CIS5200: Machine Learning Spring 2023

Homework 5

Release Date: April 17, 2023 *Due Date: April 26, 2023*

- HW5 will count for 10% of the grade. The written part of the homework is based on reading a paper and uses some part of the programming assignment. The programming contains some additional questions.
- All written homework solutions are required to be formatted using LaTeX. Please use the template here. Do not modify the template. This is a good resource to get yourself more familiar with LaTeX, if you are still not comfortable.
- You will submit your Answer for the written part of HW5 as a single PDF file via Gradescope. The deadline is **11:59 PM ET**. Contact TAs on Ed if you face any issues uploading your homework.
- Collaboration is permitted and encouraged for this homework, though each student must understand, write, and hand in their own submission. In particular, it is acceptable for students to discuss problems with each other; it is not acceptable for students to look at another student's written Answers when writing their own. It is also not acceptable to publicly post your (partial) Answer on Ed, but you are encouraged to ask public questions on Ed. If you choose to collaborate, you must indicate on each homework with whom you collaborated.
- **Optional Question**: We have added one optional question in this homework for **no** credit. This is intended to be purely for interest.

Please refer to the notes and slides posted on the website if you need to recall the material discussed in the lectures.

1 Programming Questions (22 points + 3 bonus points)

Use the link here to access the Google Colaboratory (Colab) file for this homework. Be sure to make a copy by going to "File", and "Save a copy in Drive". As with the previous homeworks, this assignment uses the PennGrader system for students to receive immediate feedback. As noted on the notebook, please be sure to change the student ID from the default '99999999' to your 8-digit PennID.

Instructions for how to submit the programming component of HW 5 to Gradescope are included in the Colab notebook. You may find this PyTorch linear algebra reference and this general PyTorch reference to be helpful in perusing the documentation and finding useful functions for your implementation.

2 Written Questions (30 points)

In this homework you will read the paper Reconciling modern machine learning practice and the bias-variance trade-off, relate it to the concepts you have studied in class so far, and get a deeper understanding of the double descent phenomenon.

Note: You do not need a very through understanding of the entire paper, use it as a reference. Appendix is optional. Recommendation is to read the assignment before reading the paper.

The main goal of the paper is to resolve the apparent contradiction between the classical bias-variance trade-off theory (that we studied in class) and the success of modern machine learning models, particularly deep learning.

Bias-Variance-Noise Decomposition (5 points). First derive the bias-variance-noise decomposition for the squared loss. In particular, show that for any algorithm $\mathcal{A}$ that takes in input training set $S \in \mathcal{D}^m$ and outputs predictor f_S,

$$\mathbb{E}_{x,y,S}[(f_S(x)-y)^2] = \underbrace{\mathbb{E}_{x,S}[(f_S(x)-\bar{f}(x))^2]}_{①} + \underbrace{\mathbb{E}_{x,S}[(\bar{f}(x)-\bar{y})^2]}_{②} + \underbrace{\mathbb{E}_{y}[(y-\bar{y})^2]}_{③}$$

where $\bar{y} = \mathbb{E}[y|x]$ and $\bar{f} = \mathbb{E}_S[f_S]$ is the average model obtained by drawing infinite training sets, learner a predictor on each of them and averaging their predictions.. What do each of the terms ①, ②, ③ correspond to among $\{\text{noise}, \text{bias}^2, \text{variance}\}$? Explain your reasoning.

Answer:

Classical Bias-Variance Trade-off (3 points). Recall that the bias-variance trade-off suggests that as model complexity increases, the bias decreases while the variance increases, leading to a trade-off between *underfitting* and *overfitting*. Explain this in the context of the bound you derived above.

Answer:

Modern Bias-Variance Trade-off (2 points). Contrary to the bias-variance trade-off you described above, when we train very large (over-parameterized) models in deep learning, they tend to generalize well, despite their high complexity. Moreover, despite exactly fitting the data, they still get low generalization error. Explain which terms in the above decomposition would you expect to be small in this setting.

Answer:

Interpolation Threshold (5 points). The paper uses a notion of *interpolation threshold* to refer to the point at which a model becomes complex enough to perfectly fit the training data, achieving zero training error. This interpolation threshold is the point after which the bias-variance curve is different from the classical curve.

In order to further understand the interpolation threshold, consider the problem of fitting polynomials to a 1-dimensional dataset. Suppose you have a training dataset of m points $S\{(x_1, y_1), \dots, (x_m, y_m)\}$ where $x_i, y_i \in \mathbb{R}$. Show that there exists a polynomial p of degree m ($p(x) = \sum_{i=0}^{m} \alpha_i x^i$ for some $\alpha_i \in \mathbb{R}$) that gets zero error on the training set S? Assume $x_i \neq x_j$ for all $i \neq j$.

Answer:

If you think of degree as a measure of complexity, then the interpolation threshold here is m. The paper shows several other settings (RFF, Random Forests) where the interpolation threshold (in terms of appropriate complexity) is equal to the number of samples.

Double Descent. At the interpolation threshold, the test error is large. Below this interpolation threshold, we are back to the classical U-shaped regime. Above this interpolation threshold, we are in the modern regime, where despite exactly fitting the data, the test error actually improves. The authors call this the *double descent* phenomenon. They exhibit this phenomenon in several settings including random features, neural networks, decision trees, etc.

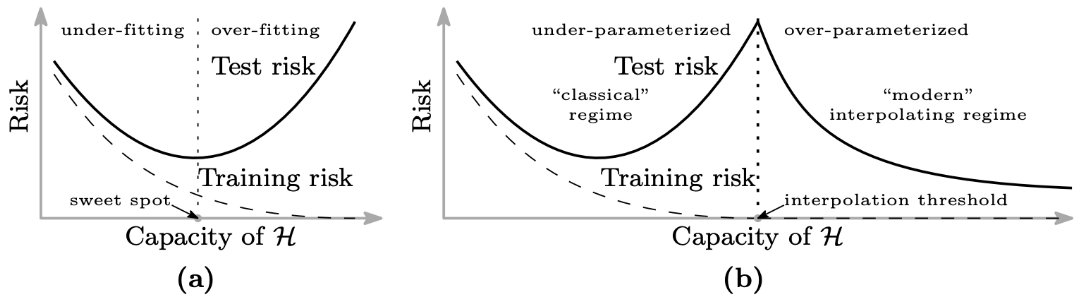

Figure 1: Classical vs Modern Regime of the bias-variance trade-off. *Src: Figure 1 from the paper.*

Double Descent in Fully Connected Network. (**3 points**) Let us look at the the experiments for fully connected networks in the paper (Section C.3). The class of single hidden layer fully connected networks is parameterized by hidden layer weights $w_1, \ldots, w_k \in \mathbb{R}^d$ (assume 0 bias term), activation function $\sigma : \mathbb{R} \to \mathbb{R}$, and combining weights $\alpha_1, \ldots, \alpha_k \in \mathbb{R}$:

$$f(x; W, \alpha) = \sum_{i=1} \alpha_i \sigma(w_i^\top x_i).$$

Here k is the size of the hidden layer (the number of hidden units).

In class we saw that,

$$\text{generalization error} \lesssim \sqrt{\frac{\text{complexity}(\mathcal{F})}{m}}.$$

Considering the above function class and the complexity as the total number of scalar parameters (for e.g. each w_i has d scalar parameters, each α_i is one scalar parameter), how large can we take k before the above bound becomes vacuous (that is, right hand side is ≥ 1)?

Answer:

However, the experiments in the paper show that increasing k beyond this number does not make the generalization error large.

Double Descent with Random ReLU Features (**4 + 4 + 2 points**). We will replicate the double descent curve in the setting of Random ReLU features. You can think of Random ReLU features as a random feature map ϕ where each feature is generated by sampling a vector $w \sim \mathcal{N}(0, I_d)$ and then applying the ReLU activation ($a \mapsto \max(0, a)$). More formally, for $x \in \mathbb{R}^d$,

$$\phi(x)_i = \max(w_i^\top x, 0) \text{ for } i \in [k].$$

Increasing the value of k generates more features and increases the complexity of this class.

Use the code in the programming notebook to replicate the double descent plot and add it below.

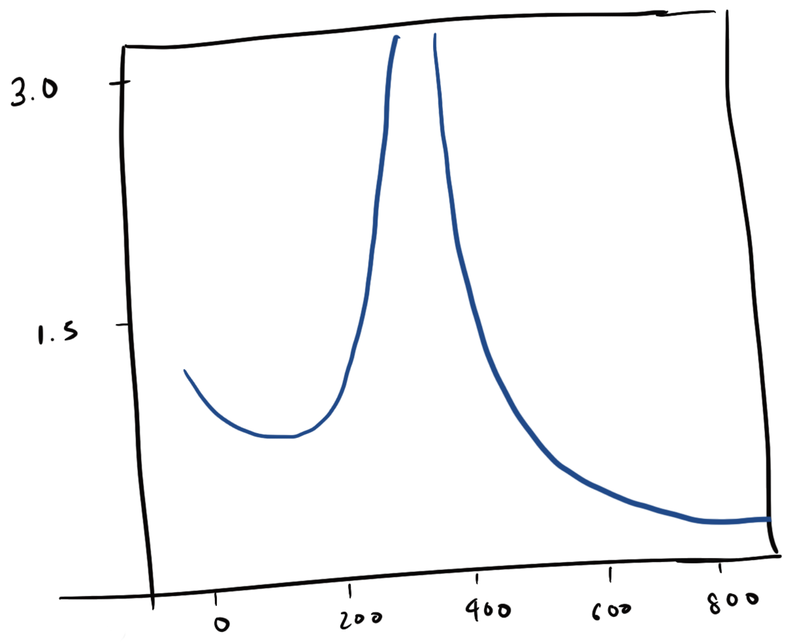

Figure 2: Double Descent for Random ReLU features

Now, instead of plotting number of parameters on the x-axis, generate the plot of test error versus norm of learned predictor and add it below. Use the same runs as before, and consider only settings where you get test error < 0.5.

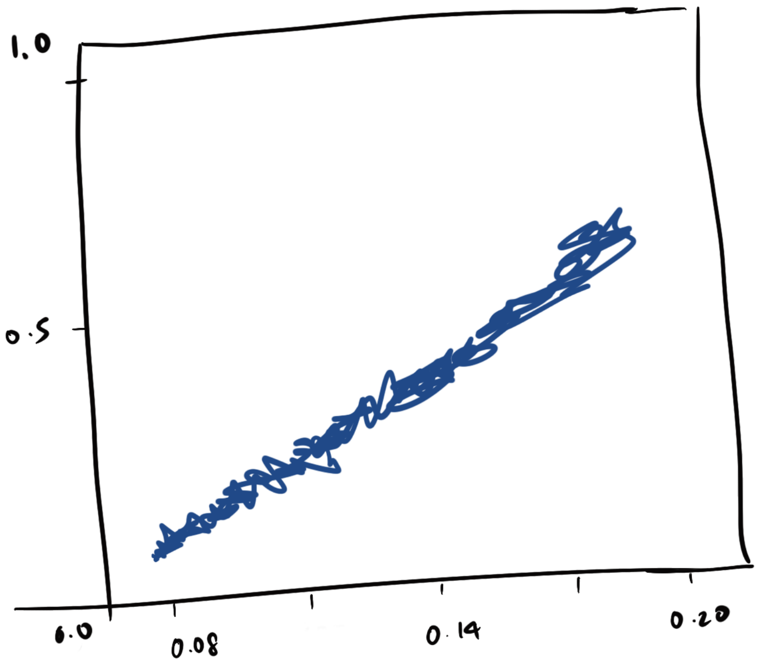

Figure 3: Norm versus Test-error for Random ReLU features

The curve does not look like the double descent curve anymore. Based on this plot, how can we use classical theory to explain why we get good generalization performance despite large parameter size?

Answer:

Explicit regularization (2 points). The above is an example of implicit regularization where we did not add a regularization term to find low norm solutions, but the algorithm implicitly regularized the solution. Suggest two ways in which we could explicitly regularize while training in the neural network setting to ensure better generalization performance?

Answer:

Optional: Minimum Norm Interpolators (no points). *The intention of this section is only for those curious to understand the theory behind implicit regularization. It is based on a 2020 paper, and is challenging.*

Here we will consider a simplified setting to better understand implicit regularization. We will look at the task of linear regression in a setting with a very large input space, and the number of samples being much smaller than the dimension.

Our data will be drawn in the following manner:

- The input $x = (x_S, x_J) \in \mathbb{R}^d$ where the first part $x_S \sim \mathcal{N}(0, I_{d_S})$ and $x_J \sim \mathcal{N}(0, \eta I_{d_J})$ where $\eta = \lambda / d_J$.
- The label $y = w^* x + \epsilon$ where $w^* = (w^*_S, 0)$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

Note that the label only depends on x_S therefore we call this the signal part and we call x_J the junk part. Now suppose we run linear regression on a dataset $X \in \mathbb{R}^{m \times d}$ with labels $Y \in \mathbb{R}^m$ with $m \leq d$, and find the minimum norm solution that interpolates.

$$\hat{w} = \arg\min_{w: Xw = Y} \|w\|_2.$$

Show that

$$\hat{w} = X^\top (X X^\top)^{-1} Y.$$

Answer:

For any x drawn according to the distribution, show that as $d_J \to \infty$,

$$\hat{w}^\top x \to [1] \hat{w}_\lambda^\top x_S,$$

where $\hat{w}_\lambda$ is the solution of a regularized problem on only X_S, that is,

$$\hat{w}_\lambda = \arg\min_{w} \|Y - X_S w\|_2^2 + \lambda \|w_S\|^2.$$

You can use the fact that as $d_J \to \infty$, any x drawn according to the distribution satisfies $X_J x = 0$.

Answer:

Observe that this result tells us that increasing the number of parameters acts as an implicit regularizer to the original problem. Since we have noise in our problem, the regularized solution will get us better generalization performance.

[1] Technically $\to$ should be converges almost surely, but for this homework ignore this.