_i$$

The full gradient can be written as

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

The contribution of each data-point to the full gradient depends on the size of the data-point x_i , i.e. data-points closer to the origin contribute less to the full gradient.

2. Two Matrix Norms from the SVD

Let $A \in \mathbb{R}^{m \times n}$ have rank r and singular value decomposition

$$A = U \Sigma V^\top,$$

where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal ($U^\top U = I_m$, $V^\top V = I_n$) and $\Sigma \in \mathbb{R}^{m \times n}$ is zero off the main diagonal, with diagonal entries the singular values

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0, \quad \sigma_i = 0 \text{ for } i > r.$$

Write $u_1, \dots, u_m$ and $v_1, \dots, v_n$ for the columns of U and V .

We will use two ways of measuring the “size” of A : the **Frobenius norm**, which treats A as a long vector of its entries,

$$\|A\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n A_{ij}^2,$$

and the **spectral norm**, which measures how much A can stretch a vector,

$$\|A\|_2 = \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \max_{\|x\|_2=1} \|Ax\|_2.$$

The goal of this problem is to show that the SVD reads off both at a glance:

$$(1) \quad \|A\|_F^2 = \sum_{i=1}^r \sigma_i^2, \quad (2) \quad \|A\|_2^2 = \sigma_1^2.$$

- (a) The Frobenius norm is defined entrywise, which on its face has nothing to do with the SVD. The trace gives us the bridge, and everything else in this problem follows from it.

Show that $\|A\|_F^2 = \text{Tr}(A^\top A)$, and use this to show that for any orthogonal $Q \in \mathbb{R}^{m \times m}$ and orthogonal $P \in \mathbb{R}^{n \times n}$,

$$\|QA\|_F = \|A\|_F = \|AP\|_F.$$

Show also that $\|Qx\|_2 = \|x\|_2$ for every $x \in \mathbb{R}^m$.

Solution: The (j, j) entry of $A^\top A$ is the squared norm of the j -th column of A , so summing over j collects every entry of A exactly once:

$$\text{Tr}(A^\top A) = \sum_{j=1}^n (A^\top A)_{jj} = \sum_{j=1}^n \sum_{i=1}^m A_{ij}^2 = \|A\|_F^2.$$

Now use $Q^\top Q = I_m$, $PP^\top = I_n$, and the cyclic property $\text{Tr}(MN) = \text{Tr}(NM)$:

$$\begin{aligned} \|QA\|_F^2 &= \text{Tr}((QA)^\top QA) = \text{Tr}(A^\top Q^\top QA) = \text{Tr}(A^\top A) = \|A\|_F^2, \\ \|AP\|_F^2 &= \text{Tr}((AP)^\top AP) = \text{Tr}(P^\top A^\top AP) = \text{Tr}(A^\top APP^\top) = \text{Tr}(A^\top A) = \|A\|_F^2. \end{aligned}$$

The same one-line computation on a vector gives $\|Qx\|_2^2 = x^\top Q^\top Qx = x^\top x = \|x\|_2^2$.

- (b) **Using part (a), show that $\|A\|_F^2 = \sum_{i=1}^r \sigma_i^2$.**

Solution: Peel off the two orthogonal factors one at a time, each time invoking part (a):

$$\|A\|_F = \|U\Sigma V^\top\|_F = \|\Sigma V^\top\|_F = \|\Sigma\|_F.$$

Since Σ is zero off the diagonal, its only nonzero entries are $\sigma_1, \dots, \sigma_r$, so

$$\|A\|_F^2 = \|\Sigma\|_F^2 = \sum_{i=1}^r \sigma_i^2.$$

(c) **Using part (a), show that $\|A\|_2^2 = \sigma_1^2$, and identify a unit vector x that attains the maximum.**

Solution: Fix a unit vector x and substitute the SVD. Since U is orthogonal it does not change lengths, and the change of variables $y = V^\top x$ is a bijection of the unit sphere onto itself ($\|y\|_2 = \|x\|_2 = 1$, again by part (a)):

$$\|Ax\|_2^2 = \|U\Sigma V^\top x\|_2^2 = \|\Sigma y\|_2^2 = \sum_{i=1}^r \sigma_i^2 y_i^2.$$

So $\|A\|_2^2$ is the largest that this weighted average of the σ_i^2 can be. Bounding every weight by the largest one,

$$\sum_{i=1}^r \sigma_i^2 y_i^2 \leq \sigma_1^2 \sum_{i=1}^r y_i^2 \leq \sigma_1^2 \|y\|_2^2 = \sigma_1^2,$$

which gives $\|A\|_2^2 \leq \sigma_1^2$. The bound is attained: taking $y = e_1$, i.e. $x = Ve_1 = v_1$, gives

$$Av_1 = U\Sigma V^\top v_1 = U\Sigma e_1 = \sigma_1 u_1, \quad \|Av_1\|_2 = \sigma_1 \|u_1\|_2 = \sigma_1.$$

Hence $\|A\|_2 = \sigma_1$, achieved at the top right singular vector $x = v_1$, and the output direction is the top left singular vector u_1 .

(d) **Combine the two results to show that $\|A\|_2 \leq \|A\|_F \leq \sqrt{r} \|A\|_2$. When is each inequality tight?**

Solution: Since σ_1 is the largest of the r singular values,

$$\sigma_1^2 \leq \sum_{i=1}^r \sigma_i^2 \leq r \sigma_1^2,$$

and taking square roots gives $\|A\|_2 \leq \|A\|_F \leq \sqrt{r} \|A\|_2$.

The left inequality is tight exactly when $\sigma_2 = \dots = \sigma_r = 0$, i.e. when A has rank one: all of the matrix's “energy” sits in a single direction. The right inequality is tight exactly when $\sigma_1 = \dots = \sigma_r$, i.e. when A acts as a scaled orthogonal map on its row space and spreads its energy evenly across all r directions.

3. 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 [q_relu_vis.ipynb](#)

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.

Solution:

Answers to notebook questions:

- *What is causing test error to be lower than train error?* **Ans:** We don't add noise in the test samples, which results in lower effective test error compared to training error.
- *Train Output Layer Weights Only*
 - (a) *Hidden layer's impact?* **Ans:** The larger the hidden layer is, the lower the test error (no overfitting here if any student ask) and the better learnt function we get.
 - (b) *Hand-pick the elbow locations?* **Ans:** At the discontinuities. Having an elbow at each kink is sufficient to capture the non-linearities in this example.
 - (c) *How do the final test error and learned function compare to the ridge regression version of training?* **Ans:** Ridge regression version is faster and has a lower test error.
- *Train All Layers*
 - (a) *Hidden layer's impact?* **Ans:** Same as the above.
 - (b) *Elbow positions?* **Ans:** Gradually converge to the kinks of the function, although the elbows don't align to the exact positions of the kinks.
 - (c) *How do the final test error and learned function compare to the ridge regression version of training? Which one is more efficient?* **Ans:** The answer differs depending on the width. For the test error, in the case of $width=10$, we see:

$$AllLayers \approx OutputLayerOnly > Ridge$$

In the case of $width=20$:

$$OutputLayerOnly > Ridge > AllLayers$$

As for the efficiency (ie. training time), *Training All Layers* is the slowest since it requires optimizing more parameters. *Ridge on Output Layer* is the fastest in this example since it can be solved in a closed form solution, whereas in *Training Output Layer* we are using SGD.

Contributors:

- Ashwin Pananjady.
- Josh Sanz.
- Yichao Zhou.
- Anant Sahai.
- Gireeja Ranade.

- Matteo Guarrera.
- Mert Cemri.
- Sizhe Chen.
- Kumar Krishna Agrawal.
- Kevin Li.