EECS 182 Deep Neural Networks
Fall 2023 Anant Sahai

Homework 2

This homework is due on Sunday, September 10, 2023, at 10:59PM.

1. Why Learning Rates Cannot be Too Big

To understand the role of the learning rate, it is useful to understand it in the context of the simplest possible problem first.

Suppose that we want to solve the scalar equation

$$\sigma w = y \tag{1}$$

where we know that $\sigma > 0$. We proceed with an initial condition $w_0 = 0$ by using gradient descent to minimize the squared loss

$$L(w) = (y - \sigma w)^2 \tag{2}$$

which has a derivative with respect to the parameter w of $-2\sigma(y - \sigma w)$.

Gradient descent with a learning rate of η follows the recurrence-relation or discrete-time state evolution of:

$$\begin{aligned} w_{t+1} &= w_t + 2\eta\sigma(y - \sigma w_t) \\ &= (1 - 2\eta\sigma^2)w_t + 2\eta\sigma y. \end{aligned} \tag{3}$$

(a) **For what values of learning rate $\eta > 0$ is the recurrence (3) stable?**
(HINT: Remember the role of the unit circle in determining the stability or instability of such recurrences. If you keep taking higher and higher positive integer powers of a number, what does that number has to be like for this to converge?)

(b) The previous part gives you an upper bound for the learning rate η that depends on σ beyond which we cannot safely go. **If η is below that upper bound, how fast does w_t converge to its final solution $w^* = \frac{y}{\sigma}$? i.e. if we wanted to get within a factor $(1 - \epsilon)$ of w^*, how many iterations t would we need?**
(HINT: The absolute value of the error of current w to the optimality might help.)

(c) Suppose that we now have a vector problem where we have two parameters $w[1], w[2]$. One with a large σ_ℓ and the other with a tiny σ_s. i.e. $\sigma_\ell \gg \sigma_s$ and we have the vector equation we want to solve:

$$\begin{bmatrix} \sigma_\ell & 0 \\ 0 & \sigma_s \end{bmatrix} \begin{bmatrix} w[1] \\ w[2] \end{bmatrix} = \begin{bmatrix} y[1] \\ y[2] \end{bmatrix}. \tag{4}$$

We use gradient descent with a single learning rate η to solve this problem starting from an initial condition of $\mathbf{w} = \mathbf{0}$.
For what learning rates $\eta > 0$ will we converge? Which of the two σ_i is limiting our learning rate?

(d) **For the previous problem, depending on $\eta, \sigma_\ell, \sigma_s$, which of the two dimensions is converging faster and which is converging slower?**

(e) The speed of convergence overall will be dominated by the slower of the two. **For what value of η will we get the fastest overall convergence to the solution?**

(f) Comment on what would happen if we had more parallel problems with σ_i that all were in between σ_ℓ and σ_s? **Would they influence the choice of possible learning rates or the learning rate with the fastest convergence?**

(g) Using what you know about the SVD, **how is the simple scalar and parallel scalar problem analysis above relevant to solving general least-squares problems of the form $X\mathbf{w} \approx \mathbf{y}$ using gradient descent?**

2. Accelerating Gradient Descent with Momentum

Consider the problem of finding the minimizer of the following objective:

$$\mathcal{L}(w) = ||y - Xw||_2^2 \tag{5}$$

In the previous homework, we proved that gradient descent (GD) algorithm can converge and derive the convergence rate. In this homework, we will add the momentum term and how it affects to the convergence rate. The optimization procedure of gradient descent+momentum is given below:

$$\begin{aligned} w_{t+1} &= w_t - \eta z_{t+1} \\ z_{t+1} &= (1-\beta) z_t + \beta g_t, \end{aligned} \tag{6}$$

where $g_t = \nabla \mathcal{L}(w_t)$, η is learning rate and β defines how much averaging we want for the gradient. Note that when $\beta = 1$, the above procedure is just the original gradient descent.

Let's investigate the effect of this change. We'll see that this modification can actually 'accelerate' the convergence by allowing larger learning rates.

(a) Recall that the gradient descent update of 5 is

$$w_{t+1} = \left(I - 2\eta(X^T X)\right) w_t + 2\eta X^T y \tag{7}$$

and the minimizer is

$$w^* = (X^T X)^{-1} X^T y \tag{8}$$

The geometric convergence rate (in the sense of what base is there for convergence as rate^t) of this procedure is

$$\text{rate} = \max_i |1 - 2\eta\sigma_i^2| \tag{9}$$

You saw on the last homework that if we choose the learning rate that maximizes Eq. 9, the optimal learning rate, η^* is

$$\eta^* = \frac{1}{\sigma_{\text{min}}^2 + \sigma_{\text{max}}^2}, \tag{10}$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and minimum singular value of the matrix X. The corresponding optimal convergence rate is

$$\text{optimal rate} = \frac{(\sigma_{\max}/\sigma_{\min})^2 - 1}{(\sigma_{\max}/\sigma_{\min})^2 + 1} \tag{11}$$

Therefore, how fast ordinary gradient descent converges is determined by the ratio between the maximum singular value and the minimum singular value as above.

Now, let's consider using momentum to smooth the gradients before taking a step in Eq.6.

$$\begin{aligned} w_{t+1} &= w_t - \eta z_{t+1} \\ z_{t+1} &= (1-\beta) z_t + \beta(2X^T X w_t - 2X^T y) \end{aligned} \tag{12}$$

We can use the SVD of the matrix $X = U\Sigma V^T$, where $\Sigma = \text{diag}(\sigma_{\max}, \sigma_2, \ldots, \sigma_{\min})$ with the same (potentially rectangular) shape as X. This allows us to reparameterize the parameters w_t and averaged gradients z_t as below:

$$\begin{aligned} x_t &= V^T(w_t - w^*) \\ a_t &= V^T z_t. \end{aligned} \tag{13}$$

Please rewrite Eq. 12 with the reparameterized variables, $x_t[i]$ and $a_t[i]$. ($x_t[i]$ and $a_t[i]$ are i-th components of x_t and a_t respectively.)

(b) Notice that the above 2×2 vector/matrix recurrence has no external input. We can derive the 2×2 system matrix R_i from above such that

$$\begin{bmatrix} a_{t+1}[i] \\ x_{t+1}[i] \end{bmatrix} = R_i \begin{bmatrix} a_t[i] \\ x_t[i] \end{bmatrix} \tag{14}$$

Derive R_i.

(c) **Use the computer to symbolically find the eigenvalues of the matrix R_i.**
When are they purely real? When are they repeated and purely real? When are they complex?

(d) **For the case when they are repeated, what is the condition on η, β, σ_i that keeps them stable (strictly inside the unit circle)? What is the highest learning rate η as a function of β and σ_i that results in repeated eigenvalues?**

(e) **For the case when the eigenvalues are real, what is the condition on η, β, σ_i that keeps them stable (strictly inside the unit circle)? What is the range of the learning rate? Express with β, σ_i**

(f) **For the case when the eigenvalues are complex, what is the condition on η, β, σ_i that keeps them stable (strictly inside the unit circle)? What is the highest learning rate η as a function of β and σ_i that results in complex eigenvalues?**

(g) (This question might take more time than others) Now, apply what you have learned to the following problem. Assume that $\beta = 0.1$ and we have a problem with two singular values $\sigma^2_{\max} = 5$ and $\sigma^2_{\min} = 0.05$. **What learning rate η should we choose to get the fastest convergence for gradient descent with momentum? Compare how many iterations it will take to get within 99.9% of the optimal solution (starting at 0) using this learning rate and momentum with what it would take using ordinary gradient descent.**

(h) The 2 questions below are based on the Jupyter Notebook given in this url. Please open the corresponding notebook and follow the instructions to answer the following questions. You don't need to submit the `ipynb` file.

How does σ_i (the eigenvalues) influence the gradients and paramters updates?

(i) Question: Comparing gradient descent and gradient descent with momentums, **which one converges faster for this task? Why?**

3. Regularization and Instance Noise

Say we have m labeled data points $(\mathbf{x}_i, y_i)_{i=1}^m$, where each $\mathbf{x}_i \in \mathbb{R}^n$ and each $y_i \in \mathbb{R}$. We perform data augmentation by adding some noise to each vector every time we use it in SGD. This means for all points i, we have a true input $\mathbf{x}_i$ and add noise $\mathbf{N}_i$ to get the effective random input seen by SGD:

$$\check{\mathbf{X}}_i = \mathbf{x}_i + \mathbf{N}_i$$

The i.i.d. random noise vectors $\mathbf{N}_i$ are distributed as $\mathbf{N}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_n)$.

We can conceptually arrange these noise-augmented data points into a random matrix $\check{X} \in \mathbb{R}^{m \times n}$, where row $\check{\mathbf{X}}_i^\top$ represents one augmented datapoint. Similarly we arrange the labels y_i into a vector $\mathbf{y}$.

$$\check{X} = \begin{bmatrix} \check{\mathbf{X}}_1^\top \\ \check{\mathbf{X}}_2^\top \\ \dots \\ \check{\mathbf{X}}_m^\top \end{bmatrix}, \text{ where } \check{\mathbf{X}}_i \in \mathbb{R}^n, \text{ and } \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix} \in \mathbb{R}^m$$

One way of thinking about what SGD might do is to consider learning weights that minimize the *expected* least squares objective for the **noisy** data matrix:

$$\operatorname*{argmin}_{\mathbf{w}} \mathbb{E}[\|\check{X}\mathbf{w} - \mathbf{y}\|^2] \tag{15}$$

(a) **Show that this problem (15) is equivalent to a regularized least squares problem:**

$$\operatorname*{argmin}_{\mathbf{w}} \frac{1}{m}\|X\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2 \tag{16}$$

You will need to determine the value of λ.

Hint: write the squared norm of a vector as an inner product, expand, and apply linearity of expectation.

Now consider a simplified example where we only have a single scalar datapoint $x \in \mathbb{R}$ and its corresponding label $y \in \mathbb{R}$. We are going to analyze this in the context of gradient descent. For the t-th step of gradient descent, we use a noisy datapoint $\check{X}_t = x + N_t$ which is generated by adding different random noise values $N_t \sim \mathcal{N}(0, \sigma^2)$ to our underlying data point x. The noise values for each iteration of gradient descent are i.i.d. We want to learn a weight w such that the squared-loss function $\mathcal{L}(w) = \frac{1}{2}(\check{X}w - y)^2$ is minimized. We initialize our weight to be $w_0 = 0$.

(b) Let w_t be the weight learned after the t-th iteration of gradient descent with data augmentation. **Write the gradient descent recurrence relation between $\mathbb{E}[w_{t+1}]$ and $\mathbb{E}[w_t]$ in terms of x, σ^2, y, and learning rate η.**

(c) **For what values of learning rate η do we expect the expectation of the learned weight to converge using gradient descent?**

(d) Assuming that we are in the range of η for which gradient-descent converges, **what would we expect $\mathbb{E}[w_t]$ to converge to as $t \to \infty$? How does this differ from the optimal value of w if there were no noise being used to augment the data?**

(HINT: You can also use this to help check your work for part (a).)

4. An Alternate MAP Interpretation of Ridge Regression

Consider the Ridge Regression estimator,

$$\underset{\mathbf{w}}{\operatorname{argmin}} \|X\mathbf{w} - y\|_2^2 + \lambda\|\mathbf{w}\|^2$$

We know this is solved by

$$\hat{\mathbf{w}} = (X^T X + \lambda I)^{-1} X^T \mathbf{y} \tag{17}$$

An alternate form of the Ridge Regression solution (often called the Kernel Ridge form) is given by

$$\hat{\mathbf{w}} = X^T (X X^T + \lambda I)^{-1} \mathbf{y}. \tag{18}$$

We know that Ridge Regression can be viewed as finding the MAP estimate when we apply a prior on the (now viewed as random parameters) $\mathbf{W}$. In particular, we can think of the prior for $\mathbf{W}$ as being $\mathcal{N}(\mathbf{0}, I)$ and view the random Y as being generated using $Y = \mathbf{x}^T\mathbf{W} + \sqrt{\lambda}N$ where the noise N is distributed iid (across training samples) as $\mathcal{N}(0,1)$. At the vector level, we have $\mathbf{Y} = X\mathbf{W} + \sqrt{\lambda}\mathbf{N}$, and then we know that when we try to maximize the log likelihood we end up minimizing

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{\lambda}\|X\mathbf{w} - y\|_2^2 + \|\mathbf{w}\|^2 = \underset{\mathbf{w}}{\operatorname{argmin}} \|X\mathbf{w} - y\|_2^2 + \lambda\|\mathbf{w}\|^2.$$

The underlying probability space is that defined by the d iid standard normals that define the $\mathbf{W}$ and the n iid standard normals that give the n different N_i on the training points. Note that the X matrix whose rows consist of the n different inputs for the n different training points are not random.

Based on what we know about joint normality, it is clear that the random Gaussian vectors $\mathbf{W}$ and $\mathbf{Y}$ are jointly normal. **Use the following facts to show that the two forms of solution are identical.**

- (17) is the MAP estimate for $\mathbf{W}$ given an observation $\mathbf{Y} = \mathbf{y}$ (We showed this in HW1 last week, and in discussion section)
- For jointly normal random variables, when you condition one set of variables on the values for the others, the resulting conditional distribution is still normal.
- A normal random variable has its density maximized at its mean.
- For jointly normal random vectors that are zero mean, the formula for conditional expectation is

$$E[\mathbf{W}|\mathbf{Y} = \mathbf{y}] = \Sigma_{WY}\Sigma_{YY}^{-1}\mathbf{y} \tag{19}$$

  where the Σ_{YY} is the covariance $E[\mathbf{Y}\mathbf{Y}^T]$ of $\mathbf{Y}$ and $\Sigma_{WY} = E[\mathbf{W}\mathbf{Y}^T]$ is the appropriate cross-covariance of $\mathbf{W}$ and $\mathbf{Y}$.

5. Coding Question: Initialization and Optimizers

In this question, you'll implement He Initialization and Different Optimizers. You will have the choice between two options:

Use Google Colab (Recommended). Open this url and follow the instructions in the notebook.

Use a local Conda environment. Clone `https://github.com/Berkeley-CS182/cs182hw2` and refer to `README.md` for further instructions.

(a) What you observe in the mean of gradient norm plot above in the above plots? **Try to give an explanation.**

6. (Optional) Visualizing Features from Local Linearization of Neural Nets (Part II)

This question is a sequel to the question about local linearization of the network in the neighborhood of the parameters. In this part, We will now compare shallow and deep networks, examine the effects of different initialization strategies, and investigate the consequences of training a neural network on a mismatched training data distribution.

We provide you with some starter code on Google Colab. For this question, **please do *not* submit your code to Gradescope**. Instead, just include the answers to the questions in your submission of the written assignment.

(a) **What are the gradient norms of each layer when the init weight scale is small** (`-0.03` to `0.03`)?

(b) **Describe the performance of the model** initialized with the small weight scale.

(c) **Record and explain your observation of the singular values and principal features of the local linearization gradient matrix of the model initialized with the small weight scale before and after training.**

(d) **What are the gradient norms of each layer when the init weight scale is large** (`-3.0` to `3.0`)?

(e) **What happened when we try to train the model initialized with the large weight scale?**

(f) **What are the gradient norms of each layer when the neural network is initialize with your implemented method?**

(g) Based on your observation of the singular values and principal features of the local linearization gradient matrix, **compare the properly-initialized neural network with the neural network initialized with a very large weight scale.** (before training)

(h) **Describe the performance of the model** initialized with your implemented method.

(i) Based on your observation of the singular values and principal features of the local linearization gradient matrix, **compare the *trained* shallow neural network (1 hidden layer) with the *trained* deep neural network (4 hidden layers).**

(j) **Describe the performance of the model** when the x values of training data range from `-1.0` to `0.4`

(k) Based on your observation of the singular values and principal features of the local linearization gradient matrix, **compare the model trained with mismatched training data with models (both shallow and deep model) trained with matched training data.**

7. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

(c) **Roughly how many total hours did you work on this homework? Write it down here where you'll need to remember it for the self-grade form.**

Contributors:

- Anant Sahai.
- Sheng Shen.
- Suhong Moon.
- Gabriel Goh.
- Peter Wang.
- Saagar Sanghavi.
- Hao Liu.
- Andrew Ng.
- Linyuan Gong.

