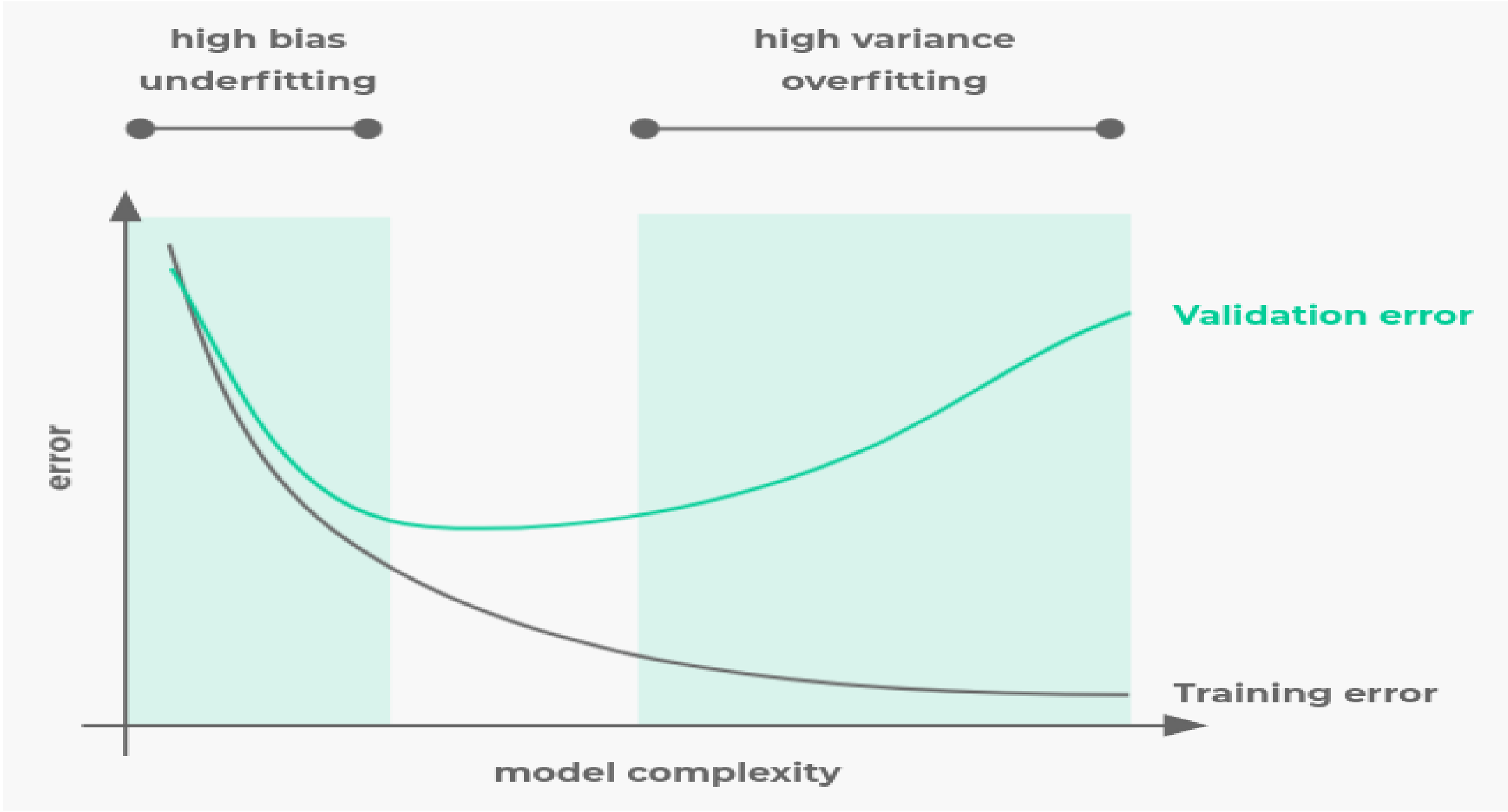
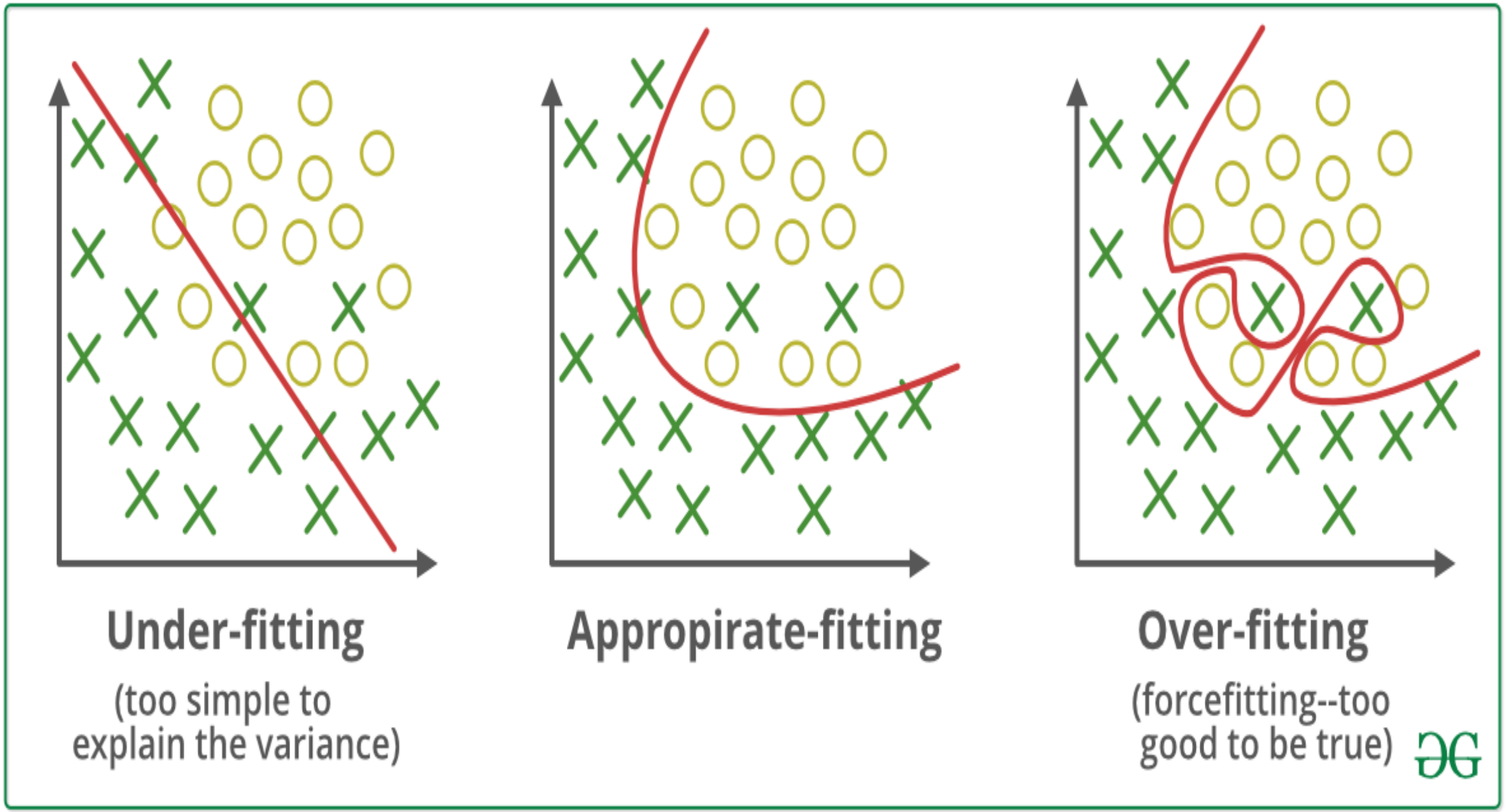
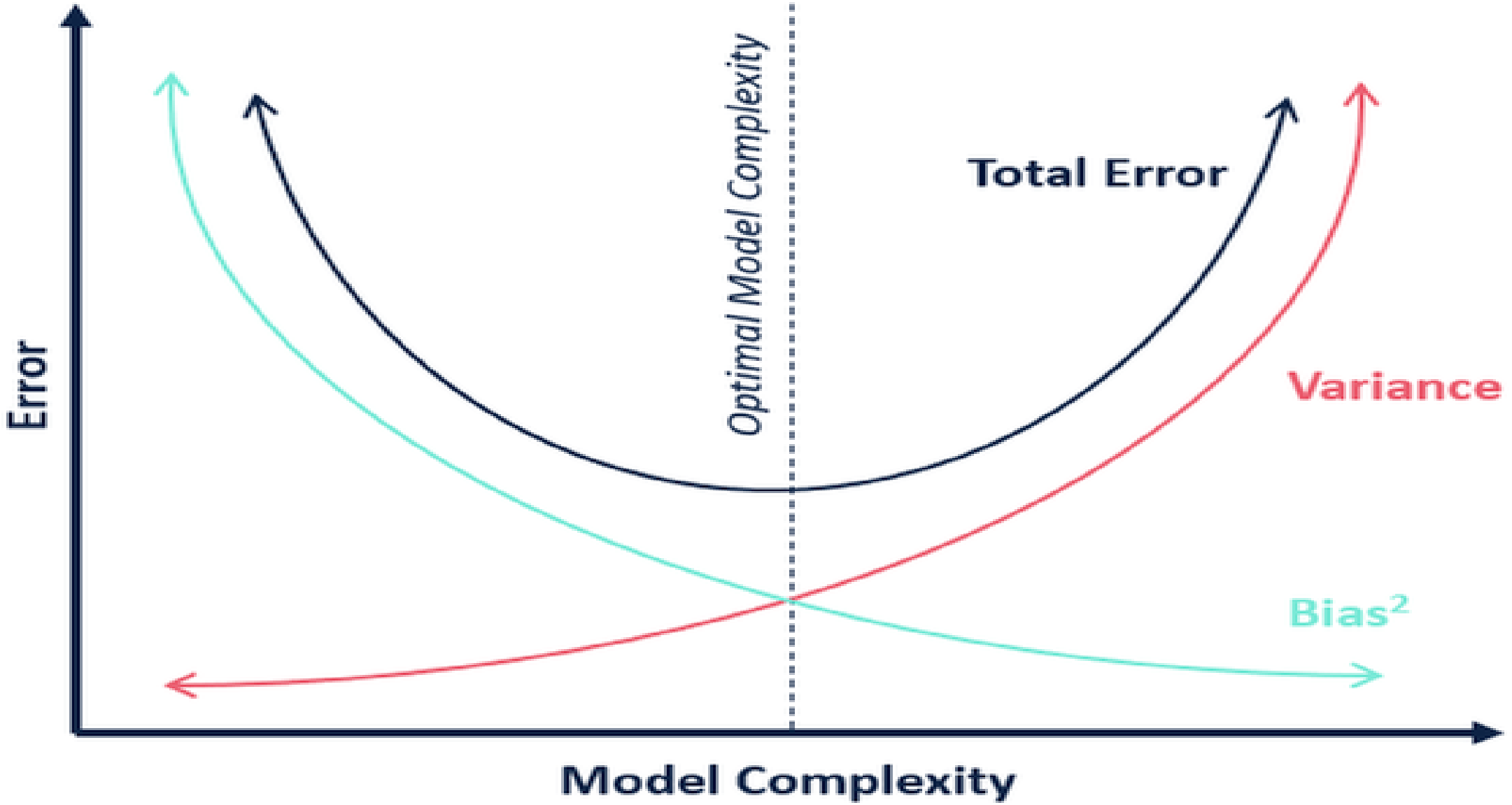
Dropout in Neural network


high bias
underfitting
high variance
overfitting
error
Validation error
Training error
model complexity

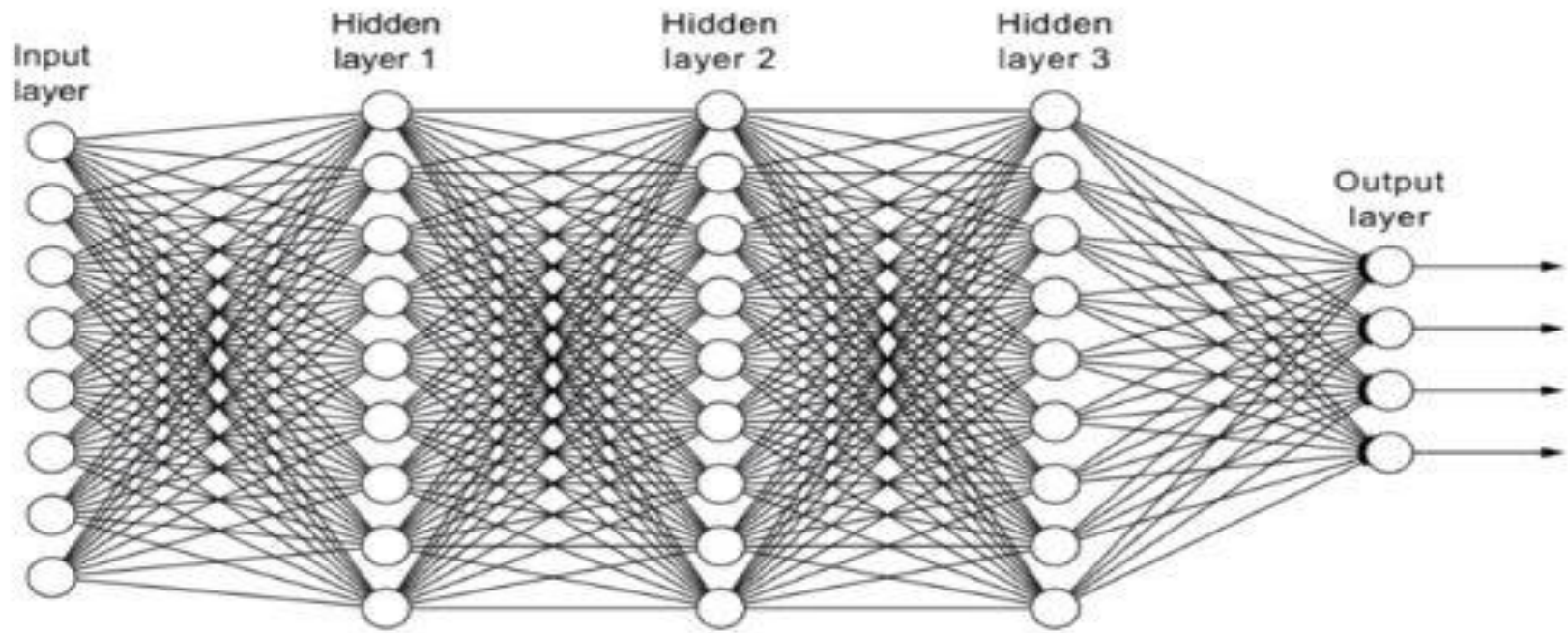

Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

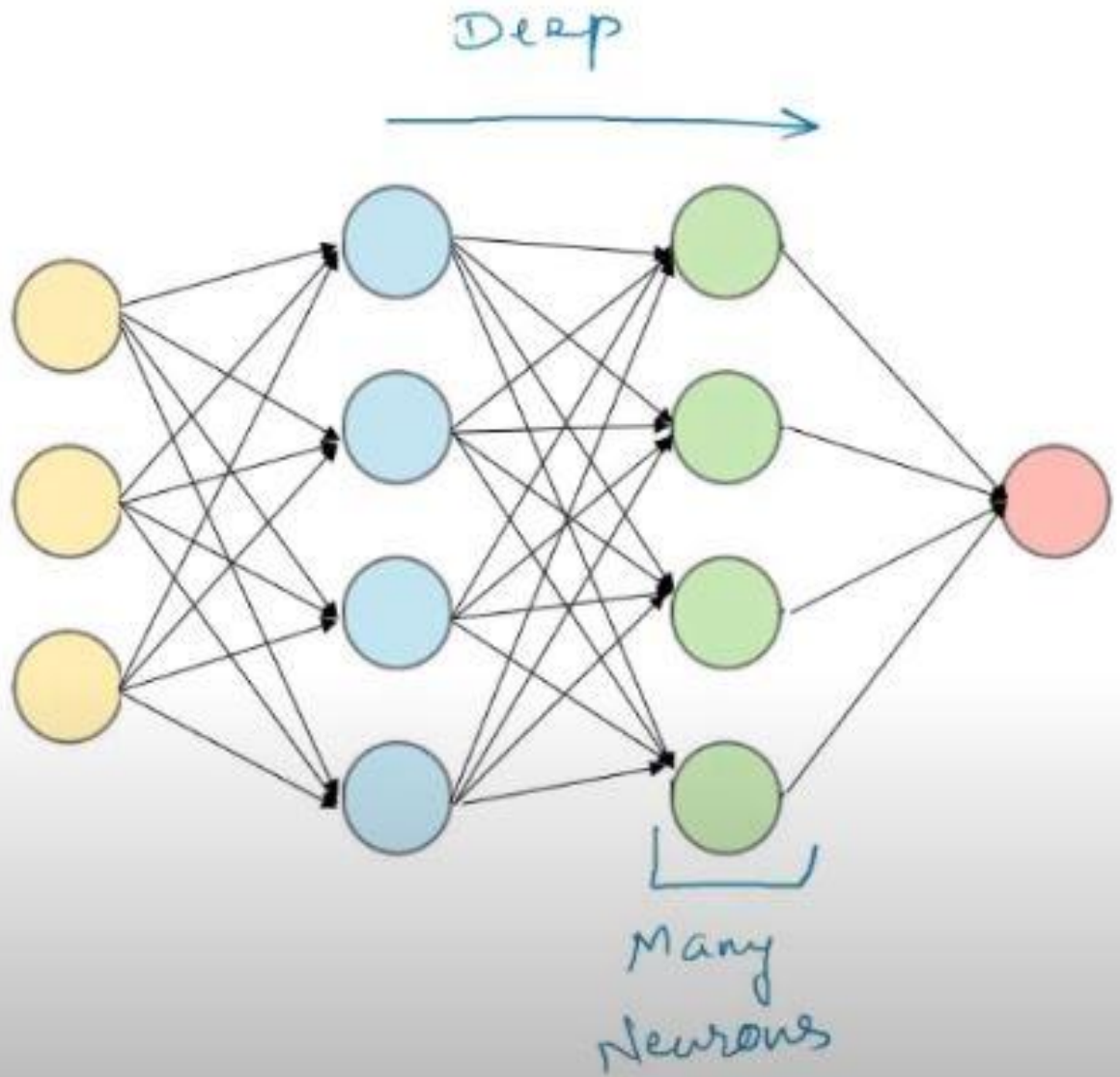
- low bias refers to a model that is able to capture the underlying patterns and relationships in the data accurately. It means that the model is flexible enough to fit the training data well.
- On the other hand, high variance refers to a model that is overly complex and sensitive to the training data. It means that the model may fit the training data too closely, resulting in poor generalization to new, unseen data.
- In simpler terms, low bias implies that the model is not making many assumptions about the data, allowing it to learn complex patterns. High variance, however, suggests that the model is overfitting the training data, resulting in poor performance on new data.
- Finding the right balance between low bias and high variance is crucial in machine learning, as both extremes can lead to poor model performance.

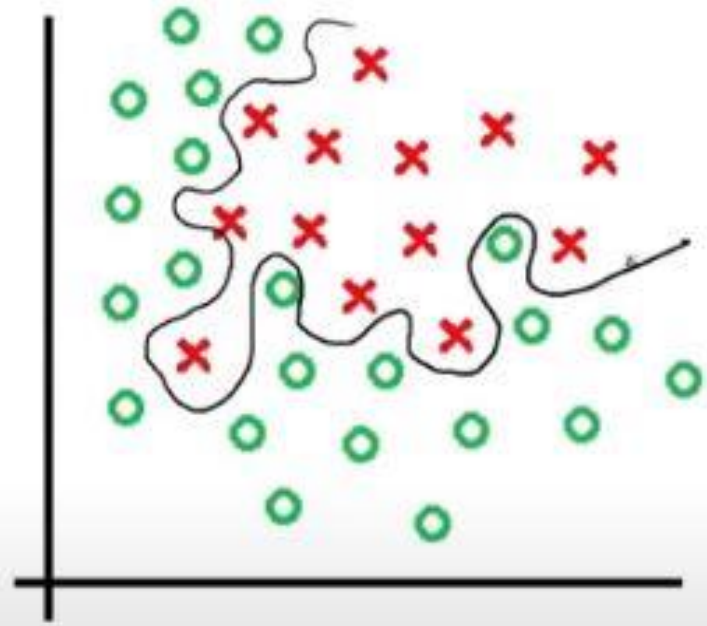

Error
Optimal Model Complexity
Total Error
Variance
Bias²
Model Complexity

How to reduce Overfitting?

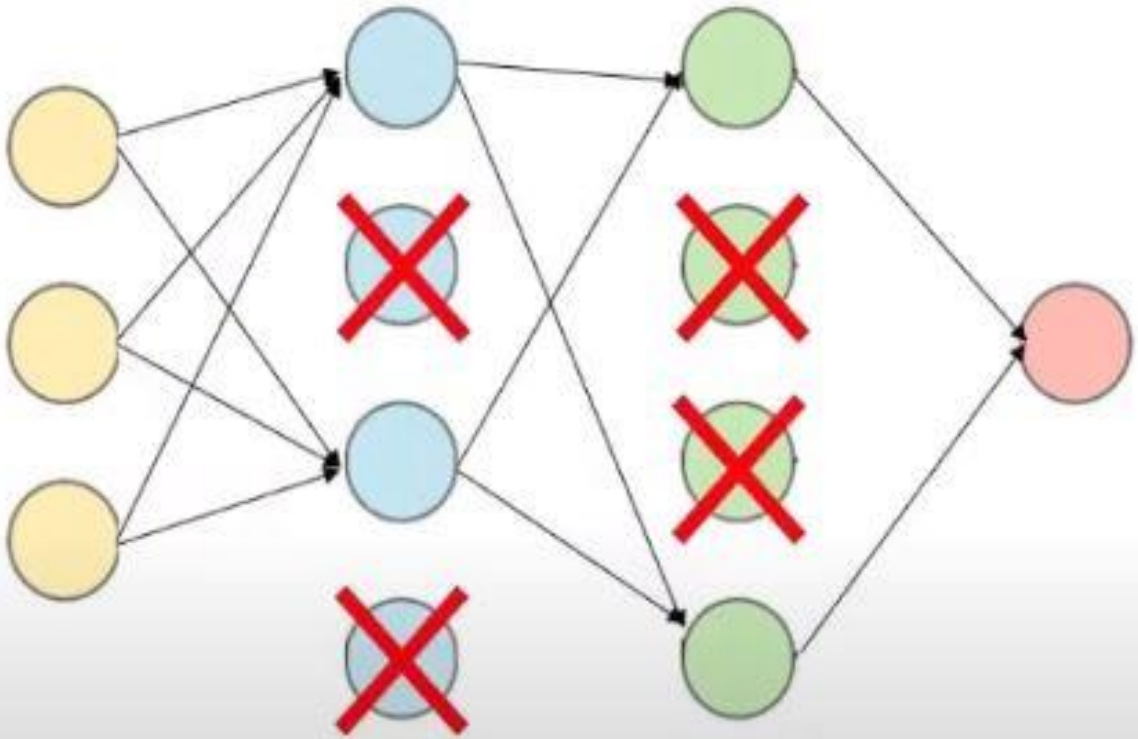
- Reduce complexity of Neural Network
- Regularization
- Dropout
- Reduce the number of epochs - the training duration

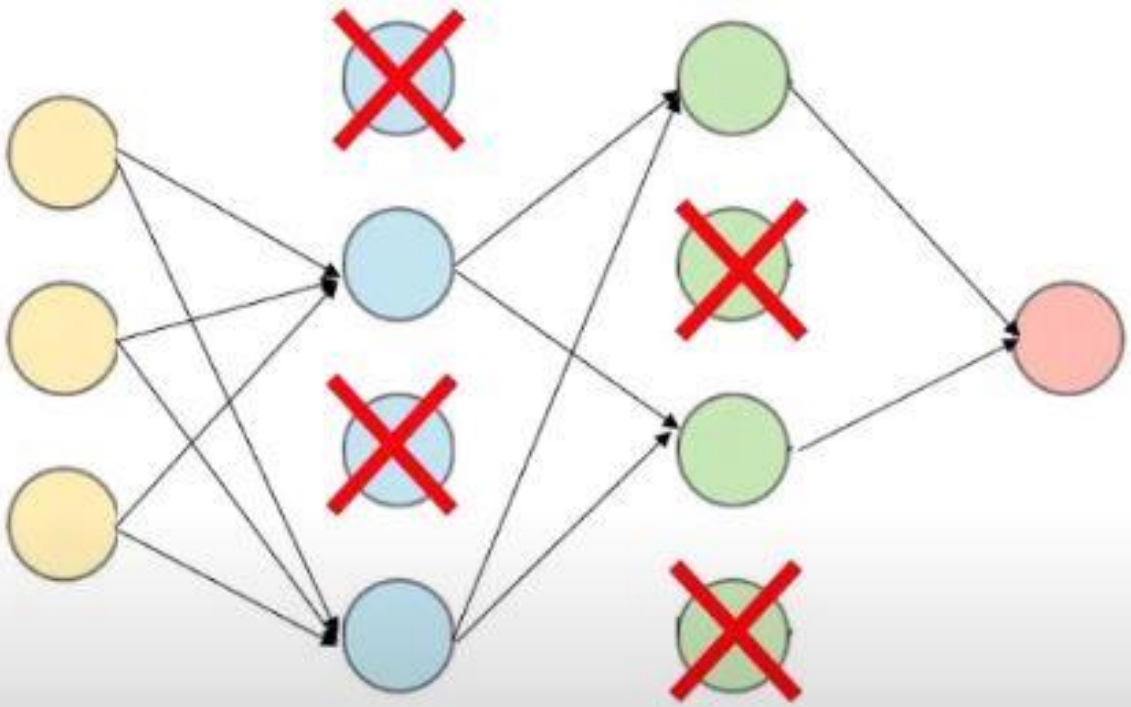






Train Dataset





New = Ak

```
D = np.random.rand(A.shape[0], A.shape[1])
```

```
[[0.13691879 0.12486968 0.76703112 0.80144702]
 [0.61693956 0.71061175 0.39941722 0.57400298]
 [0.01633737 0.94949367 0.93394508 0.27578972]
 [0.46148637 0.09595069 0.96184166 0.66788549]
 [0.15366178 0.32287791 0.47652757 0.26758814]]
```

```
D = D < keep_rate
```

= 0.8

```
[[ True  True  True False]
 [ True  True  True  True]
 [ True False False  True]
 [ True  True False  True]
 [ True  True  True  True]]
```

```
A = A * D
```

```
[[0.23869521 0.96682018 0.11877874 0.57609517]
 [0.94703416 0.51971021 0.11884226 0.26263551]
 [0.02637477 0.60834206 0.97092361 0.32589815]
 [0.34884988 0.36680243 0.95451758 0.83512696]
 [0.43897514 0.54082403 0.90827132 0.82675546]]
```

*

```
[[ True  True  True False]
 [ True  True  True  True]
 [ True False False  True]
 [ True  True False  True]
 [ True  True  True  True]]
```

=

```
[[0.23869521 0.96682018 0.11877874 0.        ]
 [0.94703416 0.51971021 0.11884226 0.26263551]
 [0.02637477 0.         0.         0.32589815]
 [0.34884988 0.36680243 0.         0.83512696]
 [0.43897514 0.54082403 0.90827132 0.82675546]]
```

Forward Propagation

Z1 = W1 * A0 + B1
A1 = f (Z1)
D1 = np.random.rand(A1.shape[0], A1.shape[1]) < keep_rate
A1 = A1 * D1

Z2 = W2 * A1 + B2
A2 = f (Z2)
D2 = np.random.rand(A2.shape[0], A2.shape[1]) < keep_rate
A2 = A2 * D2

Z3 = W3 * A2 + B3
A3 = f (Z3)

Forward Propagation

$$Z1 = W1 * A0 + B1$$
$$A1 = f(Z1)$$
$$D1 = np.random.rand(A1.shape[0], A1.shape[1]) < keep\_rate$$
$$A1 = A1 * D1 \ / \ keep\_rate$$

$$Z2 = W2 * A1 + B2$$
$$A2 = f(Z2)$$
$$D2 = np.random.rand(A2.shape[0], A2.shape[1]) < keep\_rate$$
$$A2 = A2 * D2 \ / \ keep\_rate$$

$$Z3 = W3 * A2 + B3$$
$$A3 = f(Z3)$$

How to reduce Underfitting?

THE CODING LIBRAR

- Make the Neural Network more complex
- Add more training data so the network can learn more about the specific task
- Preprocessing of the data - Remove noise as an example
- Increase number of epochs - Increase training duration
- Fine-tune parameters for the network and training process