# OVERFITTING

## INTRODUCTION:

### GENERALIZATION ABILITY:

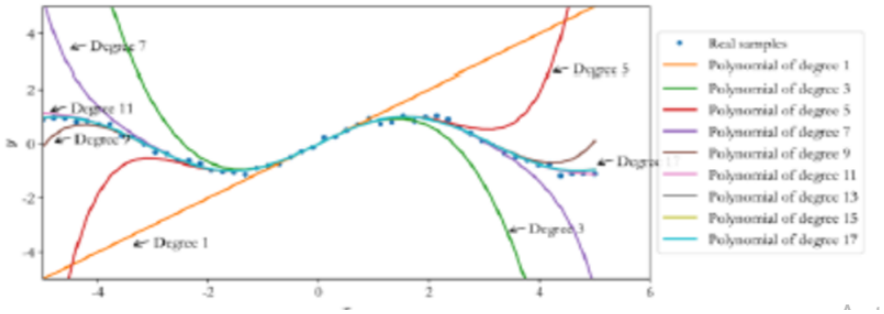
- The ability of machine learning to learn the real model of the data from the training set, so that it can perform well on the unseen test set.

### CAPACITY OF THE MODEL:

- The expressive power of the model.
- When the model's expressive power is weak, such as a single linear layer, it can only learn a linear model and not perform well on nonlinear model.
- When the model's expressive power is too strong, it may be possible to reduce the noise modalities of the training set, but leads to poor performance on the test set (generalization ability is weak).
- Thus, the model's ability to fit complex functions is called Model capacity.
- Its major indicator is the size of its hypothesis space.
- Consider the following examples , to understand the concept of model capacity in a better way:

**EXAMPLE:**

- $P_{data} = \{(x, y)|y = \sin(x)\ ,\ x \in [-5,5]\}$
- A small number of points are sampled from the real distribution to form the training set, which contains the observation error $\epsilon$, as shown by the small dots in Figure.



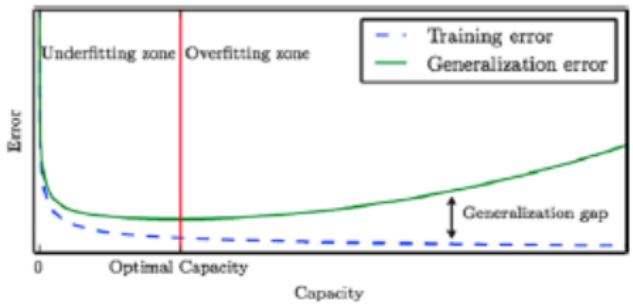

- Initially, If we only search the model space of all first-degree polynomials and set the bias to 0, that is, $y = ax$, as shown by the straight line of the first-degree polynomial.
- After increasing the hypothesis space again, as shown in the polynomial curves of 7, 9, 11, 13, 15, and 17 in Figure, the larger the hypothesis space of the function, the more likely it is to find a function model that better approximates the real distribution.

**CONS OF USING EXCESSIVELY LARGE HYPOTHESIS SPACE:**

- will undoubtedly increase the search difficulty
- Increase in computational cost.
- Doesn't guarantee a better model
- Presence of Observation errors in training hurts the generalization ability of the model.

**OVERFITTING AND UNDERFITTING:**

- Because the distribution of real data is often unknown and complicated, it is impossible to deduce the type of distribution function and related parameters.
- Therefore, when choosing the capacity of the learning model, people often choose a slightly larger model capacity based on empirical values.
- When the capacity of the model is too large, it may appear to perform better on the training set, but perform worse on the test set.
- When the capacity of the model is too small, it may have poor performance in both the training set and the testing set as shown in the area to the left of the red vertical line in Figure.




REASON FOR OVER FITTING:

- When the capacity of the model is too large, in addition to learning the modalities of the training set data, the network model also learns additional observation errors, resulting in the learned model performing better on the training set, but poor in unseen samples.
- Thus, the generalization ability of the model is weak.
- We call this phenomenon as overfitting

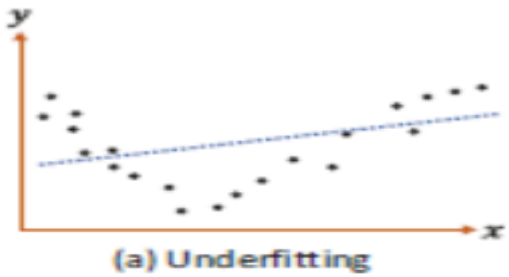
REASON FOR UNDERFITTING:

- When the capacity of the model is too small, the model cannot learn the modalities of the training set data well, resulting in poor performance on both the training set and the unseen samples.

- We call this phenomenon as under fitting.

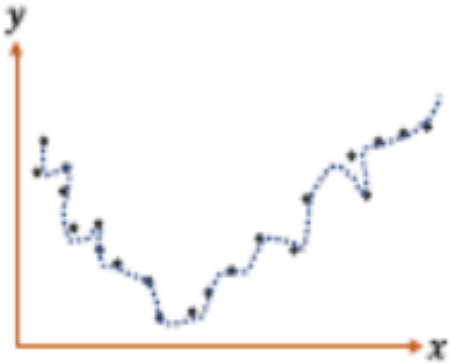
EXAMPLE:

- Consider a degree 2 polynomial data distribution.
- If we use a simple linear function to learn, we will find it difficult to learn a better function, resulting in the underfitting phenomenon that the training set and the test set do not perform well, as shown in Figure.



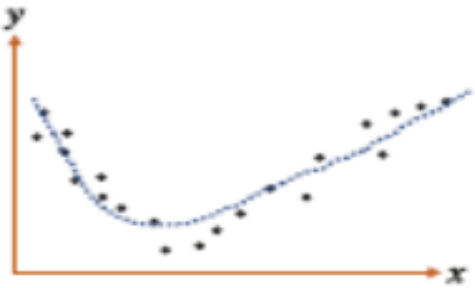
(a) Underfitting

- If you use a more complex function model to learn, it is possible that the learned function will excessively “fit” the training set samples, but resulting in poor performance on the test set, that is, overfitting, as shown in Figure



(c) Overfitting

- Only when the capacity of the learned model and the real model roughly match, the model can have a good generalization ability, as shown in Figure



(b) Matching capacity

SOLUTION TO UNDERFITTING:

- The problem of under fitting can be solved by increasing the number of layers of the neural network.
- It can also be solved by increasing the size of the intermediate dimension.
- However, because modern deep neural network models can easily reach deeper layers, the capacity of the model used for learning is generally sufficient.
- In real applications, more overfitting phenomena occur.

SOLUTION TO OVERFITTING:

1. Data set Division
2. Model Design
3. Regularization
4. Drop Out
5. Data Augmentation

DATASET DIVISION:

- Earlier we used to divide the data set into a training set and a test set.
- In order to select model hyper parameters and detect over fitting, it is generally necessary to split the original training set into three subsets:

  **Training set, validation set, and test set.**
- We know that training set *Dtrain* is used to train model parameters,
- The test set *Dtest* is used to test the generalization ability of the model.
- Example, Training set = 80% of MNIST dataset and Test set = 20% of MNIST data set.

*Training and testing dataset division*

- the performance of the test set cannot be used as feedback for model training.
- we need to be able to pick out more suitable model hyperparameters during model training to determine whether the model is overfitting.
- Therefore, we need to divide the training set into training set and validation set.

*. Training, validation, and test dataset*

- The divided training set has the same function as the original training set and is used to train the parameters of the model, **while the validation set is used to select the hyperparameters of the model.**

**FUNCTIONS OF VALIDATION DATASET:**

- Adjust the learning rate, weight decay coefficient, training times, etc. according to the performance of the validation set.
- Readjust the network topology according to the performance of the validation set.
- According to the performance of the validation set, determine whether it is overfitting or underfitting.
- the training set, validation set, and test set can be divided according to a custom ratio, such as the common 60%-20%-20% division.

**DIFFERENCE BETWEEN TEST & VALIDATION SETS:**

- The algorithm designer can adjust the settings of various hyperparameters of the model according to the performance of the validation set to improve the generalization ability of the model.
- But the performance of the test set cannot be used to adjust the model.

**EARLY STOPPING:**

**EPOCH:**

- one batch updating in the training set one Step, and iterating through all the samples in the training set once is called an Epoch.
- It is generally recommended to perform a validation operation after several Epochs else it introduces additional computation costs.
- If the training error of the model is low and the training accuracy is high, but the validation error is high and the validation accuracy rate is low, overfitting may occur.
- If the errors on both the training set and the validation set are high and the accuracy is low, then underfitting may occur.


*Training process diagram*

EXAMPLE: A TYPICAL CLASSIFICATION

NOTE 1: In the laterstage of training, even with the same network structure, due to the change in the actual capacity of the model, we observed the phenomenon of overfitting

NOTE2:

- This means that for neural networks, even if the network hyperparameters amount remains unchanged (i.e., the maximum capacity of the network is fixed), the model may still appear to be overfitting.
- It is because the effective capacity of the neural network is closely related to the state of the network parameters
- As the number of training Epochs increased, the overfitting became more andmore serious.
- We can observe early stopping epoch as the vertical dotted line is in the best state of the network, there is no obvious overfitting phenomenon, and the generalization ability of the network is the best.

  When it is found that the validation accuracy has not decreased for successive Epochs, we can predict that the most suitable Epoch may have been reached, so we can stop training.

**REGULARIZATION:**

- By designing network models with different layers and sizes, the initial function hypothesis space can be provided for the optimization algorithm, but the actual capacity of the model can change as the network parameters are optimized and updated.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \varepsilon$$

- The capacity of the preceding model can be simply measuredthrough *n*.
- By limiting the sparsity of network parameters, the actual capacity of the network can be constrained.
- This constraint is generally achieved by adding additional parameter sparsity penalties to the loss function.
- Optimization goal before adding constraint:

$$min\, L\left(f_\theta(x), y\right), (x, y) \in D^{train}$$

- Optimisation goal before adding constraint:

$$min\, L\left(f_\theta(x), y\right) + \lambda \cdot \Omega(\theta), (x, y) \in D^{train}$$

- where $\Omega(\vartheta)$ represents the sparsity constraint function on the network parameters $\vartheta$.
- The sparsity constraint of the parameter $\vartheta$ is achieved by constraining the *L* norm of the parameter, that is:

$$\Omega(\theta) = \sum_{\theta_i} \|\theta_i\|_l$$

- The goal of an optimization algorithm is to minimize the original loss function L(x,y) and also to reduce network sparisty $\Omega(\vartheta)$
- Here $\lambda$ is the weight parameter to balance the importance of $L(x, y)$ and $\Omega(\vartheta)$.
- Larger $\lambda$ means that the sparsity of the network is more important; smaller $\lambda$ means that the training error of the network is more important.
- By selecting the appropriate $\lambda$, you can get better training performance, while ensuring the sparsity of the network, which lead to a good generalization ability.
- Commonly used regularization methods are L0, L1, and L2 regularization.

**L0 regularization:**

- L0 regularization refers to the regularization calculation method using the L0 norm as the sparsity penalty term $\Omega(\vartheta)$.

$$\Omega(\theta) = \sum_{\theta_i} \|\theta_i\|_0$$

- The L0 norm $\|\vartheta i\|$o is defined as the number of non-zero elements in $\vartheta i$.
- This constraint can force the connection weights in the network to be mostly 0, thereby reducing the actual amount of network parameters and network capacity.
- DISADVANTAGE: However, because the L0 norm is not derivable, gradient descent algorithm cannot be used for optimization. L0 norm is not often used in neural networks

**L1 Regularization**

- The L1 regularization refers to the regularization calculation method using the L1 norm as the sparsity penalty term $\Omega(\vartheta)$.

$$\Omega(\theta) = \sum_{\theta_i} \|\theta_i\|_1$$

- The L1 norm $\|\vartheta i\|1$ is defined as the sum of the absolute values of all elements in the tensor $\vartheta i$.
- L1 regularization is also called **Lasso** regularization, which is continuously derivable and widely used in neural networks.

- IMPLEMENTATION:

```
# Create weights w1,w2
w1 = tf.random.normal([4,3])
w2 = tf.random.normal([4,2])
# Calculate L1 regularization term
loss_reg = tf.reduce_sum(tf.math.abs(w1))\
    + tf.reduce_sum(tf.math.abs(w2))
```

**L2 regularization:**

- The L2 regularization refers to the regularization calculation method using the L2 norm as the sparsity penalty term $\Omega(\vartheta)$.

$$\Omega(\theta)=\sum_{\theta_i} \|\theta_i\|_2$$

- The L2 norm $\|\vartheta i\|2$ is defined as the sum of squares of the absolute values of all elements in the tensor $\vartheta i$.

- L1 regularization is also called **Ridge** regularization, which is continuously derivable and widely used in neural networks.

- IMPLEMENTATION:

```
# Create weights w1,w2
w1 = tf.random.normal([4,3])
w2 = tf.random.normal([4,2])
# Calculate L2 regularization term
loss_reg = tf.reduce_sum(tf.square(w1))\
     + tf.reduce_sum(tf.square(w2))
```

**What does Regularization achieve?**

- A standard least squares model tends to have some variance in it. Such model won't generalize well for a data set different than its training data.

- ***Regularization, significantly reduces the variance of the model, without substantial increase in its bias***.

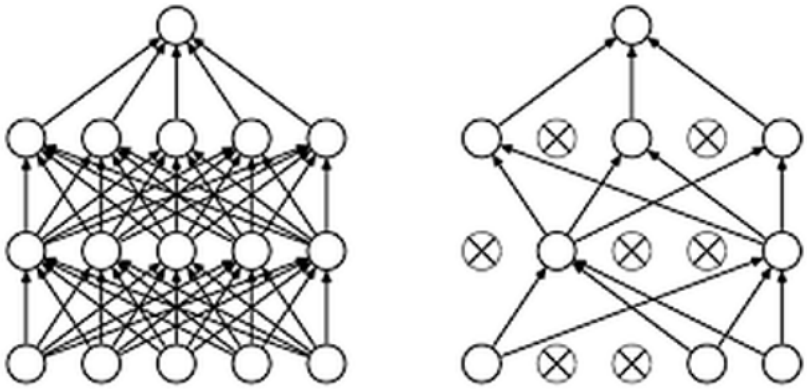
- So the tuning parameter λ, used in the regularization, controls the impact on bias and variance.

- As the value of λ rises, it reduces the value of coefficients and thus reducing the variance.

- ***Till a point, this increase in λ is beneficial as it is only reducing the variance(hence avoiding overfitting), without loosing any important properties in the data.***
- But after certain value, the model starts loosing important properties, giving rise to bias in the model and thus underfitting.
- Therefore, the value of λ should be carefully selected.

**DROPOUT:**

- Dropout works by essentially "dropping" a neuron from the input or hidden layers. Multiple neurons are removed from the network, meaning they practically do not exist — their incoming and outcoming connections are also destroyed.
- This artificially creates a multitude of smaller, less complex networks. This forces the model to not become solely dependent on one neuron, meaning it has to diversify its approach and develop a multitude of methods to achieve the same result.
- Dropout is applied to a neural network by randomly dropping neurons in every layer (including the input layer). A pre-defined dropout rate determines the chance of each neuron being dropped. For example, a dropout rate of 0.25 means that there is a 25% chance of a neuron being dropped. Dropout is applied during every epoch during the training of the model.



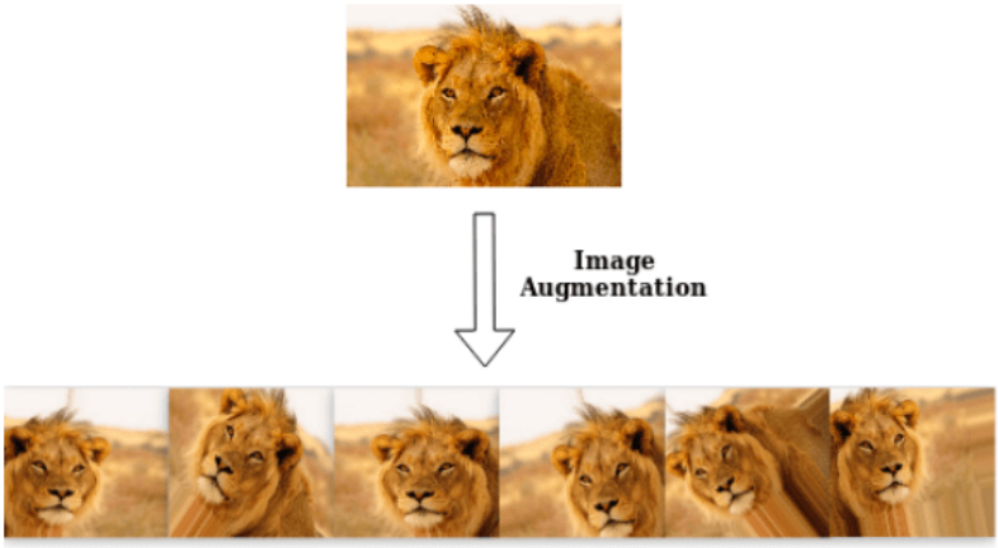
(a) Standard Neural Net (b) After applying dropout.

**DATA AUGMENTATION:**

One of the best techniques for reducing overfitting is to increase the size of the training dataset. As discussed in the previous technique, when the size of the training data is small, then the network tends to have greater control over the training data.

So, to increase the size of the training data i.e, increasing the number of images present in the dataset, we can use data augmentation, which is the easiest way to diversify our data and make the training data larger.

Some of the popular image augmentation techniques are flipping, translation, rotation, scaling,cropping, changing brightness, adding noise etc




Here are some commonly used data augmentation techniques for various types of data:

Image Data Augmentation:
a. Rotation: Rotate images by various degrees to simulate different angles.
b. Flip: Horizontally and/or vertically flip images.
c. Zoom: Randomly zoom in or out of images.
d. Crop: Randomly crop and resize images to different dimensions.
e. Translation: Shift images horizontally and/or vertically.
f. Brightness and Contrast Adjustments: Randomly adjust brightness and contrast.
g. Noise: Add random noise to the images.
h. Color Jitter: Randomly change hue, saturation, and brightness.

Text Data Augmentation:
a. Synonym Replacement: Replace words with their synonyms.
b. Random Deletion: Randomly delete words from the text.
c. Random Swap: Randomly swap the positions of two words in a sentence.
d. Back-Translation: Translate text to another language and then back to the original language.
e. Insertion: Insert random words into the text.

Time Series Data Augmentation:
a. Time Warping: Slightly warp the time series by stretching or compressing it.
b. Jittering: Add small random noise to the data points.
c. Rolling Window: Apply rolling window transformations to create new data points.

Audio Data Augmentation:
a. Pitch Shifting: Change the pitch of the audio.
b. Time Stretching: Stretch or compress the audio in time.
c. Noise Injection: Add background noise to the audio.
d. Speed Perturbation: Vary the playback speed of the audio.

Tabular Data Augmentation:
a. Feature Scaling: Scale features within a certain range.
b. Feature Selection: Randomly select a subset of features.
c. Data Perturbation: Add random noise to the data.

Synthetic Data Generation:

a. Generative Adversarial Networks (GANs): Generate synthetic data samples that mimic the real data distribution.

b. Variational Autoencoders (VAEs): Create new data points from the latent space of the VAE.

When applying data augmentation, it's essential to strike a balance. Too much augmentation can lead to model training on noisy or unrealistic data, while too little may not effectively combat overfitting. Experiment with different augmentation techniques and parameters to find the right balance for your specific problem and dataset. Cross-validation can help in evaluating the impact of data augmentation on your model's performance.