EECS 182 Deep Neural Networks
Spring 2023 Anant Sahai

Homework 1

This homework is due on Friday, February 3, 2022, at 10:59PM.

1. Bias-Variance Tradeoff Review

(a) **Show that we can decompose the expected mean squared error into three parts: bias, variance, and irreducible error σ^2:**

$$\text{Error} = \text{Bias}^2 + \text{Variance} + \sigma^2$$

Formally, suppose we have a randomly sampled training set $\mathcal{D}$ (drawn independently of our test data), and we select an estimator denoted $\theta = \hat{\theta}(\mathcal{D})$ (for example, via empirical risk minimization). The expected mean squared error for a test input x can be decomposed as below:

$$\mathbb{E}_{Y\sim p(y|x),\mathcal{D}}[(Y - f_{\hat{\theta}(\mathcal{D})}(x))^2] = \text{Bias}(f_{\hat{\theta}(\mathcal{D})}(x))^2 + \text{Var}(f_{\hat{\theta}(\mathcal{D})}(x)) + \sigma^2$$

You may find it helpful to recall the formulaic definitions of Variance and Bias, reproduced for you below:

$$\text{Bias}(f_{\hat{\theta}(\mathcal{D})}(x)) = \mathbb{E}_{Y\sim p(Y|x),\mathcal{D}}[f_{\hat{\theta}(\mathcal{D})}(x) - Y]$$
$$\text{Var}(f_{\hat{\theta}(\mathcal{D})}(x)) = \mathbb{E}_{\mathcal{D}}\left[(f_{\hat{\theta}(\mathcal{D})}(x) - \mathbb{E}[f_{\hat{\theta}(\mathcal{D})}(x)])^2\right]$$

(b) Suppose our training dataset consists of $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ where the only randomness is coming from the label vector $Y = \mathbf{X}\theta^* + \boldsymbol{\varepsilon}$ where θ^* is the true underlying linear model and each noise variable ε_i is i.i.d. with zero mean and variance 1. We use ordinary least squares to estimate a $\hat{\theta}$ from this data. **Calculate the bias and covariance of the $\hat{\theta}$ estimate and use that to compute the bias and variance of the prediction at particular test inputs x.** Recall that the OLS solution is given by

$$\hat{\theta} = (X^\top X)^{-1} X^\top Y,$$

where $X \in \mathbb{R}^{n\times d}$ is our (nonrandom) data matrix, $Y \in \mathbb{R}^n$ is the (random) vector of training targets. For simplicity, assume that $X^\top X$ is diagonal.

2. Least Squares and the Min-norm problem from the Perspective of SVD

Consider the equation $X\mathbf{w} = \mathbf{y}$, where $X \in \mathbb{R}^{m\times n}$ is a non-square data matrix, w is a weight vector, and y is vector of labels corresponding to the datapoints in each row of X.

Let's say that $X = U\Sigma V^T$ is the (full) SVD of X. Here, U and V are orthonormal square matrices, and Σ is an $m \times n$ matrix with non-zero singular values (σ_i) on the "diagonal".

For this problem, we define $\Sigma^\dagger$ an $n \times m$ matrix with the reciprocals of the singular values ($\frac{1}{\sigma_i}$) along the "diagonal".

(a) First, consider the case where $m > n$, i.e. our data matrix X has more rows than columns (tall matrix) and the system is overdetermined. **How do we find the weights $\mathbf{w}$ that minimizes the error between $X\mathbf{w}$ and y?** In other words, we want to solve $\min_{\mathbf{w}} \|X\mathbf{w} - \mathbf{y}\|^2$.

(b) **Plug in the SVD $X = U\Sigma V^T$ and simplify.** Be careful with dimensions!

(c) You'll notice that the least-squares solution is in the form $\mathbf{w}^* = A\mathbf{y}$. **What happens if we left-multiply X by our matrix A?** This is why the matrix A of the least-squares solution is called the left-inverse.

(d) Now, let's consider the case where $m < n$, i.e. the data matrix X has more columns than rows and the system is underdetermined. There exist infinitely many solutions for w, but we seek the minimum-norm solution, ie. we want to solve $\min \|\mathbf{w}\|^2 \text{s.t.} X\mathbf{w} = \mathbf{y}$. **What is the minimum norm solution?**

(e) **Plug in the SVD $X = U\Sigma V^T$ and simplify.** Be careful with dimensions!

(f) You'll notice that the min-norm solution is in the form $\mathbf{w}^* = B\mathbf{y}$. **What happens if we right-multiply X by our matrix B?** This is why the matrix B of the min-norm solution is called the right-inverse.

3. The 5 Interpretations of Ridge Regression

(a) *Perspective 1: Optimization Problem.* Ridge regression can be understood as the unconstrained optimization problem

$$\underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2, \tag{1}$$

where $X \in \mathbb{R}^{n\times d}$ is a data matrix, and $\mathbf{y} \in \mathbb{R}^n$ is the target vector of measurement values. What's new compared to the simple OLS problem is the addition of the $\lambda\|\mathbf{w}\|^2$ term, which can be interpreted as a "penalty" on the weights being too big.

Use vector calculus to expand the objective and solve this optimization problem for $\mathbf{w}$.

(b) *Perspective 2: "Hack" of shifting the Singular Values.* In the previous part, you should have found the optimal $\mathbf{w}$ is given by

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

(If you didn't get this, you should check your work for the previous part).

Let $X = U\Sigma V^T$ be the (full) SVD of the X. Recall that U and V are square orthonormal (norm-preserving) matrices, and Σ is a $n \times d$ matrix with singular values σ_i along the "diagonal". **Plug this into the Ridge Regression solution and simplify. What happens when the singular values when $\sigma_i << \lambda$? What about when $\sigma_i >> \lambda$?**

(c) *Perspective 3: Maximum A Posteriori (MAP) estimation.* 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 $\mathbf{Y} = X\mathbf{W} + \sqrt{\lambda}\mathbf{N}$. Note that the X matrix whose rows are the n different training points are not random.

Show that (1) is the MAP estimate for $\mathbf{W}$ given an observation $\mathbf{Y} = \mathbf{y}$.

(d) *Perspective 4: Fake Data.* Another way to interpret "ridge regression" is as the ordinary least squares for an augmented data set — i.e. adding a bunch of fake data points to our data. Consider the following augmented measurement vector $\hat{\mathbf{y}}$ and data matrix $\hat{\mathbf{X}}$:

$$\hat{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_d \end{bmatrix} \quad \hat{\mathbf{X}} = \begin{bmatrix} X \\ \sqrt{\lambda}\mathbf{I}_d \end{bmatrix},$$

where $\mathbf{0}_d$ is the zero vector in $\mathbb{R}^d$ and $\mathbf{I}_d \in \mathbb{R}^{d\times d}$ is the identity matrix. **Show that the classical OLS optimization problem** $\operatorname{argmin}_{\mathbf{w}} \|\hat{\mathbf{y}} - \hat{\mathbf{X}}\mathbf{w}\|_2^2$ **has the same minimizer as** (1)**.**

(e) *Perspective 5: Fake Features.* For this last interpretation, let's instead construct an augmented design matrix in the following way:

$$\check{\mathbf{X}} = [X \ \sqrt{\lambda}\mathbf{I_n}]$$

i.e. we stack X with $\sqrt{\lambda}\mathbf{I}_n$ horizontally. Now our problem is underdetermined: the new dimension $d+n$ is larger than the number of points n. Therefore, there are infinitely many values $\boldsymbol{\eta} \in \mathbb{R}^{d+n}$ for which $\check{\mathbf{X}}\boldsymbol{\eta} = \mathbf{y}$. We are interested in the **min-norm** solution, ie. the solution to

$$\operatorname*{argmin}_{\boldsymbol{\eta}} \|\boldsymbol{\eta}\|_2^2 \text{ s.t. } \hat{\mathbf{X}}\boldsymbol{\eta} = \mathbf{y}. \tag{2}$$

Show that this is yet another form of ridge regression and that the first d **coordinates of** $\boldsymbol{\eta}^*$ **form the minimizer of** (1)**.**

(f) We know that the Moore-Penrose pseudo-inverse for an underdetermined system (wide matrix) is given by $A^\dagger = A^T(AA^T)^{-1}$, which corresponds to the min-norm solution for $A\boldsymbol{\eta} = \mathbf{z}$. That is, the optimization problem

$$\operatorname{argmin} \|\boldsymbol{\eta}\|^2 \text{s.t.} A\boldsymbol{\eta} = \mathbf{z}$$

is solved by $\boldsymbol{\eta} = A^\dagger\mathbf{z}$. Let $\hat{\mathbf{w}}$ be the minimizer of (1).

Use the pseudo-inverse to show that solving to the optimization problem in (2) **yields**

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

Then, show that this is equivalent to the standard formula for Ridge Regression

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

Hint: It may be helpful to review Kernel Ridge Form.

(g) We know that the solution to ridge regression (1) *is given by* $\hat{\mathbf{w}}_\mathbf{r} = (X^\top X + \lambda\mathbf{I})^{-1}X^\top\mathbf{y}$. **What happens when** $\lambda \to \infty$**?** *It is for this reason that sometimes ridge regularization is referred to as "shrinkage."*

(h) **What happens to the solution of ridge regression when you take the limit** $\lambda \to 0$**?** *Consider both the cases when X is wide (underdetermined system) and X is tall (overdetermined system).*

4. 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, $\mathbf{x}^* = (A^TA + \lambda I)^{-1}A^Tb$.

5. Coding Fully Connected Networks

In this coding assignment, you will be building a fully-connected neural network from scratch using NumPy. 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/gonglinyuan/cs182hw1` and refer to `README.md` for further instructions.

For this question, please submit a .zip file your completed work to the Gradescope assignment titled "Homework 1 (Code)". Please answer the following question in your submission of the written assignment:

(a) **Did you notice anything about the comparative difficulty of training the three-layer net vs training the five layer net?**

6. Visualizing features from local linearization of neural nets

This problem expects you to modify the Jupyter Notebook you were given in the first discussion section for the course to allow the visualization of the effective "features" that correspond to the local linearization of the network in the neighborhood of the parameters.

We provide you with some starter code on Google Colab. For this question, **please do *not* submit your code to Gradescope**. Instead, just include your plots and comments regarding the questions in the subparts.

(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.**

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:

- Brandon Trabucco.
- Saagar Sanghavi.
- Alexander Tsigler.
- Anant Sahai.
- Jane Yu.
- Philipp Moritz.
- Soroush Nasiriany.
- Linyuan Gong.
- Sheng Shen.

