COMS 6998: Practical Deep Learning System Performance: Reading 1

Geraldi Dzakwan (gd2551)

28 September 2020

Question 1

Model selection aims to rank different models, in terms of a certain type of performance (i.e. accuracy), for a certain problem. If the bias affects all the models in consideration equally, then biased performance estimates are acceptable since the ranking order won't change anyway. For example, say we have two models, having 70% and 80% accuracy consecutively. If, anyhow, we overestimates both performance by, let say, 15%, we will still choose the second model regardless of that optimistic bias.

On the other hand, when reporting generalization performance, accurate (unbiased) performances estimates are crucial. Unbiased means that we should evaluate our models using a test set that represents the actual/real life data as close as possible because we expect that the generalization performance to accommodate the actual data (including unseen data). So, if we have pessimistic bias (underfitting) or optimistic bias (overfitting), it won't picture how the model actually performs in the real life data.

Question 2

1. When doing random sampling for holdout validation, there's a high chance that we obtain train set and test set in which the class proportion doesn't represent the actual proportion. For example, suppose that in the dataset we have two classes, each accounts for 50% of the population. When we do random sampling, we might obtain different proportions, e.g. the train set has 40% : 60% class proportion and the test set has the opposite, which is 60% : 40% class proportion. This is a problem because then our model might not perform well in the actual/real life data because it learns and is evaluated on data with different class proportions.

2. Doing a holdout validation introduces a dilemma in regards to whether we should refit the best model using the full dataset (train + test) after we're

done with the model selection. If we don't refit, we might lose the chance to increase the "capacity" of our model, meaning that we have pessimistic bias in our model. If we refit, we lose our test set, so we couldn't estimate the generalization performance of the refit model.

Question 3

Model parameters are learned parameters, meaning that, during the course of the learning process, their value will change from their initial value to their final value once the training is done. Hyperparameters, on the other hand, have a fixed value over the course of the training process, which values are usually handpicked at the start of the learning process. Hyperparameters will highly influence the final value of the model parameters and the learning process, e.g. for some algorithms, hyperparameter values could influence training time.

Let's pick a specific type of linear regression algorithm which is Elastic Net. Reference: https://en.wikipedia.org/wiki/Elastic_net_regularization. Below are the model parameters and the hyperparameters:

1. This algorithm is basically a linear regression algorithm. So, suppose we have d features, denoted by $x_1, x_2, x_3, ..., x_d$, then the model parameters would be:

   $$\beta_0, \beta_1, \beta_2, \beta_3, ..., \beta_d$$

   where β_0 is a constant and the rest of them are coefficients for each feature, forming this equation:

   $$f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_d x_d$$

2. There are two hyperparameters for Elastic Net, which are λ_1 and λ_2, that makes Elastic Net has a different loss function to that of a simple linear regression. Suppose y is the label, X is the data vector: $((1, x_1, x_2, ..., x_d))$ and β is the coefficients including bias: $(\beta_0, \beta_1, \beta_2, ..., \beta_d)$, the loss function for a data point is:

   $$L(\beta) = ||y - X\beta||^2 + \lambda_1||\beta|| + \lambda_2||\beta||^2$$

   We can see that not only minimizing bias $||y - X\beta||^2$, but Elastic Net also tries to minimize L1 and L2 value of the solution vector β, picking solution with the least variance. Thus, both λ_1 and λ_2 affect the value of the model parameters $\beta : (\beta_0, \beta_1, \beta_2, ..., \beta_d)$.

Question 4

1. Computationally, 3-way holdout is far less expensive. When using 3-way holdout for hyperparameter tuning using validation set, we only fit each

model once, i.e. fit on the train set and evaluate on the validation set. Meanwhile, when using k-fold cross validation, the bigger the k, the more heavy the computation. We need to fit the model k times on k train sets and evaluate it k times on k validation sets. Thus, because Deep Neural Networks (DNN) often involve heavy computation, 3-way holdout is more suitable in terms of resources usage.

2. Variance wise, we might not need to worry too much of the possibility of ending up with a high variance in our model when using 3-way holdout method for Deep Neural Networks (DNN). Most of the time, DNN is used for a relatively large dataset. Hence, even we only fit our model once using one train set, we might cover most of the examples from the real distribution, resulting in a model with relatively low variance. To add, our model is less sensitive to changes in split scenario (3-way holdout vs k-fold cross val) because the dataset is large, so we might as well use one that is computationally cheap.

Question 5

The classical U-shaped curve in bias–variance trade-off implies that a model/hypopthesis should balance underfitting and overfitting through the sweet spot. It should be rich enough to express underlying structure in data (low bias) and simple enough to avoid fitting spurious patterns (low variance). Below is the hand drawn curve with marks:

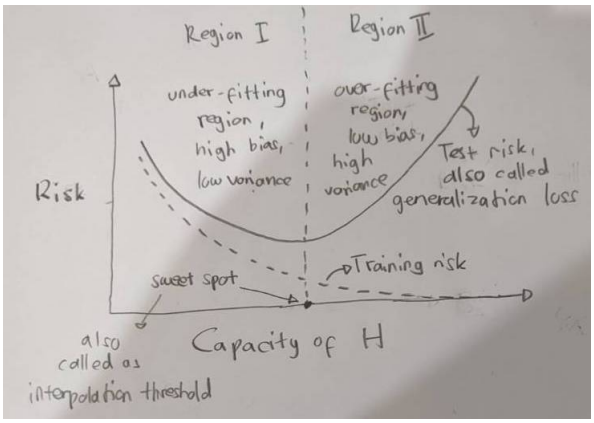

Figure 1: Bias Variance Tradeoff U-Shaped Curve

The elements in the picture are:

1. The test risk, or the generalization loss, is depicted by the solid line.
2. The train risk, or the training loss, is depicted by the dashed line.

3. The sweet spot/interpolation threshold is the dashed line in the middle.

4. The region left to the sweet spot has high bias, low variance and underfits.

5. The region right to the sweet spot has low bias, high variance and overfits.

The interpolation threshold in this U-shaped curve signifies that the model which has reached above that threshold has zero training risk, but it might possess high variance and isa highly possible to overfit.

Question 6

The Double Descent curve is basically an extension to the classical U-shaped curve, but with new/different interpolation threshold. When the capacity of the model is below this new interpolation threshold, the model behaves like that of the U-shaped curve. Capacity associates with the number of parameters used in the model. Meanwhile, when the model capacity is above this new interpolation threshold, not only it stays with extremely low (near zero) training loss, but in fact, the generalization loss also decreases, and thus not overfits anymore.

This is because the model capacity does not necessarily reflect how well the model matches a certain problem's inductive bias. Instead, an appropriate inductive bias could be the smoothness of a function, measured by some space norm (a form of Occam's razor). By exploring space with larger capacity than the new interpolation threshold, in which there are more appropriate candidate models, there will be interpolating models with smaller norms. Thus, they interpolate, but are simple (not overfitting). Below is the hand drawn curve with marks:

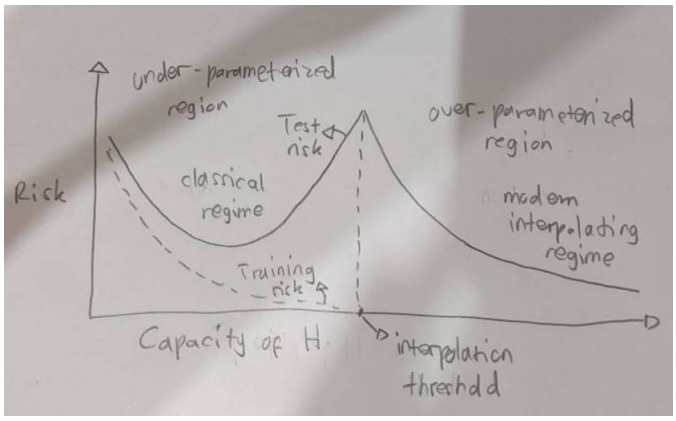

Figure 2: Double Descent Curve

The elements in the picture are:

1. The region left to the interpolation threshold is the under-parameterized region.

2. The region right to the interpolation threshold is the over-parameterized region.

Question 7

Some reasons on why it's difficult to observe double descent curve when using deep learning:

1. The computational cost of Empirical Risk Minimization (ERM) when using deep learning is high.

2. When we start with far less parameters than number of data ($N << n$), we have non-convexity behavior, i.e. the local search-based heuristics, which highly depends on the parameter initialization. Thus, if for this ERM we don't find an optimal solution, then increasing the network size, i.e. the N, doesn't guarantee a decrease of the risk. We might have a fluctuation in both the training and testing risks which makes it difficult to observe the double descent curve.

The size of the networks should be around the multiplication of the number of data points and the classes, which is $10^9 * 10^3 = 10^{12}$. Thus, we need to have around 10^{12} parameters in our network to be able to interpolate.