Generalization through Prior Knowledge

Liam Hazan, Eyal Bar-Natan

September 17, 2020

Abstract

In this research the main topic we reviewed is the use of prior knowledge to improve generalization. Frequently, in machine-learning we use our knowledge to get better generalization. The reason we use methods like regularization, early stopping, dropout and more is because we know, from our experience that simpler explanations/models tend to generalize better than the complicated ones. For example, given n data points for regression we would prefer a line or parabola but not a $n-1$ degree polynomial even though it would get zero loss on training data.
Our work deals with approaches like using generalization bounds to choose the optimal complexity level for our model and through an experiment - extracting the relevant features out of our data to avoid fitting to meaningless features.
In addition, a significant subject that is still being researched in the machine-learning theory field, is the gap between the high accuracy in fitting the test data in practice, achieved by very rich models such as neural networks, to the lack of theoretical explanations for this phenomena. We will review this gap and would offer different kinds of approaches to it, that were done in multiple researches.

1 Introduction

The target in supervised learning is to find a model/function that maps input to some output based on given input-output pairs known as training data. One challenge is to determine the complexity of the model; choose the degree of a polynomial kernel, the number of hidden nodes in a neural network, the depth of a decision tree and so on.

As visualized in Figure 1, usually as the model complexity increases, the model becomes more "flexible" and able to fit the training data better, but the generalization error (test error) tend to decrease to some optimal point and then increase, creating the classical U-shaped risk curve. We will focus on finding that **optimal complexity parameter**, depending on the amount of data in hand and the estimation confidence we want to achieve.

As known, there are some model selection methods, such as cross-validation, which have some disadvantages we want to avoid here. We would like to have a method which demands moderate computation complexity, as well as allowing us to train a model on the full training data available instead of "wasting" part of it for validation set.
We will introduce a method to find these classical U-shaped optimal model complexity parameters.

Notation

$\mathcal{X} \times Y \sim D$: D is the probability distribution over $\mathcal{X}$ (input space) and Y (binary output space).
$S = \{(x_i, y_i)\}_{i=1}^m$: S is the training set and $|S| = m$.
$h : \mathcal{X} \to Y$, is a mapping function from input space to output space (label) and h_S is the learned classifier.
$\mathcal{H}$ is a set of these functions which called hypothesis class.
$L_S(h) = \frac{1}{m}\sum_{i=1}^m \mathbb{1}(h(x_i) \neq y_i)$: the empirical error (training error),
$L_D(h) = \mathbb{E}_{(x,y)\sim D}[\mathbb{1}(h(x) \neq y)]$: the generalization error.

2 Generalization Bounds

A generalization error bound provides a high confidence bound on the (absolute) difference $L_D(h_S) - L_S(h_S)$. The general form of such error bound is $L_D(h_S) - L_S(h_S) \leq \sqrt{\frac{N}{m}}$, while N is the complexity parameter. We would like to discuss bounds for finite and infinite function classes.
By using the **Uniform error bound for finite $\mathcal{H}$ Theorem**, we receive that with probability at least $1-\delta$:
$L_D(h_S) - L_S(h_S) \leq \sqrt{\frac{\ln|\mathcal{H}| + \ln(1/\delta)}{2m}}$.

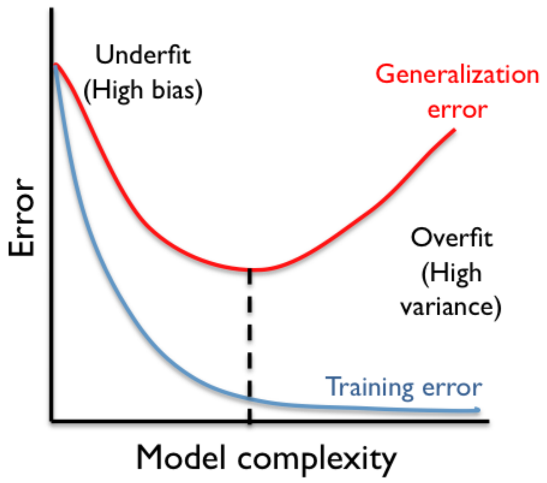

Figure 1: Classical Error VS. Capacity

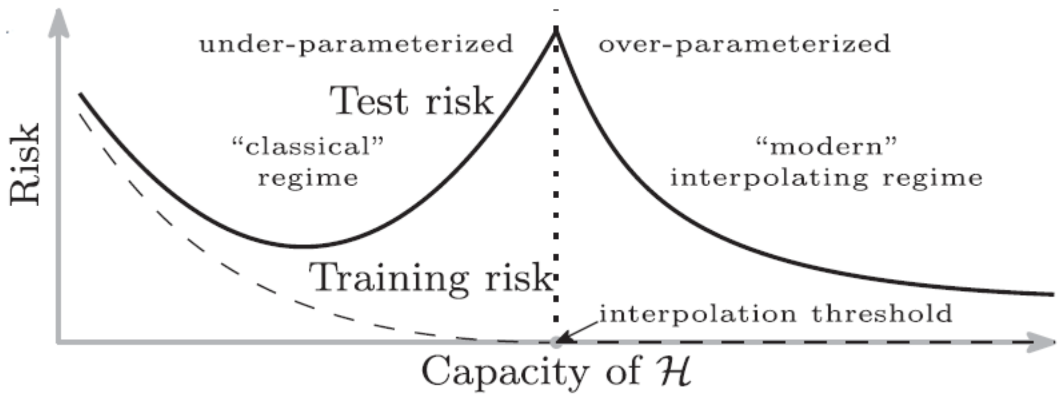

Figure 2: Modern Error VS. Capacity

For achieving a bound for an infinite class, we need to use a different notion to measure the capacity of $\mathcal{H}$. Therefore, we will use the **VC dimension**, denoted by VCdim($\mathcal{H}$), which is the cardinality of the largest set of points in $\mathcal{X}$ that can be shattered[1] by $\mathcal{H}$. If $\mathcal{H}$ shatters arbitrarily large sets of points in $\mathcal{X}$, then VCdim($\mathcal{H}$) $= \infty$.

Now by using the **Uniform error bound for general $\mathcal{H}$ Theorem**, we receive that with probability at least $1 - \delta$:

$$L_D(h_S) - L_S(h_S) \leq \sqrt{\frac{8\left(VCdim(\mathcal{H})\cdot\left(\ln(2m)+1\right)+\ln(4/\delta)\right)}{m}}.$$

Thus, one can select a model by training classifiers from different function classes on the given training data. Then compare the relevant upper bounds we saw, and choose the classifier with the smallest bound value.
However, an attempt to use such methods for deep neural networks (DNNs), would be meaningless, when having many more parameters than available training data. That is to say, the generalization bounds would be quantitatively vacuous.
Therefore, let us expand our view on generalization in rich models such as neural networks, thus we will try to explain how in modern practice, these models are trained to exactly fit (i.e., interpolate) the data. Classically, such models would be considered overfitted (for example according to the bias–variance trade-off), and yet they often obtain high accuracy on test data. Then, we will briefly review some **other complexity measures**[2] that help us to achieve and explain this observed **generalization phenomena** in deep learning.

3 Double-Descent Generalization

We will start with presenting the extension to the classical curve showed in Figure 1, published in

[1]all possible 2^m binary labelings of the points can be realized by functions in $\mathcal{H}$.

a work by Belkina, Hsuc, Maa and Mandala(1). In Figure 2 (taken from their work), we can notice the double-descent risk curve, which incorporates the classical U-shaped risk curve together with the observed behavior (which is detailed in their paper(1)) from using high capacity function classes like deep neural networks (i.e. the “modern” interpolating), separated by the **interpolation threshold**. When the function class capacity increases high enough (e.g. by increasing the number of features or the size of the neural network architecture), the learned predictors (parameters) achieve (almost) perfect fits to the training data— which means, interpolation. Although the learned predictors obtained at the interpolation threshold typically have high test risk (error), they showed in their work(1) that increasing the function class capacity beyond this point, leads to decreasing risk, typically going **below the risk achieved at the optimal complexity parameter** in the “classical” curve. To the right of the interpolation threshold, where the parameters number N equals to the number of data points m, all function classes are rich enough to achieve zero training risk. They found that in order to achieve the best possible generalization, one has to choose a function with the appropriate regularity factors that perfectly fits observed data. This is actually a form of Occam’s razor: The simplest explanation compatible with the observations should be preferred. By considering larger function classes, which contain more candidate predictors compatible with the data, we are able to find interpolating functions that have **smaller norm** and are thus “simpler”. Thus, increasing function class capacity improves performance of classifiers, which was shown in their paper.

4 Candidates for generalization in DNNs

Neyshabur, Bhojanapalli, McAllesterand and Srebro suggested(2) some explanations for this phenomena. Let us review those candidates briefly.

Norms and Margins

Capacity of linear predictors can be controlled independent of the number of parameters, e.g. through regularization of its l_2 norm. Similar **norm based** complexity measures have also been established for feed-forward neural networks with ReLU activations. A work by Bartlett et al. proves generalization bounds with capacity is proportional to $\Pi_{i=1}^{d} \|W_i\|_2^2 \left(\Sigma_{j=1}^{d}(\|W_j\|_1 / \|W_j\|_2)^{2/3}\right)^3$, where W is the parameters of a d layer feed-forward network.

However, when the training error goes to zero, in order to minimize any positive loss that diminish at infinity (as cross-entropy), the outputs of the network, along with the norm of the weights, must go to infinity. Therefore, the scaling of the network outputs, should be taken into account in this problem. They defined that for a given training set and small value $\epsilon > 0$, the margin γ_{min} is the lowest value of γ such that $\lceil \epsilon m \rceil$ data points have margin lower than γ, where a 'margin' γ (for a single data point) is the difference between the score of the correct label and the maximum score of other labels.
Their conclusion(2) was that this margin should be taken into account (inversely) when bounding the capacity of the network, using different norm expressions.

Sharpness

They notion of sharpness as a generalization measure (as visualized in Figure 3) was recently suggested by Keskar et al.(3) and corresponds to robustness to adversarial perturbations on the parameter space:
$\max_{|\nu_i| \le \alpha(|w_i|+1)} \hat{L}(f_{w+\nu}) - \hat{L}(f_w)$, where $\hat{L}(f_{w+\nu})$ is the training error of the function computed by ReLU activations, on the parameters w pertubated by random variable ν. In their research(2) they concluded that expected sharpness in the context of the PAC-Bayesian framework (which will not be furthered discussed), would be sufficient to control the capacity of the network. Together, sharpness and norm do provide capacity control and can explain many of the observed phenomena.

5 Feature Extraction

Another way to improve the model’s generalization ability (i.e. reduce overfitting) is **feature selection**. Redundant features can cause the model to use them to fit the training data, although they can be completely meaningless in regard to the granted mission.

When given more complicated data like images for classification, there is no way to do feature selection. Every pixel can sometimes contain relevant information (e.g. part of the edge defines the main object in the image) and some other times irrelevant information (e.g. part of the background). In cases like that we would want to extract/learn the ”right” features from the data, like edges and shapes which define the main object in the image.
As Zhang et al. introduced in their research(4), DNNs can reach perfect classification even on random pixels, nevertheless, in practice they generalize well. Despite of their success there is still

need for specialized architectures for certain problems. That leads us to our experiment where we are trying to compare between an architecture that takes advantage of the data's structure and extracts features accordingly (convolutional network - CNN), and another architecture that makes no assumptions on the data (multi-layered perceptron - MLP).

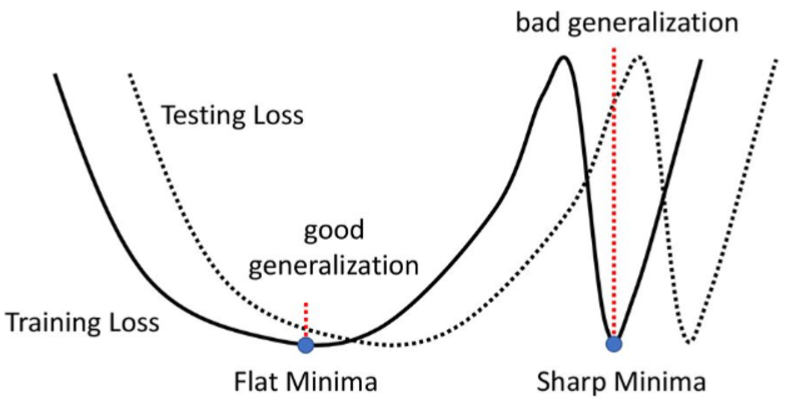

Figure 3: Sharp Minima VS. Flat Minima

6 Experiment

For the experiment we generated 5000 32×32 black and white images containing squares and circles with random placement in the image. The mission is a simple binary classification, and for the comparison, both of the models have about the same amount of parameters (∼400k). The models architecture can be seen in Figure 7. In order to confirm that both models have the capacity to "remember" that amount of data, we trained them and replaced the real labels with random ones. As can be seen in Figure 4, the models can not perfectly fit the data. But simply because there are limited options for each class, there can be 2 or more exact same images with different labels. With the addition of a Gaussian noise, we can see they do have the capacity to "remember" any sample they have seen.

In Figure 5 we can see a comparison between MLP and CNN. In order to check what will improve the MLP results, we tried to expand the number of parameters (∼8 Million) and to enlarge the dataset size (16k samples). Results are displayed in Figure 6.

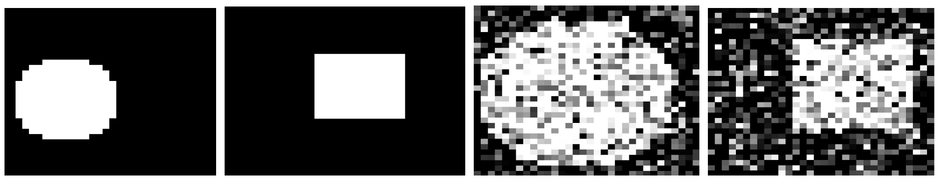

Figure 4: shapes with and without noise

7 Conclusions

- We can see that the number of parameters is not sufficient for measuring a model complexity, as we can see that CNN does a lot better because it is less "flexible" and specialized to extract location invariant features, which is needed in image classification. On the contrary, the MLP doesn't look for specified features, there is no assumptions made so it need to learn that there is no meaning for the location for every location in the image which requires more data. That can explain the reason that with random labels, where there is nothing to learn, the MLP reached a perfect fit, before the CNN did.
- The **Universal Approximation Theorem** claims that our MLP (i.e. neural network with one hidden layer) given unlimited width can, represent any function, including the one defined by our CNN. That can lead us to think that if we enlarge our MLP hidden dimension, it would generalize like the CNN. But in the experiment we tried to do

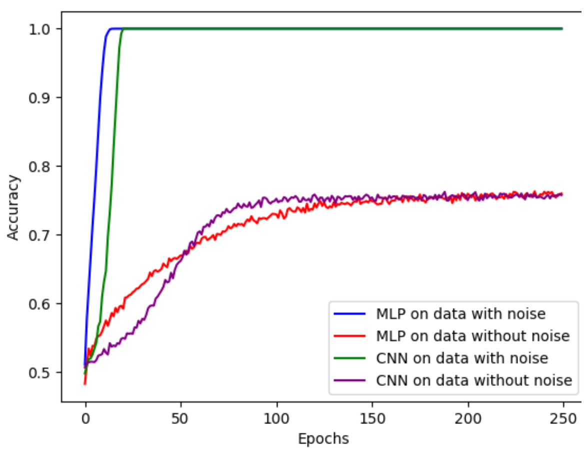

Figure 5: Results with random labels

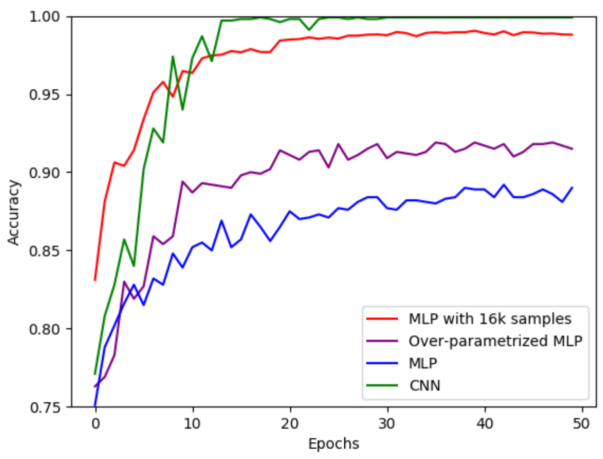

Figure 6: Performance comparison

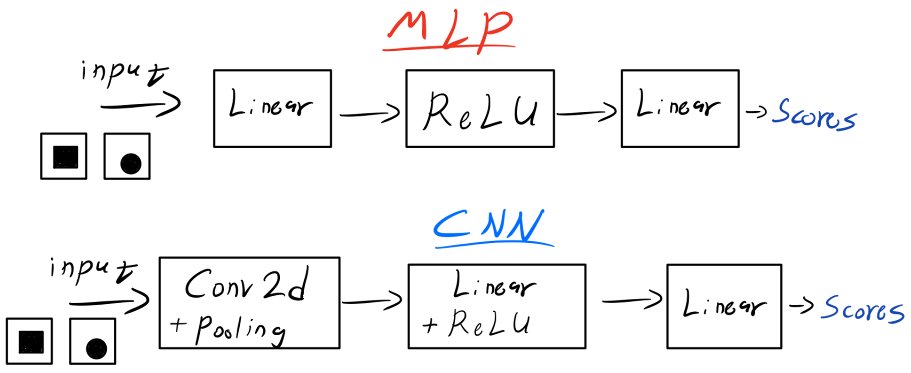

Figure 7: Models sketch

that with the over-parametrized MLP and it didn't lead to a significant improvement. Yet, when we enlarged the amount of data to 16k samples, their performance became closer, but still in the lead of the CNN.

We can assume it happens because when there are fewer samples, the MLP succeeds to fit the training data according to meaningless patterns he finds in the data, but as the number of samples grow, it becomes harder to do that mistake and eventually it considers only meaningful features.

- For unstructured data like spreadsheet, where there is no location invariant features, the CNN might be useless and even do worse than ordinary MLP. In such cases, similar to the conclusion from the **No Free Lunch Theorem**, there is no universal learner and limitations based on prior knowledge are essential to get good generalization.

References

[1] M. Belkina, D. Hsuc, S. Maa, and S. Mandala. Reconciling modern machine-learning practice and the classical bias–variance trade-off, *www.pnas.org/cgi/doi/10.1073/pnas.1903070116*, 2019.

[2] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring Generalization in Deep Learning, *Toyota Technological Institute at Chicago*, 2017.

[3] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[4] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2017.