EECS 182 Deep Neural Networks
Spring 2025 Anant Sahai

Homework 1

This homework is due on Sunday, Feb 9, 2025, 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. General Case Tikhonov Regularization

Consider the optimization problem:

$$\min_{\mathbf{x}} ||W_1(A\mathbf{x} - \mathbf{b})||_2^2 + ||W_2(\mathbf{x} - \mathbf{c})||_2^2$$

Where W_1, A, and W_2 are matrices and $\mathbf{x}$, $\mathbf{b}$ and $\mathbf{c}$ are vectors. W_1 can be viewed as a generic weighting of the residuals and W_2 along with c can be viewed as a generic weighting of the parameters.

(a) **Solve this optimization problem manually** by expanding it out as matrix-vector products, setting the gradient to $\mathbf{0}$, and solving for $\mathbf{x}$.

(b) **Construct an appropriate matrix $\mathbf{C}$ and vector $\mathbf{d}$ that allows you to rewrite this problem as**

$$\min_x \|C\mathbf{x} - \mathbf{d}\|^2$$

and use the OLS solution ($\mathbf{x}^* = (C^TC)^{-1}C^T\mathbf{d}$) to solve. Confirm your answer is in agreement with the previous part.

(c) **Choose a W_1, W_2, and $\mathbf{c}$** such that this reduces to the simple case of ridge regression that you've seen in the previous problem, $\mathbf{x}^* = (A^TA + \lambda I)^{-1}A^T\mathbf{b}$.

3. 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

$$\hat{\mathbf{w}} = (X^TX + \lambda I)^{-1}X^T\mathbf{y} \tag{5}$$

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

$$\hat{\mathbf{w}} = X^T(XX^T + \lambda I)^{-1}\mathbf{y}. \tag{6}$$

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.**

- (5) 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

$$E[\mathbf{W}|\mathbf{Y} = \mathbf{y}] = \Sigma_{WY}\Sigma_{YY}^{-1}\mathbf{y} \tag{7}$$

  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 $\mathbf{Y}$.

4. 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:

$$\underset{\mathbf{w}}{\operatorname{argmin}}\, \mathbb{E}[\|\check{X}\mathbf{w} - \mathbf{y}\|^2] \tag{8}$$

(a) **Show that this problem (8) is equivalent to a regularized least squares problem:**

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{m}\|X\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2 \tag{9}$$

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).)

5. Visualizing features from local linearization of neural nets

In the first discussion, you trained a 1-hidden-layer neural network with SGD and visualized how the network fitted the function leveraging the "elbows" of the non-linear activation function ReLU. In this question, we are going to visualize the effective "features" that correspond to the local linearization of this network in the neighborhood of the parameters.

We provide you with some starter code in the course repo, or you can use Google Colab. For this question, please submit the `.pdf` export of the jupyter notebook when it is completed. In addition, answer the questions below, including plots from the notebook where relevant.

(a) **Visualize the features corresponding to $\frac{\partial}{\partial w_i^{(1)}} y(x)$ and $\frac{\partial}{\partial b_i^{(1)}} y(x)$ where $w_i^{(1)}$ are the first hidden layer's weights and the $b_i^{(1)}$ are the first hidden layer's biases.** These derivatives should be evaluated at at least both the random initialization and the final trained network. When visualizing these features, plot them as a function of the scalar input x, the same way that the notebook plots the constituent "elbow" features that are the outputs of the penultimate layer.

(b) During training, we can imagine that we have a generalized linear model with a feature matrix corresponding to the linearized features corresponding to each learnable parameter. We know from our analysis of gradient descent, that the singular values and singular vectors corresponding to this feature matrix are important.

Use the SVD of this feature matrix to plot both the singular values and visualize the "principle features" that correspond to the d-dimensional singular vectors multiplied by all the features corresponding to the parameters.

(HINT: Remember that the feature matrix whose SVD you are taking has n rows where each row corresponds to one training point and d columns where each column corresponds to each of the learnable features. Meanwhile, you are going to be plotting/visualizing the "principle features" as functions of x even at places where you don't have training points.)

(c) Augment the jupyter notebook to add a second hidden layer of the same size as the first hidden layer, fully connected to the first hidden layer. **Allow the visualization of the features corresponding to the parameters in both hidden layers, as well as the "principle features" and the singular values.**

6. 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?**

Contributors:

- Anant Sahai.
- Sheng Shen.
- Saagar Sanghavi.

