NEURAL NETWORKS AND DEEP LEARNING CST 395 CS 5TH SEMESTER HONORS COURSE- Dr Binu V P, 9847390760

CS 5th Semester Honors course for the Computer Science at KTU- Dr Binu V P

Capacity, Overfitting and Underfitting

October 01, 2022

The central challenge in machine learning is that we must perform well on new, previously unseen inputs—not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called **generalization.**
Typically, when training a machine learning model, we have access to a training set, we can compute some error measure on the training set called the training error, and we reduce this training error. So far, what we have described is simply an optimization problem. What separates machine learning from optimization is that we want the **generalization error**, also called the **test error**, to be low as well. The generalization error is defined as the expected value of the error on a new input. Here the expectation is taken across different possible inputs, drawn from the distribution of inputs we expect the system to encounter in practice.
We typically estimate the generalization error of a machine learning model by measuring its performance on a test set of examples that were collected separately from the training set.
In our linear regression example, we trained the model by minimizing the training error $\frac{1}{m^{(train)}}||X^{(train)}w - y^{(train)}||_2^2$, but we actually care about the test error $\frac{1}{m^{(test)}}||X^{(test)}w - y^{(test)}||_2^2$

How can we affect performance on the test set when we get to observe only the training set? The field of **statistical learning theory** provides some answers. If the training and the test set are collected arbitrarily, there is indeed little we can do. If we are allowed to make some assumptions about how the training and test set are collected, then we can make some progress.

The train and test data are generated by a probability distribution over datasets called the data generating process. We typically make a set of assumptions known collectively as the **i.i.d. assumptions**. These assumptions are that the examples in each dataset are **independent** from each other, and that the train set and test set are **identically distributed**, drawn from the same probability distribution as each other. This assumption allows us to describe the data generating process with a probability distribution over a

single example. The same distribution is then used to generate every train example and every test example.We call that shared underlying distribution the data generating distribution, denoted pdata . This probabilistic framework and the i.i.d. assumptions allow us to mathematically study the relationship between training error and test error.

One immediate connection we can observe between the training and test error is that the expected training error of a randomly selected model is equal to the expected test error of that model. Suppose we have a probability distribution $p(x, y)$ and we sample from it repeatedly to generate the train set and the test set. For some fixed value w, the expected training set error is exactly the same as the expected test set error, because both expectations are formed using the same dataset sampling process. The only difference between the two conditions is the name we assign to the dataset we sample.

Of course, when we use a machine learning algorithm, we do not fix the parameters ahead of time, then sample both datasets. We sample the training set, then use it to choose the parameters to reduce training set error, then sample the test set. Under this process, the expected test error is greater than or equal to the expected value of training error. The factors determining how well a machine learning algorithm will perform are its ability to:
1. Make the training error small.
2. Make the gap between training and test error small.
These two factors correspond to the two central challenges in machine learning:
underfitting and overfitting . Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set. Overfitting occurs when the gap between the training error and test error is too large.
We can control whether a model is more likely to overfit or underfit by altering its capacity. Informally, a model's capacity is its ability to fit a wide variety of functions. Models with low capacity may struggle to fit the training set. Models with high capacity can overfit by memorizing properties of the training set that do not serve them well on the test set.

One way to control the capacity of a learning algorithm is by choosing its hypothesis space, the set of functions that the learning algorithm is allowed to select as being the solution. For example, the linear regression algorithm has the set of all linear functions of its input as its hypothesis space. We can generalize linear regression to include polynomials, rather than just linear functions, in its hypothesis space. Doing so increases the model's capacity.
A polynomial of degree one gives us the linear regression model with which we are already familiar, with prediction
$\hat{y} = wx + b$

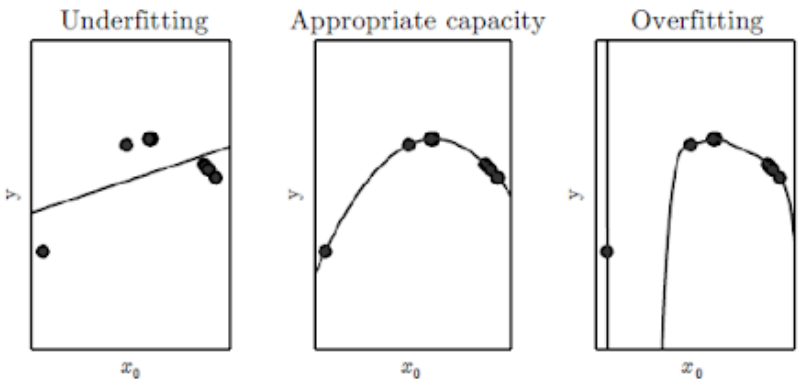
By introducing x^2 as another feature provided to the linear regression model, we can

learn a model that is quadratic as a function of x:
Though this model implements a quadratic function of its input, the output is still a linear function of the parameters, so we can still use the normal equations to train the model in closed form. We can continue to add more powers of x as additional features, for example to obtain a polynomial of degree 9:
$\hat{y} = \sum_{I=1}^{9} w_i x^i + b$

Machine learning algorithms will generally perform best when their capacity is appropriate for the true complexity of the task they need to perform and the amount of training data they are provided with. Models with insufficient capacity are unable to solve complex tasks. Models with high capacity can solve complex tasks, but when their capacity is higher than needed to solve the present task they may overfit.

$\hat{y} = w_1 x + w_2 x^2 + b$

Figure below shows this principle in action. We compare a linear, quadratic and degree-9 predictor attempting to fit a problem where the true underlying function is quadratic. The linear function is unable to capture the curvature in the true underlying problem, so it underfits. The degree-9 predictor is capable of representing the correct function, but it is also capable of representing infinitely many other functions that pass exactly through the training points, because we have more parameters than training examples. We have little chance of choosing a solution that generalizes well when so many wildly different solutions exist. In this example, the quadratic model is perfectly matched to the true structure of the task so it generalizes well to new data.



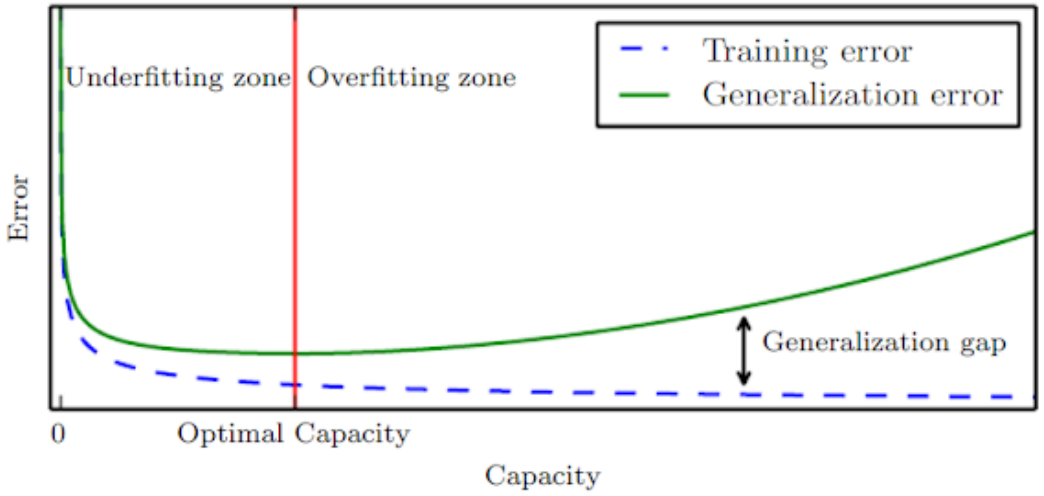
We fit three models to this example training set. The training data was generated synthetically, by randomly sampling x values and choosing y deterministically by evaluating a quadratic function. (Left)A linear function fit to the data suffers from underfitting—it cannot capture the curvature that is present in the data. (Center)A quadratic function fit to the data generalizes well to unseen points. It does not suffer from a significant amount of overfitting or underfitting. (Right)A polynomial of degree 9 fit to the data suffers from overfitting. Here we used the Moore-Penrose pseudoinverse to solve the underdetermined normal equations. The solution passes through all of the

training points exactly, but we have not been lucky enough for it to extract the correct structure. It now has a deep valley in between two training points that does not appear in the true underlying function. It also increases sharply on the left side of the data, while the true function decreases in this area.

So far we have described only one way of changing a model's capacity: by changing the number of input features it has, and simultaneously adding new parameters associated with those features. There are in fact many ways of changing a model's capacity. Capacity is not determined only by the choice of model. The model specifies which family of functions the learning algorithm can choose from when varying the parameters in order to reduce a training objective. This is called the **representational capacity** of the model. In many cases, finding the best function within this family is a very difficult optimization problem. In practice, the learning algorithm does not actually find the best function, but merely one that significantly reduces the training error. These additional limitations, such as the imperfection of the **optimization algorithm**, mean that the learning algorithm's effective capacity may be less than the representational capacity of the model family.

Our modern ideas about improving the generalization of machine learning models are refinements of thought dating back to philosophers at least as early as Ptolemy. Many early scholars invoke a principle of parsimony that is now most widely known as Occam's razor (c. 1287-1347). This principle states that among competing hypotheses that explain known observations equally well, one should choose the "simplest" one. This idea was formalized and made more precise in the 20th century by the founders of statistical learning theory.

We must remember that while simpler functions are more likely to generalize (to have a small gap between training and test error) we must still choose a sufficiently complex hypothesis to achieve low training error. Typically, training error decreases until it asymptotes to the minimum possible error value as model capacity increases (assuming the error measure has a minimum value). Typically, generalization error has a U-shaped curve as a function of model capacity. This is illustrated in figure below.



Typical relationship between capacity and error. Training and test error behave differently. At the left end of the graph, training error and generalization error are both high. This is the **underfitting** regime. As we increase capacity, training error decreases, but the gap between training and generalization error increases. Eventually, the size of this gap outweighs the decrease in training error, and we enter the **overfitting** regime, where capacity is too large, above the **optimal capacity**.

To reach the most extreme case of arbitrarily high capacity, we introduce the concept of non-parametric models. So far, we have seen only parametric models, such as linear regression. Parametric models learn a function described by a parameter vector whose size is finite and fixed before any data is observed.Non-parametric models have no such limitation.

Sometimes, non-parametric models are just theoretical abstractions (such as an algorithm that searches over all possible probability distributions) that cannot be implemented in practice. However, we can also design practical non-parametric models by making their complexity a function of the training set size. One example of such an algorithm is **nearest neighbor regression.** Unlike linear regression, which has a fixed-length vector of weights, the nearest neighbor regression model simply stores the X and y from the training set. When asked to classify a test point x , the model looks up the nearest entry in the training set and returns the associated regression target. In other words, $\hat{y} = y_i$ where $i = argmin||X_i, : -x||_2^2$.The algorithm can also be generalized to distance metrics other than the $L2$ norm, such as learned distance metrics (Goldberger et al., 2005). If the algorithm is allowed to break ties by averaging the y_i values for all $X_i, :$ that are tied for nearest,then this algorithm is able to achieve the minimum possible training error (which might be greater than zero, if two identical inputs are associated with different outputs) on any regression dataset.

Finally, we can also create a non-parametric learning algorithm by wrapping a

parametric learning algorithm inside another algorithm that increases the number of parameters as needed. For example, we could imagine an outer loop of learning that changes the degree of the polynomial learned by linear regression on top of a polynomial expansion of the input.

The ideal model is an oracle that simply knows the true probability distribution that generates the data. Even such a model will still incur some error on many problems, because there may still be some noise in the distribution. In the case of supervised learning, the mapping from x to y may be inherently stochastic, or y may be a deterministic function that involves other variables besides those included in x. The error incurred by an oracle making predictions from the true distribution p(x, y) is called the **Bayes error.**

Training and generalization error vary as the size of the training set varies.Expected generalization error can never increase as the number of training examples increases. For non-parametric models, more data yields better generalization until the best possible error is achieved. Any fixed parametric model with less than optimal capacity will asymptote to an error value that exceeds the Bayes error. See figure below for an illustration. Note that it is possible for the model to have optimal capacity and yet still have a large gap between training and generalization error.In this situation, we may be able to reduce this gap by gathering more training examples.

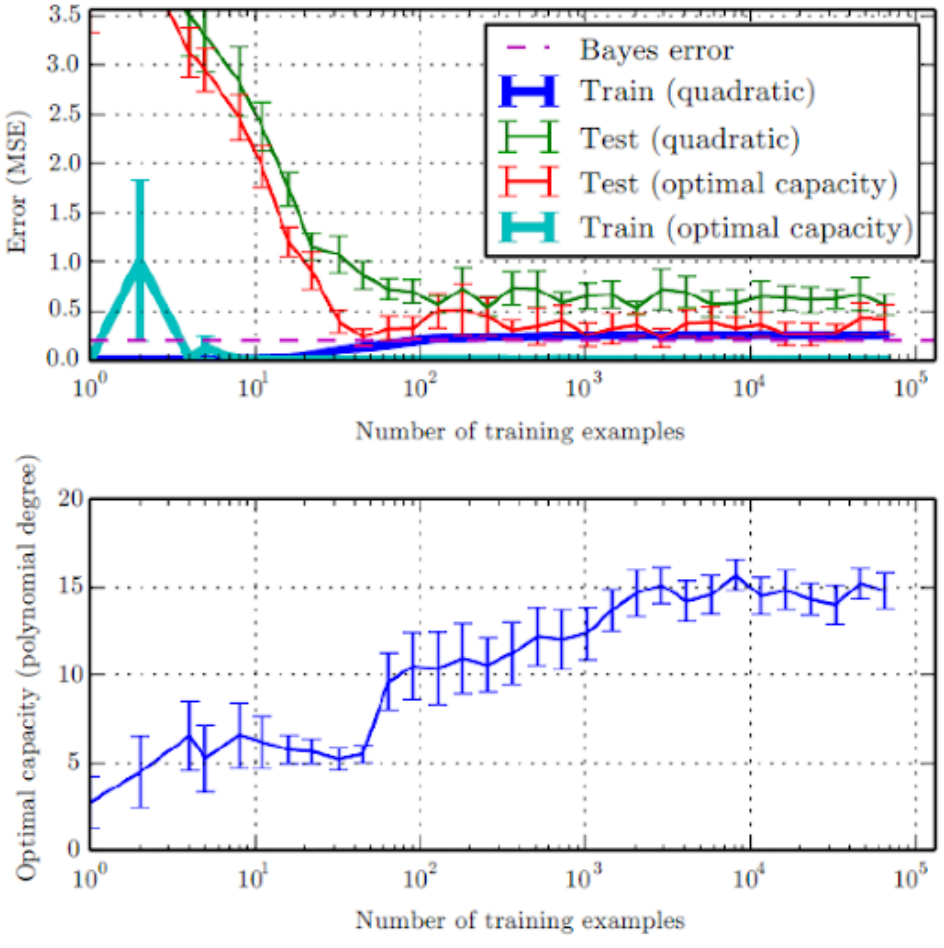


Fig shows the effect of the training dataset size on the train and test error, as well as on

the optimal model capacity. We constructed a synthetic regression problem based on adding a moderate amount of noise to a degree-5 polynomial, generated a single test set, and then generated several different sizes of training set. For each size, we generated 40 different training sets in order to plot error bars showing 95 percent confidence intervals. (Top)The MSE on the training and test set for two different models: a quadratic model, and a model with degree chosen to minimize the test error. Both are fit in closed form. For the quadratic model, the training error increases as the size of the training set increases. This is because larger datasets are harder to fit. Simultaneously, the test error decreases, because fewer incorrect hypotheses are consistent with the training data. The quadratic model does not have enough capacity to solve the task, so its test error asymptotes to a high value. The test error at optimal capacity asymptotes to the Bayes error. The training error can fall below the Bayes error, due to the ability of the training algorithm to memorize specific instances of the training set. As the training size increases to infinity, the training error of any fixed-capacity model (here, the quadratic model) must rise to at least the Bayes error. As the training set size increases, (Bottom) the optimal capacity (shown here as the degree of the optimal polynomial regressor) increases. The optimal capacity plateaus after reaching sufficient complexity to solve the task.

The No Free Lunch Theorem

Learning theory claims that a machine learning algorithm can generalize well from a finite training set of examples. This seems to contradict some basic principles of logic. Inductive reasoning, or inferring general rules from a limited set of examples, is not logically valid.

To logically infer a rule describing every member of a set, one must have information about every member of that set. In part, machine learning avoids this problem by offering only probabilistic rules, rather than the entirely certain rules used in purely logical reasoning. Machine learning promises to find rules that are probably correct about most members of the set they concern.

Unfortunately, even this does not resolve the entire problem. The no free lunch theorem for machine learning (Wolpert, 1996) states that, averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points. In other words, in some sense, no machine learning algorithm is universally any better than any other. The most sophisticated algorithm we can conceive of has the same average performance (over all possible tasks) as merely predicting that every point belongs to the same class.

Fortunately, these results hold only when we average over all possible data generating distributions. If we make assumptions about the kinds of probability distributions we encounter in real-world applications, then we can design learning algorithms that

perform well on these distributions.This means that the goal of machine learning research is not to seek a universal learning algorithm or the absolute best learning algorithm. Instead, our goal is to understand what kinds of distributions are relevant to the "real world" that an AI agent experiences, and what kinds of machine learning algorithms perform well on data drawn from the kinds of data generating distributions we care about.

Popular posts from this blog

NEURAL NETWORKS AND DEEP LEARNING CST 395 CS 5TH SEMESTER HONORS COURSE NOTES - Dr Binu V P, 9847390760

October 03, 2022

About Me Syllabus Question Paper Dec 2022 Module 1 (Basics of Machine Learning) Overview of Machine Learning Machine Learning Algorithm Linear Regression Capacity, Overfitting and Underfitting Regularization Hyperparameters and Validatio ...

READ MORE

Machine Learning Algorithm

October 01, 2022

A machine learning algorithm is an algorithm that is able to learn from data. But what do we mean by learning? Mitchell (1997) provides the definition "A computer program is said to learn from experience E with respect to some class of tasks T and ...

Powered by Blogger

READ MORE

Theme images by Michael Elkan

Syllabus

October 01, 2022

Syllabus Module - 1 (Basics of Machine Learning) Machine Learning basics - Learning algorithms - Supervised, Unsupervised, Reinforcement, overfitting, Underfitting, Hyper parameters and Validation sets, Estimators -Bias and Variance. Challenges ...

READ MORE

DR.BINU V P-9847390760

Associate Professor and Head Dept of Computer Science-IHRD Karunagappally.Love to share the thoughts and views .I play lot of Games and basically an Athlet in my school and college days.I never keep my studies as a second option.Stood first In MTech and BTech.I did my resea ...

VISIT PROFILE

Archive

Report Abuse