Regularization for Deep Learning

The Problem

So far, we have focused on minimizing the objective function using a variety of optimization algorithms.
Deep learning models typically have BILLIONS of parameters whereas the training data may have only MILLIONS of samples.
Therefore, they are called over-parameterized models.
Over-parameterized models are **prone** to a phenomenon called over-fitting.
To understand this, let's start with the bias and variance of a model with respect to its capacity

Overfitting

One of the most important aspects when training neural networks is avoiding overfitting. We have addressed the issue of overfitting in more detail in this article.

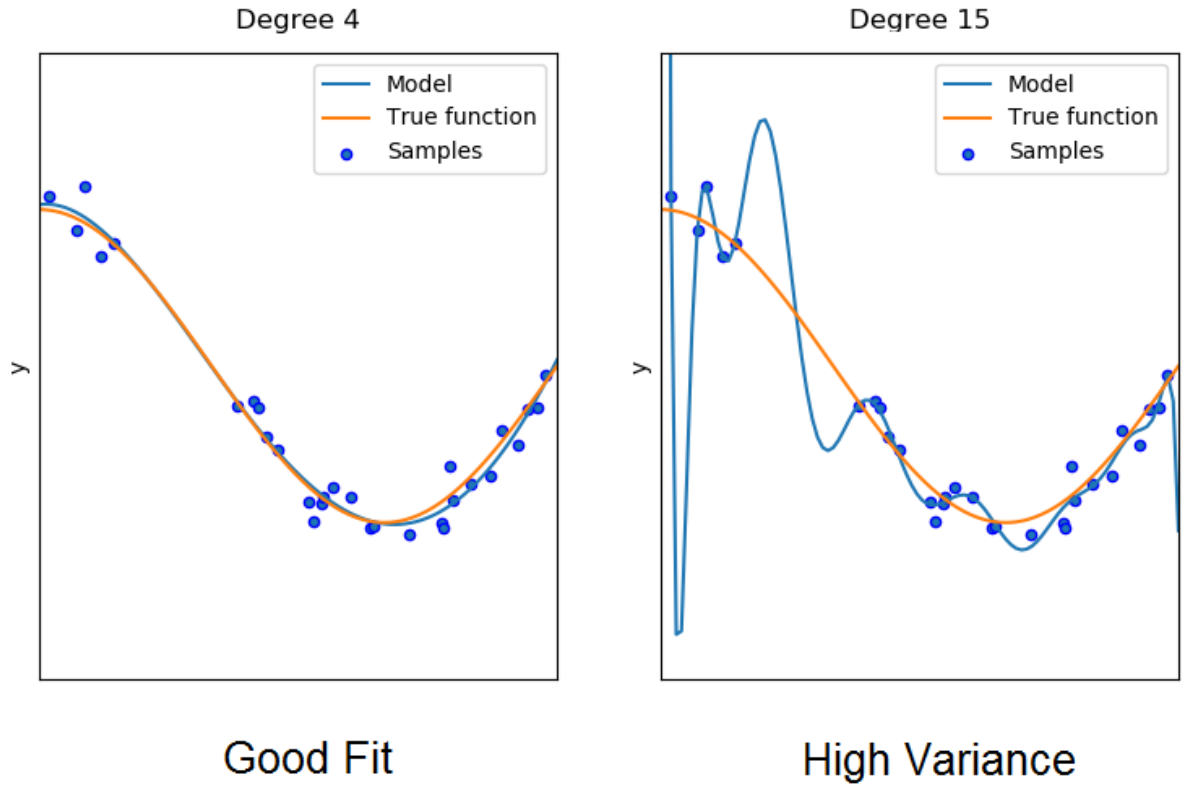
However, let us do a quick recap: Overfitting refers to the phenomenon where a neural network models the training data very well but fails when it sees new data from the same problem domain. Overfitting is caused by noise in the training data that the neural network picks up during training and learns it as an underlying concept of the data.

This learned noise, however, is unique to each training set. As soon as the model sees new data from the same problem domain, but that does not contain this noise, the performance of the neural network gets much worse.

“Why does the neural network picks up that noise in the first place?”

The reason for this is that the complexity of this network is too high. A fit of a neural network

with higher complexity is shown in the image on the right-hand side.



Graph 1. Model with a good fit and high variance.
Source: https://www.researchgate.net/publication/332412613

The model with a higher complexity can pick up and learn patterns (noise) in the data that are just caused by some random fluctuation or error. The network would be able to model each data sample of the distribution one-by-one, while not recognizing the true function that describes the distribution.

New arbitrary samples generated with the true function would have a high distance to the fit of the model. We also say that the model has a high variance.

On the other hand, the lower complexity network on the left side models the distribution much better by not trying too hard to model each data pattern individually.

In practice, overfitting causes the neural network model to perform very well during training, but the performance gets much worse during inference time when faced with brand new data.

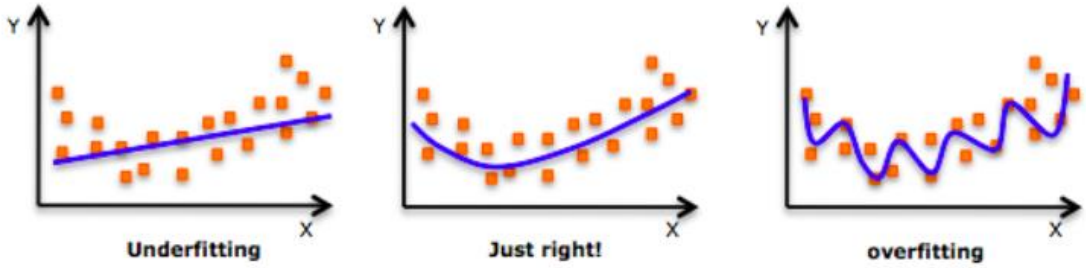
In short: Less complex neural networks are less susceptible to overfitting. To prevent overfitting or a high variance we must use something that is called regularization.

What is Regularization?

Regularization is a technique used in machine learning and deep learning to prevent overfitting and improve the generalization performance of a model. It involves adding a penalty term to the loss function during training.

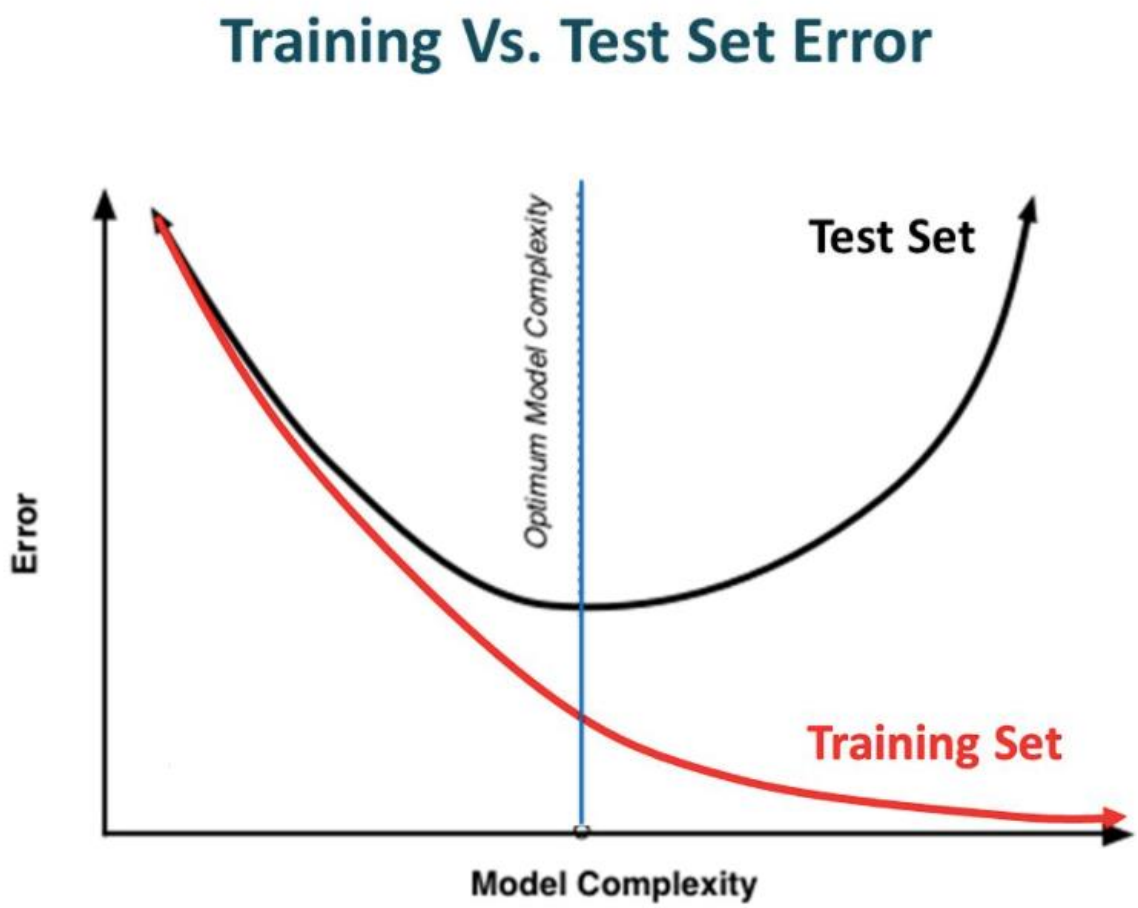
This penalty discourages the model from becoming too complex or having large parameter values, which helps in controlling the model's ability to fit noise in the training data. Regularization methods include L1 and L2 regularization, dropout, early stopping, and more. By applying regularization, models become more robust and better at making accurate predictions on unseen data.

Before we deep dive into the topic, take a look at this image:



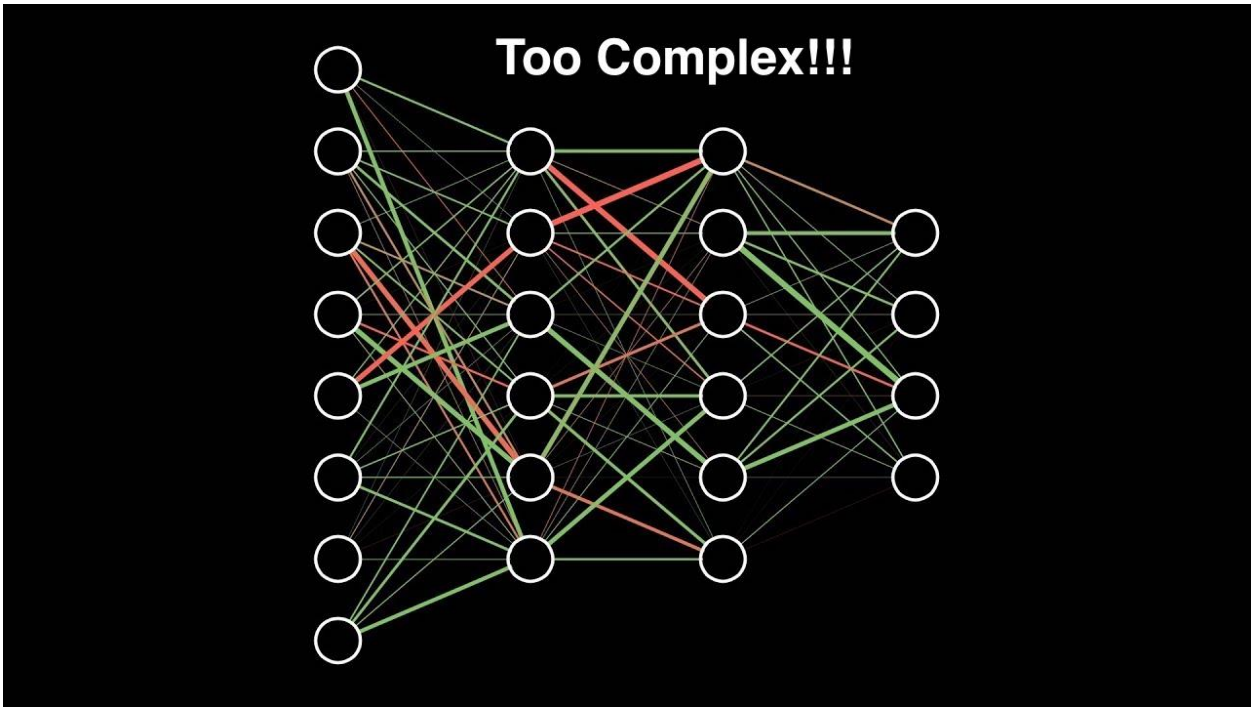
Have you seen this image before? As we move towards the right in this image, our model tries to learn too well the details and the noise from the training data, which ultimately results in poor performance on the unseen data.

In other words, while going towards the right, the complexity of the model increases such that the training error reduces but the testing error doesn't. This is shown in the image below.



Source: Slideplayer

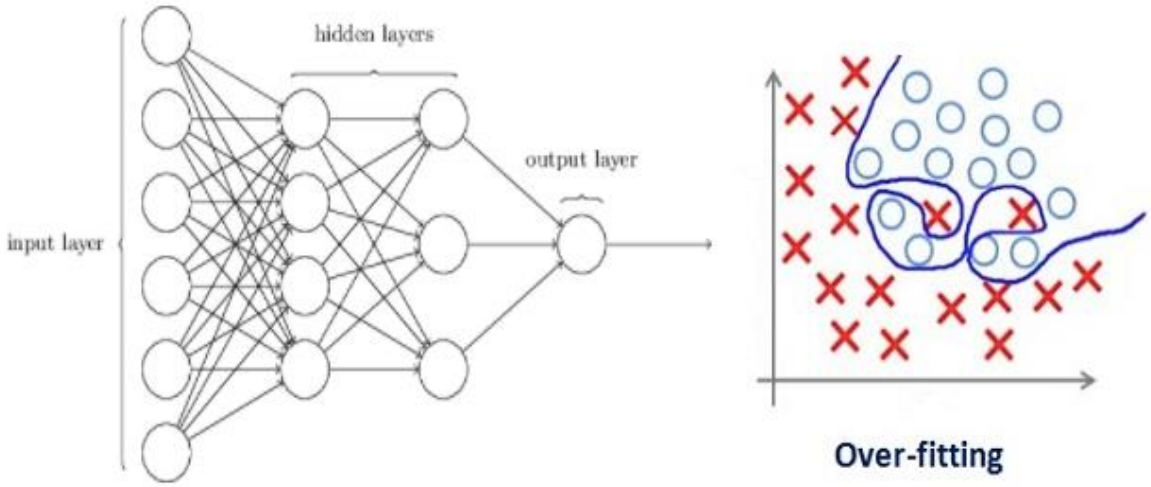
If you've built a neural network before, you know how complex they are. This makes them more prone to overfitting.



Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better. This in turn improves the model's performance on the unseen data as well.

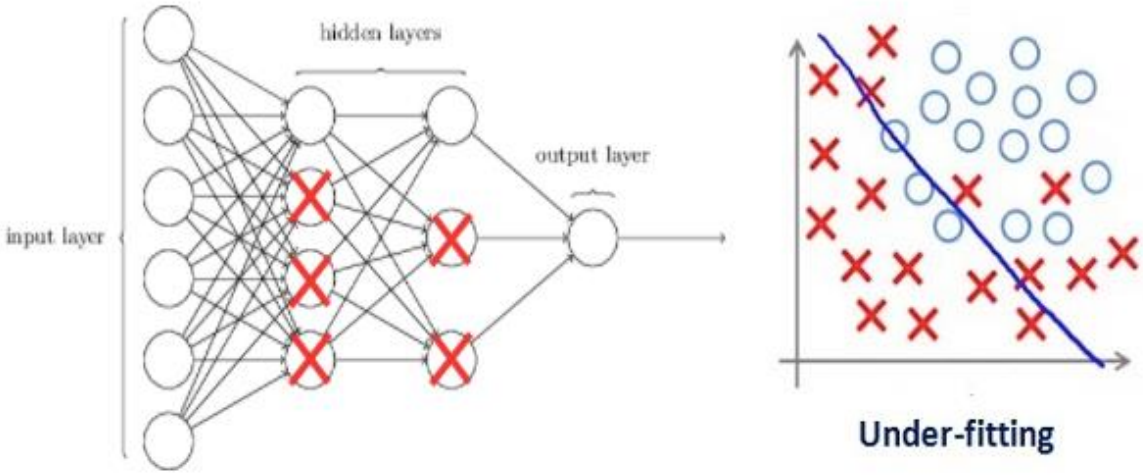
How does Regularization help reduce Overfitting?

Let's consider a neural network which is overfitting on the training data as shown in the image below.



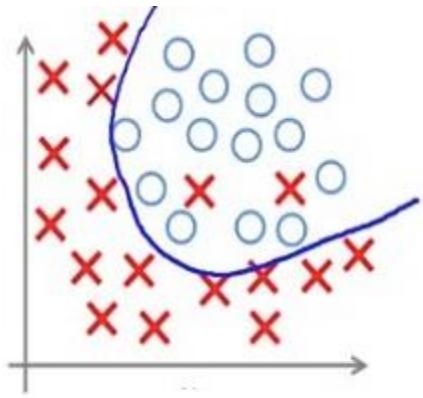
If you have studied the concept of regularization **in machine learning, you will have a fair idea that regularization penalizes the coefficients. In deep learning, it actually penalizes the weight matrices of the nodes.**

Assume that our regularization coefficient is so high that some of the weight matrices are nearly equal to zero.



This will result in a much simpler linear network and slight underfitting of the training data.

Such a large value of the regularization coefficient is not that useful. We need to optimize the value of regularization coefficient in order to obtain a well-fitted model as shown in the image below.



Appropriate-fitting

Different Regularization Techniques in Deep Learning

Now that we have an understanding of how regularization helps in reducing overfitting, we'll learn a few different techniques in order to apply regularization in deep learning.

L2 & L1 regularization

L1 and L2 are the most common types of regularization. These update the general cost function by adding another term known as the regularization term.

- *Cost function = Loss (say, binary cross entropy) + Regularization term*

Due to the addition of this regularization term, the values of weight matrices decrease because it assumes that a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent.

However, this regularization term differs in L1 and L2.

In L2, we have:

$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|^2$$

Here, **lambda** is the regularization parameter. It is the hyperparameter whose value is optimized for better results. L2 regularization is also known as *weight decay* as it forces the weights to decay towards zero (but not exactly zero).

In L1, we have:

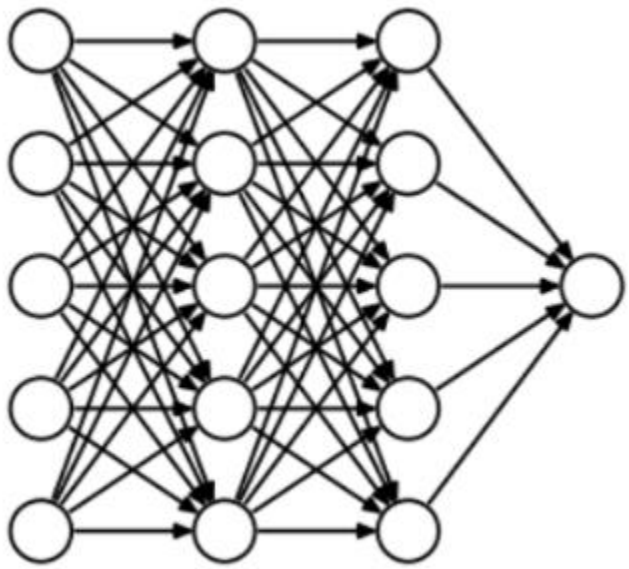
$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|$$

In this, we penalize the absolute value of the weights. Unlike L2, the weights may be reduced to zero here. **Hence, it is very useful when we are trying to compress our model. Otherwise, we usually prefer L2 over it.**

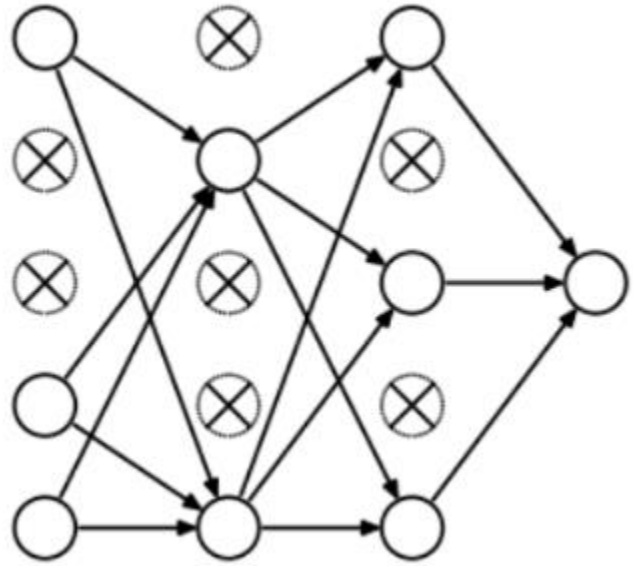
Dropout

This is the one of the most interesting types of regularization techniques. It also produces very good results and is consequently the most frequently used regularization technique in the field of deep learning.

To understand dropout, let's say our neural network structure is akin to the one shown below:



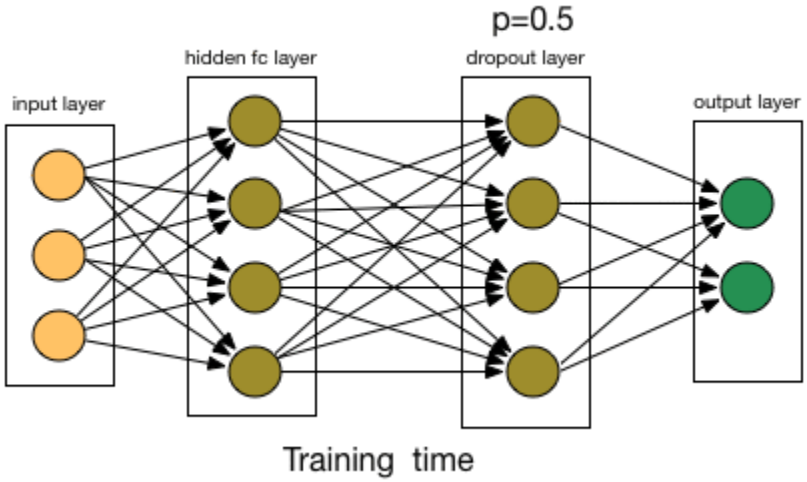
So what does dropout do? At every iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connections as shown below.



So each iteration has a different set of nodes and this results in a different set of outputs. **It can also be thought of as an ensemble technique in machine learning.**

Ensemble models usually perform better than a single model as they capture more randomness. Similarly, dropout also performs better than a normal neural network model.

This probability of choosing how many nodes should be dropped is the hyperparameter of the dropout function. As seen in the image above, dropout can be applied to both the hidden layers as well as the input layers.



Source: chatbotslife

Due to these reasons, dropout is usually preferred when we have a large neural network structure in order to introduce more randomness.

Data Augmentation

The simplest way to reduce overfitting is to increase the size of the training data. In machine learning, we were not able to increase the size of training data as the labeled data was too costly.

But, now let's consider we are dealing with images. In this case, there are a few ways of increasing the size of the training data – rotating the image, flipping, scaling, shifting, etc. In the below image, some transformation has been done on the handwritten digits dataset.

This technique is known as data augmentation. This usually provides a big leap in improving the accuracy of the model. *It can be considered as a mandatory trick in order to improve our predictions.*

Early stopping

Early stopping is a kind of cross-validation strategy where we keep one part of the training set as the validation set. When we see that the performance on the validation set is getting worse, we immediately stop the training on the model. This is known as early stopping.

In the above image, we will stop training at the dotted line since after that our model will start overfitting on the training data.