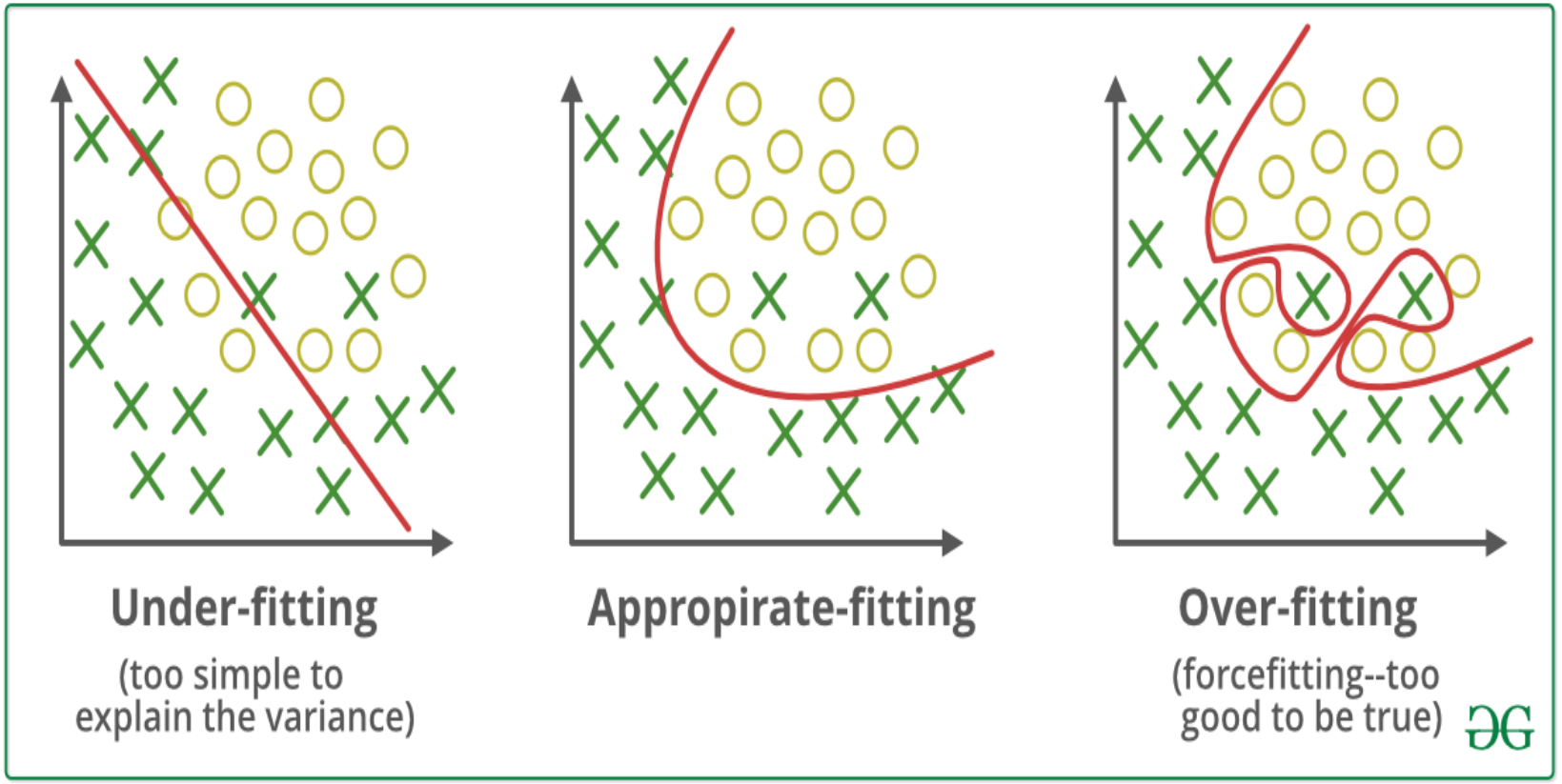
Regularization

Overfitting



Overfitting is an undesirable machine learning behavior that occurs when the machine learning model gives accurate predictions for training data but not for new data.

Why does overfitting occur?

- Overfitting occurs when the model cannot generalize and fits too closely to the training dataset instead.

Overfitting happens due to several reasons

1. The training data size is too small and does not contain enough data samples to accurately represent all possible input data values.
2. The training data contains large amounts of irrelevant information, called noisy data.
3. The model trains for too long on a single sample set of data.
4. The model complexity is high, so it learns the noise within the training data.

How can you detect overfitting?

- K-fold cross-validation

How can you prevent overfitting?

1. **Early stopping**
   Early stopping pauses the training phase before the machine learning model learns the noise in the data.

2. **Pruning**

3. **Regularization** is a collection of training/optimization techniques that seek to reduce overfitting. These methods try to eliminate those factors that do not impact the prediction outcomes by grading features based on importance.

4. **Ensembling**
   Ensembling combines predictions from several separate machine learning algorithms.

5. **Data augmentation**
   Data augmentation is a machine learning technique that changes the sample data slightly every time the model processes it.

Regularization

- Regularization helps mitigate **overfitting** by ensuring the model does not become excessively complex.

How Does Regularization Work?

- Regularization works by adding a penalty term to the loss function, discouraging large weights (parameters). This forces the model to prioritize **simpler** patterns and reduces sensitivity to noise.

$$L = L_{\text{original}} + \lambda R(W)$$

where:

- L_{original} is the original loss function (e.g., Mean Squared Error, Cross-Entropy).
- $R(W)$ is the regularization term.
- λ (lambda) is a hyperparameter controlling the strength of regularization.

A **higher** λ makes the model simpler (more bias, less variance).

A **lower** λ allows more complexity (less bias, more variance).

Types of Regularization

- L1 Regularization (Lasso Regression)
  - Encourages sparsity by shrinking some weights to exactly zero.
  - Effectively performs feature selection by removing irrelevant features.
  - Commonly used in high-dimensional datasets.

$$R(W) = \lambda \sum |w_i|$$

λ is a regularization parameter

Example: Selecting the most important features in a regression model when there are hundreds of variables.

Types of Regularization

- L2 Regularization (Ridge Regression)

$$R(W) = \lambda \sum w_i^2$$

- Shrinks weights smoothly but does not set them to zero.
- Prevents extreme weight values, leading to a more stable and generalizable model

  Example: Used in regression tasks where all features contribute to the outcome but need balanced importance.

Elastic Net Regularization

- A combination of L1 and L2: Useful when there are correlated features.
- Retains both feature selection (L1) and weight stabilization (L2).

$$R(W) = \lambda_1 \sum |w_i| + \lambda_2 \sum w_i^2$$

- **Example :**
  - Handling datasets where some features are redundant while others are crucial.

Regularization in Neural Networks

- In deep learning, overfitting is a significant issue due to the high complexity of models. Apart from L1/L2 regularization (often called **weight decay**), additional techniques include:

1. **Dropout:**
2. **Batch Normalization:**
3. **Early Stopping:**

Choosing the Right Regularization

Regularization Type	Effect	When to Use
1 (Lasso)	Encourages sparsity	Feature selection, high-dimensional data
L2 (Ridge)	Prevents large weights	When all features are relevant
Elastic Net	Combination of L1 & L2	When features are correlated
Dropout	Drops neurons randomly	Deep learning models
Early Stopping	Stops training early	To prevent excessive overfitting

L1 reguralization

Compute the Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_{\text{true},i})^2$$

Compute the L1 regularization term:

$$\text{L1 regularization} = \lambda\left(|w_0| + |w_1|\right)$$

Compute the total loss:

$$\text{Total loss} = \text{MSE} + \text{L1 regularization}$$

Given:

Data: $x = [1, 2, 3, 4, 5], y_{\text{true}} = [2, 4, 6, 8, 10]$

Predicted values: $\hat{y} = w_0 + w_1 \times x$

Initial weights: $w_0 = 0.5, w_1 = 0.5$

Regularization parameter: $\lambda = 0.1$

L1 regularization

- L1 regularization penalty encourages sparsity in the weights, which means that some of the weights may become exactly zero.
- In this example, it is possible that either w0 or w1 (or both) become zero, depending on the strength of the regularization and the size of the dataset.

Ridge Regression (L2 Regularization)

For a linear regression model with 2 features and weights w1 and w2. $y = w_1x_1 + w_2x_2 + b$

The cost function for linear regression without regularization is:

$$J(w_1, w_2) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

m is the number of training examples.

To prevent overfitting, we add the **L2 regularization** term. The **regularized cost function** becomes:

$$J(w_1, w_2) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2$$

Sum

The actual values of the targets (y) are [3,5,7]

The predicted values (y hat) from the model are [2.8,5.2,6.9].

The weights of the model are $w1$=1.2 and $w2$=0.8 λ=0.1 is the regularization parameter.

L2 Regularization

- **Without regularization**, the model focuses purely on minimizing the error between predicted and actual values. But, if the model starts relying too much on certain features, it could overfit the training data—meaning the model is too complex and may not perform well on new data (test data).

- **With L2 regularization**, we penalize the model for having large weights. The regularization term is proportional to the sum of the squares of the weights. By adding this penalty, the model is forced to find a balance between fitting the data and keeping the weights small. Smaller weights mean a simpler model, which tends to generalize better to unseen data, reducing the risk of overfitting.