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

Discussion 3

1. Optimizers as Penalized Linear Improvement

In lecture, you saw the locally linear perspective of a neural network and the loss by Taylor expanding the loss around the current value of the parameters. This approximation is only very good in a near neighborhood of those values. One way to proceed with optimization is to consider the size of the neighborhood as a hyperparameter and to bound our update to stay within that neighborhood while minimizing our linear approximation to the loss. You saw in lecture that the choice of norm in defining that neighborhood also matters.

In this problem (and the homework), you will work out for yourself a slightly different perspective. Instead of treating the norm as a constraint (with the size of the acceptable norm as a hyperparameter), we can do an unconstrained optimization with a weighted penalty that corresponds to the squared norm — where that weight is a hyperparameter.

At each iteration, we wish to maximize linear improvement of the objective (as defined by the dot-product between the gradient and the update) locally regularized by a penalty on the size of the update. This can be expressed (in traditional minimization form) as:

$$u = \underset{\Delta\theta}{\operatorname{argmin}} \quad \underbrace{g^T \Delta\theta}_{\text{Linear Improvement}} \quad + \quad \frac{1}{\alpha} \quad \underbrace{d(\Delta\theta)}_{\text{Distance Penalty}} \quad , \tag{1}$$

where $g = \nabla f(\theta)$ is the gradient of the loss, α is a scalar, and d is a scalar-output distance function $\mathbb{R}^{dim(\theta)} \to \mathbb{R}^+$.

Let's assume *Euclidean distance* is the norm that captures our sense of relevant neighborhoods in parameter space. Then we can be interested in:

$$u = \underset{\Delta\theta}{\operatorname{argmin}} \; g^T \Delta\theta \; + \frac{1}{\alpha} \|\Delta\theta\|_2^2. \tag{2}$$

What is the analytical solution for u in the above problem? What standard optimizer does this recover?

Solution: To solve for u, we can take the gradient of the objective with respect to $\Delta\theta$ and set it to zero:

$$\nabla_{\Delta\theta} \left(g^T \Delta\theta + \frac{1}{\alpha} \|\Delta\theta\|_2^2 \right) = g + \frac{2}{\alpha} \Delta\theta = 0. \tag{3}$$

Then, solving for $\Delta\theta$, we get:

$$\Delta\theta = -\frac{\alpha}{2} g, \tag{4}$$

which is simply a scaled version of the negative gradient. This solution is equivalent to standard gradient descent with a step size of $\frac{\alpha}{2}$. SGD can be understood as maximizing linear improvement subject to a local

Euclidean distance penalty over the change in parameters.

2. RMS Norm

We have learned how SGD and Adam can be seen as constrained optimization problems, defined by some norm over parameter space. In this discussion, we will explore how it is helpful to have suitable norms over the *output features* of a neural net layer.

(a) The Euclidean norm is often utilized to define the "scale" of a feature vector:

$$||\mathbf{x}||_2 = \sqrt{\sum_i x_i^2}.$$

However, the *root-mean-squared* (RMS) norm:

$$||\mathbf{x}||_{RMS} = \sqrt{\frac{1}{d}\sum_i x_i^2},$$

is often more appropriate when we care about the average scales of *individual feature elements*.

How do the two norms scale with dimension d (i.e. the number of elements in x), assuming each element is sampled from a standard Gaussian distribution?

Solution: For the Eucliean norm, we have:

$$\mathbb{E}\left[||x||_2\right] = \mathbb{E}\left[\sqrt{\sum_i x_i^2}\right] \leq \sqrt{\sum_i \mathbb{E}\left[x_i^2\right]} = O(\sqrt{d}) \tag{5}$$

where the inequality is from Jensen's inequality.

The RMS norm is simply the Euclidean norm divided by $\sqrt{d}$. This means that for an d-length vector, if each element of $\mathbf{x}$ is sampled from a standard Gaussian, the expected Euclidean norm scales with $O(\sqrt{d})$, while the expected RMS norm is constant $O(1)$. Thus, the RMS norm better captures the scale of individual elements.

(b) **For a vector space with dimension $d = 2$, draw out the set of points with distance 1 under each norm below.**

- The Euclidean norm (or 2-norm):
$$||\mathbf{x}||_2 = \sqrt{\sum_i x_i^2}.$$
- The RMS Norm:
$$||\mathbf{x}||_{RMS} = \sqrt{\frac{1}{d}\sum_i x_i^2}.$$
- The 1-Norm:
$$||\mathbf{x}||_1 = \sum_i |x_i|.$$
- The Infinity Norm:
$$||\mathbf{x}||_\infty = \max_i |x_i|.$$

Solution:

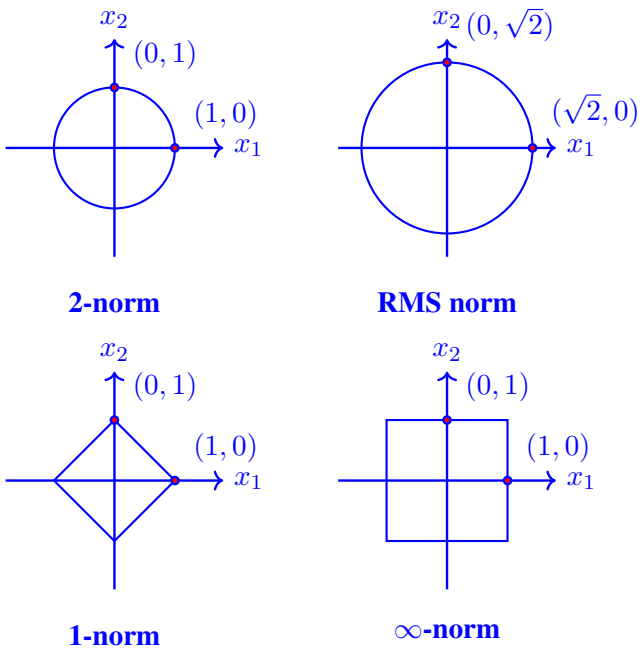

Figure 1: Unit balls for different norms in 2D. Each plot shows the set of points with distance 1 from the origin under the respective norm.

(c) Consider a neural net layer with input $\mathbf{x} \in \mathbb{R}^{d_1}$, weights $W \in \mathbb{R}^{d_2 \times d_1}$, where W is initialized with i.i.d. standard Gaussian entries, and entries of $\mathbf{x}$ are sampled from i.i.d. unit Gaussians.

What is the expected squared RMS norm of the output features $\mathbf{y} = W\mathbf{x}$? How does this scale with d_1 or d_2? What constant should we multiply W by to ensure that the expected squared RMS norm of $W\mathbf{x}$ is 1, regardless of d_1 and d_2?

Hint: Consider a simplified layer with a single output feature, $W \in \mathbb{R}^{1 \times d_1}$. What is the variance of the scalar $y = W\mathbf{x}$?

Solution: Let us first consider a single element of $\mathbf{y}$. We have:

$$\mathbb{E}\left[y_i^2\right] = \mathbb{E}\left[\left(\sum_j W_{ij}x_j\right)^2\right] = \sum_j \mathbb{E}\left[(W_{ij}x_j)^2\right] = \sum_j \mathbb{E}\left[W_{ij}^2\right]\mathbb{E}\left[x_j^2\right] = d_1\,. \tag{6}$$

Here, we use that $\mathbb{E}\left[W_{ij}\right] = \mathbb{E}\left[x_j\right] = 0$ for all j.
Substituting this in the squared RMS norm yields:

$$\mathbb{E}\left[||\mathbf{y}||_{RMS}^2\right] = \mathbb{E}\left[\frac{1}{d_2}\sum_i y_i^2\right] = d_1 \tag{7}$$

To ensure that the expected squared RMS norm is 1, we should multiply W by $\frac{1}{\sqrt{d_1}}$. This recovers the Xavier initialization scaling.
Alternatively, we could notice that since y_i has mean 0, the squared RMS norm of $\mathbf{y}$ is exactly equal to its variance. Since we have $\mathsf{Var}\left[W_{ij}\right] = \mathsf{Var}\left[x_j\right] = 1$, we get $\mathsf{Var}\left[y_i\right] = d_1$. This yields the same solution.

3. Optimizers and their convergence

In this question, we will examine how various optimizers converge to different points when there is a manifold of parameter values that all achieve zero training loss.

For this part, we have exactly $n = 1$ training point corresponding to the single equation

$$[1, 0.1, 0.01]\boldsymbol{\theta} = 1 \tag{8}$$

with a 3-dimensional column vector of parameters $\boldsymbol{\theta}$. Suppose that we start with $\boldsymbol{\theta}_0 = \mathbf{0}$ and use squared loss $f_t(\boldsymbol{\theta}) = (1 - [1, 0.1, 0.01]\boldsymbol{\theta})^2$.

- **What specific vector θ^* would standard vanilla SGD converge to assuming the learning rate was small enough to give convergence?**

  Solution: SGD in this case is identical to gradient descent since there is only one training point. To get convergence with a constant step size, we need the gradient itself to get to zero.

  Computing the gradient yields

  $$\nabla f_t(\boldsymbol{\theta}) = -2(1 - [1, 0.1, 0.01]\boldsymbol{\theta}) \begin{bmatrix} 1 \\ 0.1 \\ 0.01 \end{bmatrix} \tag{9}$$

  We see that $-2(1 - [1, 0.1, 0.01]\boldsymbol{\theta})$ is a scalar, and $\begin{bmatrix} 1 \\ 0.1 \\ 0.01 \end{bmatrix}$ defines the direction that $\boldsymbol{\theta}$ is allowed to move in.

  Therefore, gradient-descent on squared-loss can only move $\boldsymbol{\theta}$ in the direction of $\begin{bmatrix} 1 \\ 0.1 \\ 0.01 \end{bmatrix}$ and given that we start at $\boldsymbol{\theta} = \mathbf{0}$, this means that we are looking for a solution of the form $\boldsymbol{\theta} = \beta \begin{bmatrix} 1 \\ 0.1 \\ 0.01 \end{bmatrix}$. Plugging this into (8), we get

  $$\beta[1, 0.1, 0.01] \begin{bmatrix} 1 \\ 0.1 \\ 0.01 \end{bmatrix} = 1 \tag{10}$$

  which has the solution $\beta = \frac{1}{1.0101}$ giving us the overall answer $\boldsymbol{\theta^*} = \frac{1}{1.0101} \begin{bmatrix} 1 \\ 0.1 \\ 0.01 \end{bmatrix}$.

  Notice that this is quite insensitive to the second and third coordinates, and if the test-data is like the training data in having small numbers there, the overall impact of anything there will be quite small indeed.

- **What specific vector θ_2^* would signSGD converge to assuming an appropriate learning rate schedule to give convergence?**

Solution: Note that the sign of the gradient is given by:

$$\nabla f_t(\boldsymbol{\theta}) = \begin{bmatrix} \mathsf{sgn}\left(-2(1-[1,0.1,0.01]\boldsymbol{\theta})\right) \\ \mathsf{sgn}\left(-0.2(1-[1,0.1,0.01]\boldsymbol{\theta})\right) \\ \mathsf{sgn}\left(-0.02(1-[1,0.1,0.01]\boldsymbol{\theta})\right) \end{bmatrix} = c \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{11}$$

for some scalar c.

Now, we can apply the same argument applies as the previous part. So we get that the solution must take the form $\boldsymbol{\theta} = \beta \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ Again, we can solve

$$\beta[1,0.1,0.01] \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 1 \tag{12}$$

which has the solution $\beta = \frac{1}{1.11}$ giving us the overall answer $\boldsymbol{\theta}_2^* = \frac{1}{1.11} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

Note that to get this convergence, we need to use a diminishing step size schedule for this highly simplified version of Adam. A constant step size wouldn't work.

Further notice that this solution $\boldsymbol{\theta}_2^*$ is significantly more sensitive to the second and third coordinates, but the weight on the first coordinate is not that different from the previous SGD solution. Consequently, as long as the test-data is like the training data in having small numbers in the second and third coordinate, by that very nature, they won't impact the final prediction by much.

Contributors:

- Kevin Frans.
- Anant Sahai.
- Gireeja Ranade.
- Luke Jaffe.
- Kevin Li.

