EECS 182 Deep Neural Networks
Spring 2023 Anant Sahai

Homework 2

This homework is due on Friday, February 10, 2023, at 10:59PM.

1. Why Learning Rates Cannot be Too Big

Note: This question will be covered in the discussion on Tuesday/Wednesday, feel free to directly copy and paste the solutions from the discussion worksheet after the discussion (with citation).

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?)

Solution: We can rewrite the update rule as

$$\begin{aligned} w_{t+1} - \frac{y}{\sigma} &= (1 - 2\eta\sigma^2)(w_t - \frac{y}{\sigma}) \\ w_{t+1} &= \frac{y}{\sigma} + (1 - 2\eta\sigma^2)^{t+1}(w_0 - \frac{y}{\sigma}) \end{aligned} \tag{4}$$

To make the recurrence (3) stable with $\eta > 0$, we need $|1 - 2\eta\sigma^2| < 1$. This gives $\eta < \frac{1}{\sigma^2}$.

(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.)

Solution:

$$|w_T - w^*| < \epsilon |w^*| \tag{5}$$

Use the derived update rule in (4). We have

$$\begin{aligned} |w_T - \frac{y}{\sigma}| &< \epsilon |\frac{y}{\sigma}| \\ |(1 - 2\eta\sigma^2)^T| &< \epsilon \\ T &> \frac{log(\epsilon)}{log(|1 - 2\eta\sigma^2|)}, \end{aligned} \tag{6}$$

(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:

$$\begin{bmatrix} \sigma_\ell & 0 \\ 0 & \sigma_s \end{bmatrix} \begin{bmatrix} w[1] \\ w[2] \end{bmatrix} = \begin{bmatrix} y[1] \\ y[2] \end{bmatrix}. \tag{7}$$

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?

Solution: Similarly, we can rewrite the loss function and update rule w.r.t the vector form.

$$\begin{aligned} L(\mathbf{w}) = ||\mathbf{y} - \Sigma\mathbf{w}||^2, \Sigma &= \begin{bmatrix} \sigma_\ell & 0 \\ 0 & \sigma_s \end{bmatrix} \\ \nabla_{\mathbf{w}} L(\mathbf{w}) &= 2(\Sigma^2\mathbf{w} - \Sigma\mathbf{y}) \\ \mathbf{w}_{t+1} &= (I - 2\eta\Sigma^2)\mathbf{w}_t + 2\eta\Sigma\mathbf{y} \end{aligned} \tag{8}$$

To ensure the convergence, we need

$$\begin{cases} |1 - 2\eta\sigma_l^2| < 1 \\ |1 - 2\eta\sigma_s^2| < 1 \end{cases} \tag{9}$$

$$\eta < \min(\frac{1}{\sigma_l^2}, \frac{1}{\sigma_s^2}) = \frac{1}{\sigma_l^2} \tag{10}$$

(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?**

Solution: We can rewrite the update rule w.r.t each dimension, this gives

$$\begin{aligned} w[1]_t &= \frac{y[1]}{\sigma_l} + (1 - 2\eta\sigma_l^2)^t(-\frac{y[1]}{\sigma_l}) \\ w[2]_t &= \frac{y[2]}{\sigma_s} + (1 - 2\eta\sigma_s^2)^t(-\frac{y[2]}{\sigma_s}) \end{aligned}$$

This faster convergence dimension is $\min(|1-2\eta\sigma_l^2|, |1-2\eta\sigma_s^2|)$

(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?**

Solution: Recall that the minimum of the pointwise maximum of two functions occurs at a point where both functions are equal. Thus, the fastest convergence is achieved when $1-2\eta\sigma_l^2 > 0$ and $1-2\eta\sigma_l^2 = 1-2\eta\sigma_s^2$, thus $\eta = \frac{1}{\sigma_l^2+\sigma_s^2}$

(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?**

Solution: The bounds on the learning rate should still be the same. For maximal convergence, the largest and smallest values of sigma are the two that cause gradient descent to take the longest if we poorly choose a learning rate.

(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?**

Solution: We can think of SVD as a change of bases into and then back from a coordinate system where X instead is just diagonal, this will directly connect to the same problem with lots of parallel scalar problems.

2. Accelerating Gradient Descent with Momentum

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

$$\mathcal{L}(w) = ||y - Xw||_2^2 \tag{11}$$

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:

$$\begin{aligned} w_{t+1} &= w_t - \eta z_{t+1} \\ z_{t+1} &= (1-\beta)z_t + \beta g_t, \end{aligned} \tag{12}$$

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 11 is

$$w_{t+1} = \left(I - 2\eta(X^TX)\right) w_t + 2\eta X^T y \tag{13}$$

and the minimizer is

$$w^* = (X^TX)^{-1}X^Ty \tag{14}$$

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

$$\text{rate} = \max_i |1 - 2\eta\sigma_i^2| \tag{15}$$

You saw on the last homework that if we choose the learning rate that maximizes Eq. 15, the optimal learning rate, η^* is

$$\eta^* = \frac{1}{\sigma_{\text{min}}^2 + \sigma_{\text{max}}^2}, \tag{16}$$

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

$$\text{optimal rate} = \frac{(\sigma_{\text{max}}/\sigma_{\text{min}})^2 - 1}{(\sigma_{\text{max}}/\sigma_{\text{min}})^2 + 1} \tag{17}$$

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.12.

$$\begin{aligned} w_{t+1} &= w_t - \eta z_{t+1} \\ z_{t+1} &= (1-\beta)z_t + \beta(2X^TXw_t - 2X^Ty) \end{aligned} \tag{18}$$

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

$$\begin{aligned} x_t &= V^T(w_t - w^*) \\ a_t &= V^T z_t. \end{aligned} \tag{19}$$

Please rewrite Eq. 18 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.)

Solution: Let's multiply V^T both sides in Eq.(18). Then,

$$\begin{aligned} V^Tw_{t+1}[i] &= V^Tw_t[i] - \eta V^Tz_{t+1}[i] \\ x_{t+1}[i] + V^Tw^*[i] &= x_t[i] + V^Tw^*[i] - \eta a_{t+1}[i] \\ x_{t+1}[i] &= x_t[i] - \eta a_{t+1}[i] \end{aligned}$$

$$\begin{aligned} V^Tz_{t+1} &= (1-\beta)V^Tz_t + \beta V^T(2X^TXw_t - 2X^Ty) \\ a_{t+1} &= (1-\beta)a_t + \beta V^T(2V\Sigma^2V^T(Vx_t + w^*) - 2V^TX^Ty) \\ a_{t+1} &= (1-\beta)a_t + \beta(2\Sigma^2V^TVx_t + \Sigma^2V^Tw^* - 2V^TX^Ty) \end{aligned} \tag{20}$$

Note that,

$$w^* = (X^TX)^{-1}X^Ty \tag{21}$$

$$= V\Sigma^{-2}V^TX^Ty \tag{22}$$

Let's plug the above result into Eq.(20).

$$\begin{aligned} a_{t+1} &= (1-\beta)a_t + \beta(2\Sigma^2 V^T V x_t + 2\Sigma^2 V^T w^* - 2V^T X^T y) \\ &= (1-\beta)a_t + \beta(2\Sigma^2 V^T V x_t + 2\Sigma^2 V^T V \Sigma^{-2} V^T X^T y - 2V^T X^T y) \\ &= (1-\beta)a_t + \beta 2\Sigma^2 V^T V x_t \\ &= (1-\beta)a_t + \beta 2\Sigma^2 x_t \end{aligned}$$

If we express the above result element-wise,

$$\begin{aligned} a_{t+1}[i] &= (1-\beta)a_t[i] + 2\beta\sigma_i^2 x_t[i] \\ x_{t+1}[i] &= x_t[i] - \eta a_{t+1}[i] \end{aligned}$$

(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

$$\begin{bmatrix} a_{t+1}[i] \\ x_{t+1}[i] \end{bmatrix} = R_i \begin{bmatrix} a_t[i] \\ x_t[i] \end{bmatrix} \tag{23}$$

Derive R_i.

Solution:

Since x_{t+1} is expressed with x_t and a_{t+1}, let's reformulate with x_t and a_t

$$\begin{aligned} x_{t+1}[i] &= x_t[i] - \eta a_{t+1}[i] \\ &= x_t[i] - \eta(1-\beta)a_t[i] - 2\eta\beta\sigma_i^2 x_t[i] \\ &= (1 - 2\eta\beta\sigma_i^2)x_t[i] - \eta(1-\beta)a_t[i] \end{aligned}$$

Therefore, the matrix R_i can be represented as below:

$$R_i = \begin{bmatrix} (1-\beta) & 2\beta\sigma_i^2 \\ -\eta(1-\beta) & 1 - 2\eta\beta\sigma_i^2 \end{bmatrix}$$

(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?

Solution: Let's derive the characteristic equation of R_i to derive eigenvalues:

$$\begin{aligned} f(\lambda) &= |R_i - \lambda I| \qquad (24) \\ &= \left| \begin{bmatrix} (1-\beta) - \lambda & 2\beta\sigma_i^2 \\ -\eta(1-\beta) & 1 - 2\eta\beta\sigma_i^2 - \lambda \end{bmatrix} \right| \\ &= \lambda^2 - \left(2 - \beta - 2\eta\beta\sigma_i^2\right)\lambda + 1 - \beta \qquad (25) \end{aligned}$$

Setting this to zero and solving, we get that the eigenvalues are:

$$\lambda_{1,i} = 1 - \beta\eta\sigma_i^2 - \frac{\beta}{2} - \frac{\sqrt{\beta\left(4\beta\eta^2\sigma_i^4 + 4\beta\eta\sigma_i^2 + \beta - 8\eta\sigma_i^2\right)}}{2}$$
$$\lambda_{2,i} = 1 - \beta\eta\sigma_i^2 - \frac{\beta}{2} + \frac{\sqrt{\beta\left(4\beta\eta^2\sigma_i^4 + 4\beta\eta\sigma_i^2 + \beta - 8\eta\sigma_i^2\right)}}{2}$$

The discriminant of Eq.24 is:

$$D = \left(2 - \beta - 2\eta\beta\sigma_i^2\right)^2 - 4(1-\beta) \tag{26}$$
$$= \beta\left(4\beta\eta^2\sigma_i^4 + 4\beta\eta\sigma_i^2 + \beta - 8\eta\sigma_i^2\right). \tag{27}$$

To have distinctive and real eigenvalues, $D > 0$
To have repeated real eigenvalues, $D = 0$
To have complex eigenvalues, $D < 0$

(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?**

Solution: To find the maximum value, let's put the $D = 0$. Then, we can find two solutions:

$$\frac{2 - \beta - 2\sqrt{1-\beta}}{2\beta\sigma_i^2} \text{ or } \frac{2 - \beta + 2\sqrt{1-\beta}}{2\beta\sigma_i^2}$$

Therefore, the highest learning rate is $\frac{2-\beta+2\sqrt{1-\beta}}{2\beta\sigma_i^2}$
Plugging in learning rate to the eigenvalues, we have:

$$\lambda_{1,i} = \lambda_{2,i} = 1 - \beta\eta\sigma_i^2 - \frac{\beta}{2} = \pm\sqrt{1-\beta} \tag{28}$$

If eigenvalues are repeated or complex, this system is always stable as $0 < \beta < 1$. This result is very surprising in that the convergence does not depend neither on the learning rate nor on the singular values of the covariate matrix.

(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**

Solution: At first, to have the different real roots, the discriminant Eq.(26), D should be strictly positive:

$$D = \beta\left(4\beta\eta^2\sigma_i^4 + 4\beta\eta\sigma_i^2 + \beta - 8\eta\sigma_i^2\right) > 0 \tag{29}$$

If we solve Eq.(29) with respect to η,

$$\eta < \frac{2 - \beta - 2\sqrt{1-\beta}}{2\beta\sigma_i^2} \text{ or } \eta > \frac{2 - \beta + 2\sqrt{1-\beta}}{2\beta\sigma_i^2} \tag{30}$$

Also, to guarantee that those two solutions are in the open interval (0, 1) [1], the characteristic equation

[1] For stability, the norm should be less than 1

in Eq.(24) should satisfy $f(1) > 0$ and $f(-1) > 0$. We can check this by drawing the graph of $f(\lambda)$. Let's first compute $f(1)$

$$\begin{aligned} f(1) &= 1 - \left(2 - \beta - 2\eta\beta\sigma_i^2\right) + 1 - \beta \\ &= 2\eta\beta\sigma_i^2 > 0 \end{aligned}$$

Therefore, $f(1)$ is always positive.
Now, let's look at $f(-1)$.

$$\begin{aligned} f(1) &= 1 + \left(2 - \beta - 2\eta\beta\sigma_i^2\right) + 1 - \beta \\ &= 4 - 2\beta - 2\eta\beta\sigma_i^2 > 0 \end{aligned}$$

Thus,

$$0 < \eta < \frac{4 - 2\beta}{2\beta\sigma_i^2} \tag{31}$$

Let's compare Eq.(30) and Eq.(31). Since

$$4 - 2\beta \geq 2 - \beta + 2\sqrt{1-\beta}, \text{for all} \beta \in [0, 1]$$

the condition that Eq.(24) has two different real roots is

$$0 < \eta < \frac{2 - \beta - 2\sqrt{1-\beta}}{2\beta\sigma_i^2} \text{or} \frac{2 - \beta - 2\sqrt{1+\beta}}{2\beta\sigma_i^2} < \eta < \frac{4 - 2\beta}{2\beta\sigma_i^2}$$

(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?**

Solution: If eigenvalues are repeated or complex, this system is always stable as $0 < \beta < 1$. This result is very surprising in that the convergence does not depend neither on the learning rate nor on the singular values of the covariate matrix
To find the maximum value, let's put $D < 0$. Then, we can find two solutions:

$$\frac{2 - \beta - 2\sqrt{1-\beta}}{2\beta\sigma_i^2} < \eta < \frac{2 - \beta + 2\sqrt{1-\beta}}{2\beta\sigma_i^2}$$

Therefore, the highest learning rate is $\frac{2-\beta+2\sqrt{1-\beta}}{2\beta\sigma_i^2}$. From the questions (d) (f), the takeaway is that we have to choose η such that every eigenvalues of R_i are either the same real roots or complex roots. Because η can be much higher for those cases.

(g) Now, apply what you have learned to the following problem. Assume that $\beta = 0.1$ and we have a problem with two singular values $\sigma^2_{\text{max}} = 5$ and $\sigma^2_{\text{min}} = 0.05$. **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.**

Solution: For OGD, referring Eq.(17) the optimal convergence rate is $R_1 = (5-0.05)/(5+0.05) =$

0.98. Therefore, the minimum number of iterations (T) to get within 99.9% of the optimal solution is:

$$||w_{T_1} - w^*||_2 = R_1^{T_1}||w_0 - w^*||_2 = (\frac{5-0.05}{5+0.05})^{T_1}||w^*|| \leq \epsilon||w^*||$$
$$T_1 = \lfloor \frac{\log 1000}{\log 505/495} \rfloor = 346$$

For GD+momentum, let's derive the optimal learning rate first. The optimal learning rate η^* and corresponding convergence rate R_2 are:

$$\begin{aligned}\eta^* &= \underset{\eta}{\operatorname{argmin}} \max\left\{ \left\| \begin{bmatrix} 1-\beta & 2\beta\sigma_{\max}^2 \\ -\eta(1-\beta) & 1-2\eta\beta\sigma_{\max}^2 \end{bmatrix} \right\|, \left\| \begin{bmatrix} 1-\beta & 2\beta\sigma_{\min}^2 \\ -\eta(1-\beta) & 1-2\eta\beta\sigma_{\min}^2 \end{bmatrix} \right\| \right\} \\ &= \underset{\eta}{\operatorname{argmin}} \max\{|\lambda_{1,\max}|, |\lambda_{2,\max}|, |\lambda_{1,\min}|, |\lambda_{2,\min}|\} \\ R_2 &= \min_{\eta} \max\{|\lambda_{1,\max}|, |\lambda_{2,\max}|, |\lambda_{1,\min}|, |\lambda_{2,\min}|\}\end{aligned}$$

, where $\|\cdot\|$ is the maximum of absolute value of eigenvalues and $\lambda_{i,\max}, \lambda_{i,\min}$. are eigenvalues of matrices.

As you can see the above objective is seemingly very difficult to solve. But we can get some hints from the previous parts. First, Let's look at Eq.(24). The constant term is $1-\beta$, which is the product of two solutions. If one of solutions is λ, the other one is $\frac{1-\beta}{\lambda}$. In other words, if $\lambda < \sqrt{1-\beta}$, the other solution is larger than $\sqrt{1-\beta}$. Then R_2 cannot be smaller than $\sqrt{1-\beta}$ Therefore R_2 has the lower bound, $R_2 \geq \sqrt{1-\beta}$.

Then, let's hypothesize that we can achieve that lower bound. Remind that if two solutions are repeated real or complex, their absolute values are always $\sqrt{1-\beta}$. So to prove the hypothesis, we need to show that eigenvalues of each matrix are either repeated real or complex. In that case, Eq.(26) should be non-negative for both matrices.

$$\frac{2-\beta-2\sqrt{1-\beta}}{2\beta\sigma_{\max}^2} \leq \eta \leq \frac{2-\beta+2\sqrt{1-\beta}}{2\beta\sigma_{\max}^2}$$
$$\frac{2-\beta-2\sqrt{1-\beta}}{2\beta\sigma_{\min}^2} \leq \eta \leq \frac{2-\beta+2\sqrt{1-\beta}}{2\beta\sigma_{\min}^2}$$

Inserting β, σ_i^2, the above inequalities, we can get the following result:

$$0.002633 \leq \eta \leq 3.797$$
$$0.2633 \leq \eta \leq 379.7$$

The common interval that η satisfies both inequalities is

$$0.2633 \leq \eta \leq 3.797$$

If η is in that interval, the convergence rate is $\sqrt{1-\beta} = 0.949$

$$||w_{T_2} - w^*||_2 \leq R_2^{T_2}||w_0 - w^*||_2 = (0.949)^{T_2}||w^*|| \leq \epsilon||w^*||$$
$$T_2 \geq \lfloor \frac{\log 1/\epsilon}{\log 1/R_2} \rfloor = 132$$

We need at least 132 iterations to guarantee 0.1% error.

Note: if you find the convergence rate correctly $\sqrt{1-\beta}$ and derive any η in the interval [0.263, 3.80], then you can get the full credit for this part.

(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?

Solution: Dimension 0 has larger σ/eigenvalue, so the gradient is larger. With a relatively large stepsize (what we choose here), the gradients and parameters are oscillating a bit at the begining before converging.

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

Solution: Gradient descent with momentum is faster in convergence compared to traditional gradient descent. This is because gradient descent with momentum adjusts the magnitude of the parameter update in each dimension, allowing the optimizer to make bigger and more confident updates in the dimensions where the gradients are pointing in the same direction, and smaller updates in the dimensions where the gradients are oscillating. This leads to a more efficient optimization process and faster convergence to the optimal solution.

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 $\mathbf{y}$.

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

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:

$$\operatorname*{argmin}_{\mathbf{w}} \mathbb{E}[\|\check{X}\mathbf{w} - \mathbf{y}\|^2] \tag{32}$$

(a) **Show that this problem (32) is equivalent to a regularized least squares problem:**

$$\operatorname*{argmin}_{\mathbf{w}} \frac{1}{m}\|X\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2 \tag{33}$$

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.

Solution: Following the hint, we write our objective as

$$\begin{aligned}
\mathbb{E}[\|\check{X}\mathbf{w}-\mathbf{y}\|^2] &= \mathbb{E}\left[\sum_{i=1}^{m}(\check{\mathbf{x}}_i^\top\mathbf{w}-y_i)^2\right] \\
&= \sum_{i=1}^{m}\mathbb{E}[((\mathbf{x}_i+\mathbf{N}_i)^\top\mathbf{w}-y_i)^2] \\
&= \sum_{i=1}^{m}\mathbb{E}[(\mathbf{x}_i^\top\mathbf{w}+\mathbf{N}_i^\top\mathbf{w}-y_i)^2] \\
&= \sum_{i=1}^{m}\mathbb{E}[(\mathbf{x}_i^\top\mathbf{w}-y_i)^2-2(\mathbf{N}_i^\top\mathbf{w})(\mathbf{x}_i^\top\mathbf{w}-y_i)+(\mathbf{N}_i^\top\mathbf{w})^2] \\
&= \sum_{i=1}^{m}\mathbb{E}[(\mathbf{x}_i^\top\mathbf{w}-y_i)^2]-2\mathbb{E}[(\mathbf{N}_i^\top\mathbf{w})(\mathbf{x}_i^\top\mathbf{w}-y_i)]+\mathbb{E}[\mathbf{w}^\top\mathbf{N}_i\mathbf{N}_i^\top\mathbf{w}] \\
&= \sum_{i=1}^{m}(\mathbf{x}_i^\top\mathbf{w}-y_i)^2-2\mathbb{E}[(\mathbf{N}_i^\top\mathbf{w})](\mathbf{x}_i^\top\mathbf{w}-y_i)+\mathbf{w}^\top\mathbb{E}[\mathbf{N}_i\mathbf{N}_i^\top]\mathbf{w} \\
&= \sum_{i=1}^{m}[(\mathbf{x}_i^\top\mathbf{w}-y_i)^2-0+\mathbf{w}^\top\sigma^2 I_n\mathbf{w}] \\
&= \|X\mathbf{w}-\mathbf{y}\|^2+m\sigma^2\|\mathbf{w}\|^2
\end{aligned}$$

Dividing through by m to match the desired form of (33) gives us $\lambda=\sigma^2$.

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.**

Solution: We can first compute the gradient in this case:

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial w} &= (w\check{X}-y)\check{X} \\
&= (w(x+N_t)-y)(x+N_t) \\
&= w(x^2+2xN_t+N_t^2)-y(x-N_t).
\end{aligned}$$

And now, we can use it for gradient descent with stepsize η:

$$w_{t+1}=w_t-\eta\nabla_w L(w_t)$$

$$= w_t - \eta(w_t(x+N_t)^2 - y(x+N_t))$$
$$= w_t(1 - \eta(x^2 + 2xN_t + N_t^2)) - \eta y(x+N_t)$$

Taking expectations, and using the fact that N_t is independent of w_t since the noise values are i.i.d., we get:

$$\mathbb{E}[w_{t+1}] = \mathbb{E}[w_t](1 - \eta(x^2 + \sigma^2)) + \eta yx$$

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

Solution: For gradient descent to converge, we need the coefficient on the recurrent term to be between -1 and 1.

$$-1 < 1 - \eta(x^2 + \sigma^2) \text{ and } 1 - \eta(x^2 + \sigma^2) < 1$$

This implies

$$0 < \eta < \frac{2}{x^2 + \sigma^2}$$

(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).)

Solution: One way of doing this is to take an expectation of the gradient and set it to 0. Using the fact that $\mathbb{E}[N_t] = 0$ and $\mathbb{E}[N_t^2] = \sigma^2$,

$$w^*(x^2 + \sigma^2) - yx = 0$$

$$w^* = \frac{yx}{x^2 + \sigma^2}$$
$$= (\frac{y}{x})\frac{1}{1 + \frac{\sigma^2}{x^2}}$$

The optimal value of w if there was no data-augmenting noise would simply be $w = \frac{y}{x}$. This means the optimal expected value with noise augmentation is scaled down by a factor of $\frac{1}{1+\frac{\sigma^2}{x^2}}$. This looks just like the "shrinkage" term we saw on the eigenvalues with Ridge regression. When σ is small relative to x, the learned weight barely changes. When σ is large relative to x, the learned weight shrinks significantly towards zero.

4. An Alternate MAP Interpretation of Ridge Regression

Consider the Ridge Regression estimator,

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

We know this is solved by

$$\hat{\mathbf{w}} = (X^TX + \lambda I)^{-1}X^T\mathbf{y} \tag{34}$$

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

$$\hat{\mathbf{w}} = X^T(XX^T + \lambda I)^{-1}\mathbf{y}. \tag{35}$$

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.**

- (34) 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

$$E[\mathbf{W}|\mathbf{Y} = \mathbf{y}] = \Sigma_{WY}\Sigma_{YY}^{-1}\mathbf{y} \tag{36}$$

  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 $\mathbf{Y}$.

Solution:

We are given that (34) is MAP($\mathbf{w} \mid \mathbf{Y} = \mathbf{y}$). Let's try to find MAP($\mathbf{w} \mid \mathbf{Y} = \mathbf{y}$) in a different way.

We can condition the random variables $\mathbf{W}$ on the values for $\mathbf{Y} = \mathbf{y}$, and we know the resulting conditional distribution $\mathbf{W}|\mathbf{Y} = \mathbf{y}$ is still normal.

Since normal random variables has density maximized at the mean, the MAP estimate is equivalent to the conditional expectation $E[\mathbf{W}|\mathbf{Y} = \mathbf{y}]$. We are given the formula to calculate the conditional expectation.

Before we plug into the formula, let's calculate $\Sigma_{WY} = E[\mathbf{W}\mathbf{Y}^\mathbf{T}]$. Using $\mathbf{Y} = X\mathbf{W} + \mathbf{N}$, we have:

$$\Sigma_{WY} = E[\mathbf{W}(X\mathbf{W} + \sqrt{\lambda}\mathbf{N})^T]$$

Take transposes and distribute, apply linearity of expectation

$$\Sigma_{WY} = E[\mathbf{W}\mathbf{W}^T X^T] + \sqrt{\lambda}E[\mathbf{W}\mathbf{N}^T]$$

Since $\mathbf{W}$ and $\mathbf{N}$ are uncorrelated, and X does not involve any randomness, we can write

$$\Sigma_{WY} = E[\mathbf{W}\mathbf{W}^T]X^T + \sqrt{\lambda}E[\mathbf{W}]E[\mathbf{N}^T]$$

Use the fact that $E[\mathbf{W}\mathbf{W}^T]$ is the covariance matrix of $\mathbf{W}$, which we are given is the identity. Also use the fact that $\mathbf{W}$ and $\mathbf{N}$ are zero-meaned.

$$\Sigma_{WY} = IX^T + 0 = X^T$$

Now let's find $\Sigma_{YY} = E[\mathbf{Y}\mathbf{Y}^T]$.

$$\Sigma_{YY} = E[(X\mathbf{W} + \mathbf{N})(X\mathbf{W} + \mathbf{N})^T]$$

Take transposes and distribute

$$\Sigma_{YY} = E[X\mathbf{W}\mathbf{W}^T X^T] + \sqrt{\lambda}^2 E[\mathbf{N}\mathbf{N}^T] + \sqrt{\lambda}E[X\mathbf{W}\mathbf{N}^T] + \sqrt{\lambda}E[\mathbf{N}\mathbf{W}^T X^T]$$

The cross-terms are 0 since the random vectors are zero-meaned. With some manipulation, we have

$$\Sigma_{YY} = XE[\mathbf{W}\mathbf{W}^T]X^T + \lambda I$$

$$\Sigma_{YY} = XX^T + \lambda I$$

Finally, plugging in Σ_{WY} and Σ_{YY} to the formula for $E[\mathbf{W}|\mathbf{Y} = \mathbf{y}]$, we get $MAP(\mathbf{w}|\mathbf{Y} = \mathbf{y})$ as

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

as desired.

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.

For this question, please submit a .zip file your completed work to the Gradescope assignment titled "HW 2 (Code)".

For this question, please submit a .zip file your completed work to the Gradescope assignment titled "Homework 2 (Code)". Please answer the following question in your submission of the written assignment:

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

Solution: Random and He initialization should train as expected. When we use zero initialization, our gradients become zero as well, which makes it very difficult to train our network to obtain nonzero weights.

6. 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`)?
Solution: See solution notebook. The gradient norms are very small.

(b) **Describe the performance of the model** initialized with the small weight scale. **Solution:** See solution notebook. The model's training loss does not change during training because the gradient it tiny. It predicts a constant value for any given x because all weights are very small and almost never updated.

(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.**

Solution: See solution notebook. The matrix is dominated by the single largest singular value, and the principal feature associated with this singular value is constantly `-1.0` given different values of x.

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

Solution: See solution notebook. The gradient norms are very large.

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

Solution: See solution notebook. The training diverges due to numeric overflow, because the gradient norm is too large.

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

Solution: See solution notebook. The gradient norms of different layers are similar. The numbers are close to `0.4`.

(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)

Solution: See solution notebook. The main difference is that neural networks initialized with a large weight scale have larger scales for principal features and singular values compared to properly-initialized networks. However, their patterns turns out to be very similar except for scales.

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

Solution: See solution notebook. The model outperforms the shallow model significantly: the predicted function is more close to the true function.

(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).**

Solution: See solution notebook. In the shallow network, the principal features are pointwise linear w.r.t. x with fewer segments (usually less than 5). In this deep network, the principal features are also pointwise linear w.r.t. x, but they have much more segments. This phenomenon shows that the deep network can represent more complex functions.

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

Solution: See solution notebook. The model has good performance when $x < 0.4$. However, the model predicts the function to be a straight line that goes to infinity as x increases when $x > 0.4$, which is different from the true function. The model fails to capture the antisymmetry of the odd function f_{true}.

(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.**

Solution: See solution notebook. When $x < 0.4$, the behavior is quite similar to the deep network trained with full data. However, when $x > 0.4$ where there is no training data, almost all principal features degenerated to a straight line without any elbow.

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.

