Model Selection and Regulations

Dr. Adel Abusitta

Bias and Variance in Machine Learning

- Machine learning allows machines to perform **data analysis and make predictions.**
- If the machine learning model is not accurate, it can make **predictions errors**, and these prediction errors are usually known as **Bias and Variance.**
- These errors will always be present as there is always a **slight difference between the model predictions and actual predictions.**
- The main aim of ML/data science analysts is to **reduce these errors in order to get more accurate results.**

https://www.javatpoint.com/bias-and-variance-in-machine-learning

Variance and Bias

- In machine learning, we aim for **low variance** and **low bias** in our models.

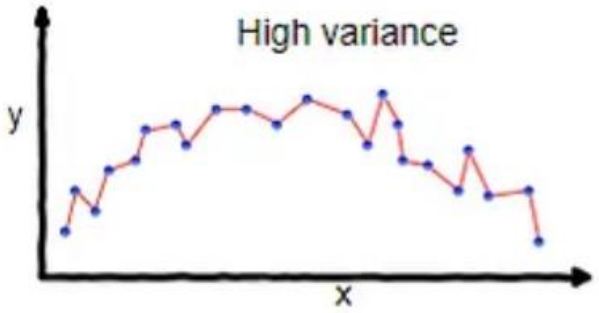
Variance: Variance refers to the variability or spread of a model's predictions when trained on different subsets of the training data. **High variance indicates that the model is sensitive to the specific data points in the training set, leading to significantly different predictions for different subsets of data.** High variance can be a sign of **overfitting**, where the model learns to fit the noise of the training data rather than capturing the underlying patterns.

Bias: Bias refers to the error introduced by approximating a real-world problem with a simplified model. It represents the difference between the predicted values by the model and the true values**. High bias indicates that the model is too simplistic, causing it to miss important patterns or relationships in the data**. High bias can be a sign of **underfitting**, where the model fails to capture the underlying complexity of the problem.

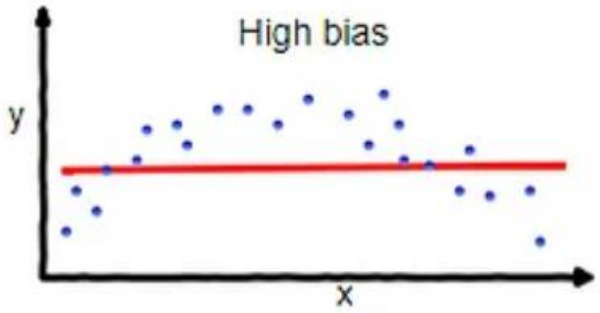
Bias and Variance

- **Ideally**, we want to find **a balance between low variance and low bias**, which is known as the **bias-variance trade-off**. The goal is to develop a model that **generalizes well to unseen data by reducing both types of errors.**
- **Overfitting** occurs when the model has **low bias but high variance**, meaning it fits the training data too closely but fails to generalize to new data.
- **Underfitting** occurs when the model has **high bias but low variance**, meaning it oversimplifies the problem and fails to capture important patterns.

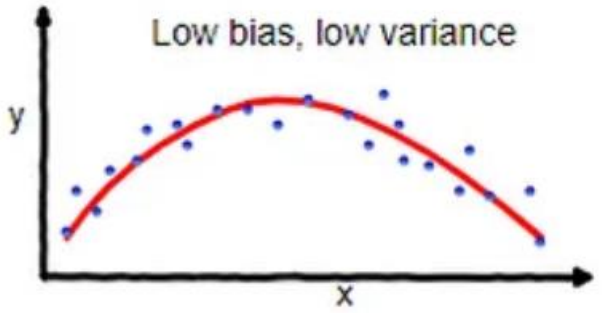
Bias and Variance



overfitting



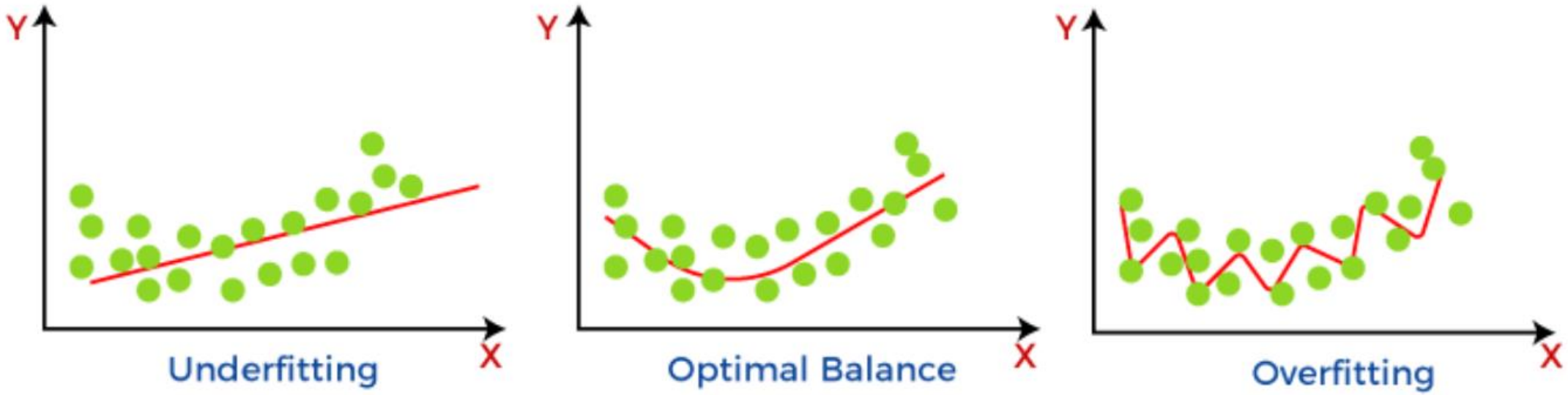
underfitting



Good balance

Source: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

Bias and Variance



Source: https://www.javatpoint.com/bias-and-variance-in-machine-learning

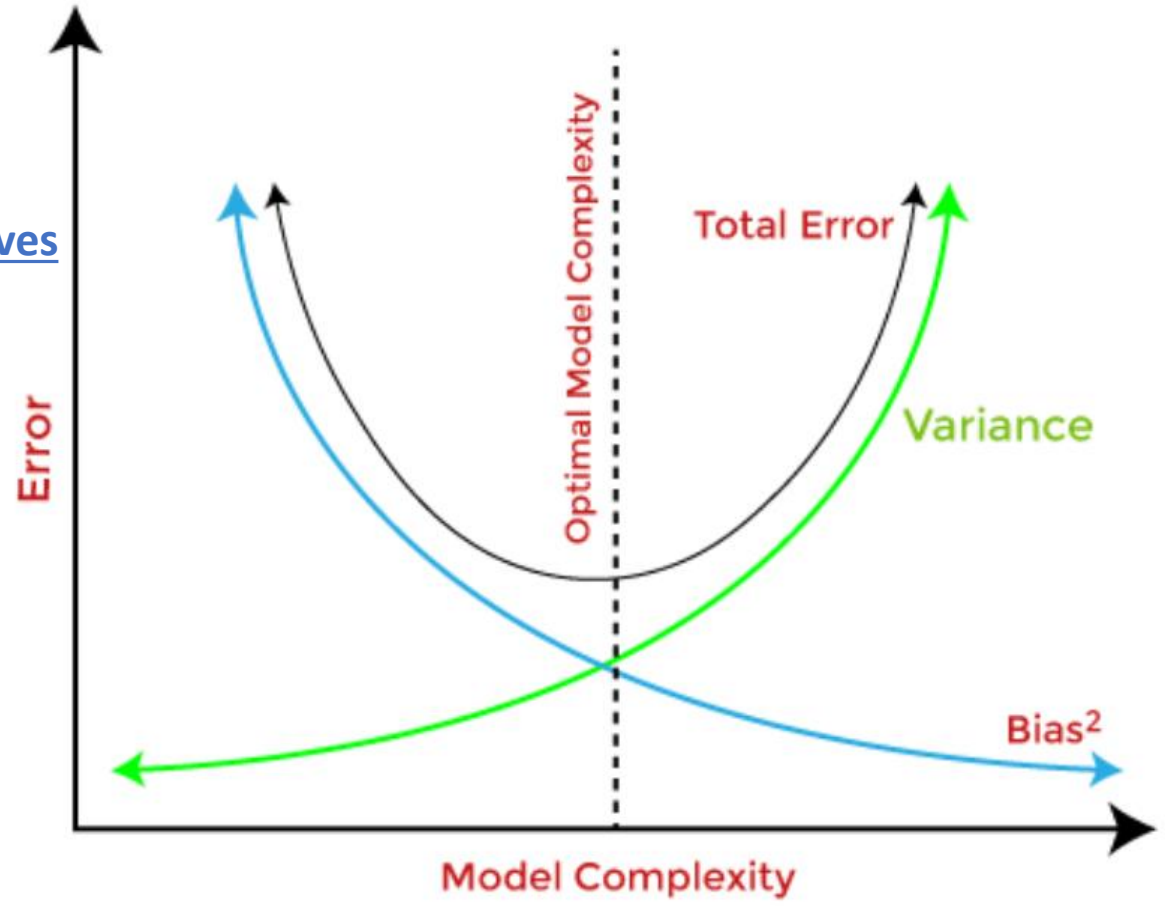
Bias-Variance Trade-Off

For an accurate prediction of the model **a low variance** and **low bias.**

The trade-off occurs because reducing bias often involves increasing model complexity, which leads to increased sensitivity to the training data and higher variance.

As a model becomes more complex, it tends to have lower bias but higher variance, and vice versa.

- If we decrease the variance, it will increase the bias.
- If we decrease the bias, it will increase the variance.



How to compute bias and variance of ML?

- The bias and variance of a model can be calculated using the following equations:

$$\textbf{Bias} = E[(\hat{y} - y)^2]$$
$$\textbf{Variance} = E[(\hat{y} - E[\hat{y}])^2]$$

where:

- $\hat{y}$ is the predicted output of the model
- y is the true output
- E[.] denotes the expected value or average over different training sets

The bias represents the average difference between the predicted values and the true values. It measures how much the model's predictions deviate from the actual values on average.

Bias and Variance

How to achieve good balance? we employ various techniques:

- **Model selection:** Choosing an appropriate model complexity can help balance bias and variance. Too simple models may have high bias, while too complex models may have high variance. Techniques like **cross-validation** can help select the optimal model complexity.
- **Regularization:** Regularization techniques, such as L1 (Lasso) and L2 (Ridge) regularization, help reduce variance by **adding a penalty term** to the model's objective function. This discourages complex or large coefficient values, thus preventing overfitting.
- **Feature engineering:** Careful feature selection and engineering can help reduce both bias and variance. **Including informative features** helps **reduce bias**, while **excluding irrelevant or noisy features** helps **reduce variance.**

Model selection

- Model selection is a crucial step in machine learning where you choose the best model for a particular task from a set of candidate models.

- The goal is to find the model that achieves the **highest performance** and **generalizes well to unseen data**.

Model selection

1. **Dataset Split:** Suppose you have a dataset with 1000 samples. The first step is to split the dataset into two parts: a training set and a test set. For example, you can use an 80:20 split, where 80% of the data (800 samples) is used for training, and the remaining 20% (200 samples) is held out for evaluation.
2. **Cross-Validation Folds:** The training set is further divided into K subsets or folds for cross-validation. Typically, K is chosen as a value between 5 and 10, but it can vary depending on the dataset size and other considerations.
3. **Model Training and Evaluation:** The cross-validation process begins by selecting one fold as the validation set and using the remaining K-1 folds for training the model. This process is repeated K times, with each fold taking a turn as the validation set. The model is trained and evaluated K times, resulting in K performance scores.
4. **Performance Metrics:** For each fold, you can evaluate the model's performance using appropriate metrics for your specific problem, such as accuracy. The performance scores obtained for each fold are then averaged to obtain an overall estimation of the model's performance.
5. **Hyperparameter Tuning:** Cross-validation is also commonly used for hyperparameter tuning. Hyperparameters are parameters that are not learned from the data, but rather set by the modeler before training. By using cross-validation, different combinations of hyperparameters can be evaluated, and the best-performing set of hyperparameters can be selected.

Model selection - Dataset Split

Suppose you have a dataset of 100 samples. You decide to use 5-fold cross-validation, meaning the dataset will be divided into 5 equal-sized subsets or folds. Here's how the process would unfold:

Dataset Split: Split the dataset into a training set and a test set. For this example, let's use an 80:20 split, where 80 samples (80%) will be used for training, and 20 samples (20%) will be held out for testing.

Model selection - Cross-Validation Folds

Model Training and Evaluation: The cross-validation process begins. For each iteration, one fold is selected as the validation set, while the remaining folds are used for training the model. The model is trained on the training folds and evaluated on the validation fold. This process is repeated for each fold, resulting in 5 iterations.

K Fold CV, K=5

Credit: https://towardsdatascience.com/cross-validation-k-fold-vs-monte-carlo-e54df2fc179b

Model selection - Cross-Validation Folds

- The purpose of the validation set is to **provide an unbiased estimate of the model's performance on unseen data.**

- It helps in **monitoring the model's behavior** during training and **tuning hyperparameters** to prevent overfitting.

- By evaluating the model on the validation set, we can make informed **decisions about the model's configuration, select the best model, or adjust parameters to improve its performance**.

Model selection - Cross-Validation Folds

- **Performance Metrics:** For each iteration, you can evaluate the model's performance using your chosen metrics, such as accuracy, precision, recall, or others. **You will have 5 performance scores**, one for each fold.
- **Average Performance:** compute the average of the performance scores obtained from the cross-validation iterations. This average provides an overall estimation of the model's performance on the training data.
- Cross-validation helps provide a more reliable estimate of a model's performance by leveraging multiple train-test splits.
- It helps **assess how well the model generalizes to unseen data** and provides insights into its stability and potential for overfitting.

Regulations

Regulations

- Regularization techniques, such as **L1 (Lasso regularization)** and **L2(Ridge regularization)**, are designed to combat overfitting.

They are used to:

- add **a penalty term to the model's objective function**, regularization encourages smaller coefficients
- help prevent the model from overemphasizing specific features or capturing noise in the data. This, in turn, reduces variance and improves **the model's ability to generalize to unseen data.**

“reducing variance alone does not guarantee the prevention of overfitting. Proper model selection, feature engineering, regularization are essential to address overfitting effectively.”

Regulations

$$Loss = Error(y, \hat{y})$$

Loss function with no regularisation

data loss and the regularization term

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

Loss function with L1 regularisation

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

Loss function with L2 regularisation

Regulations (logistic classification)

- Without regularization, the log-loss objective function for logistic regression is:

$$L(\beta) = -1/N * \Sigma[y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i)]$$

- Where:
- N is the number of samples in the dataset.
- y_i is the true label (0 or 1) for the i-th sample.
- p_i is the predicted probability of the positive class (1) for the i-th sample based on the logistic regression model.

Regulations (logistic classification)

- When L1 or L2 regularization is applied, the objective function is modified **by adding a penalty term to control the complexity of the model.**
- For L1 regularization (Lasso regularization), the modified objective function is:

$$L(\beta) = -1/N * \Sigma[y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i)] + \lambda * \Sigma|\beta|$$

Where:

- λ is the regularization parameter that controls the strength of the penalty term.
- $|\beta|$ is the L1 norm of the coefficient vector β.

Regulations (logistic classification)

- For L2 regularization (Ridge regularization), the modified objective function is:

$$L(\beta) = -1/N * \Sigma[y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i)] + \lambda * \Sigma(\beta^2)$$

Where:

- λ is the regularization parameter that controls the strength of the penalty term.
- β^2 is the square of the L2 norm (or Euclidean norm) of the coefficient vector β.

Regulations (logistic classification)

- The **addition of the penalty term in the objective** function helps in **controlling the complexity of the model** and **reducing overfitting** by discouraging large coefficient values.

- The regularization parameter (λ) determines the **trade-off between the data fit and the regularization term.**

- Higher values of λ increase the regularization effect and can lead to more shrinkage or sparsity in the coefficient values.

Regulations (logistic classification)

- Regularization, **whether L1 or L2, helps reduce overfitting in logistic regression by discouraging large coefficient values.**

- By adding a regularization term to the objective function, **the model is penalized for having large coefficient values**. This penalty encourages the model to favor smaller coefficient values, effectively shrinking the impact of individual features in the prediction process.
- As a result, the model becomes **less sensitive to noise** and **less likely to overfit the training data.**
- when the regularization parameter (λ) is increased, **the impact of the regularization term on the objective function also increases.** Consequently, the model becomes more biased towards smaller coefficient values, further reducing the risk of overfitting.
- By controlling the complexity of the model through regularization, logistic regression strikes a balance between fitting the training data and generalizing well to new, unseen data.

The choice between L1 and L2 regularization

L1 regularization (Lasso):

- **Feature selection:** L1 regularization has the property of driving some coefficients to exactly zero, which makes it **useful for feature selection.** If you have a high-dimensional dataset with many irrelevant or redundant features, L1 regularization can help in automatically selecting the most important features.
- **Simplicity and interpretability:** L1 regularization tends to create sparse models with fewer non-zero coefficients, which can be beneficial if you prefer simpler models that are easier to interpret.

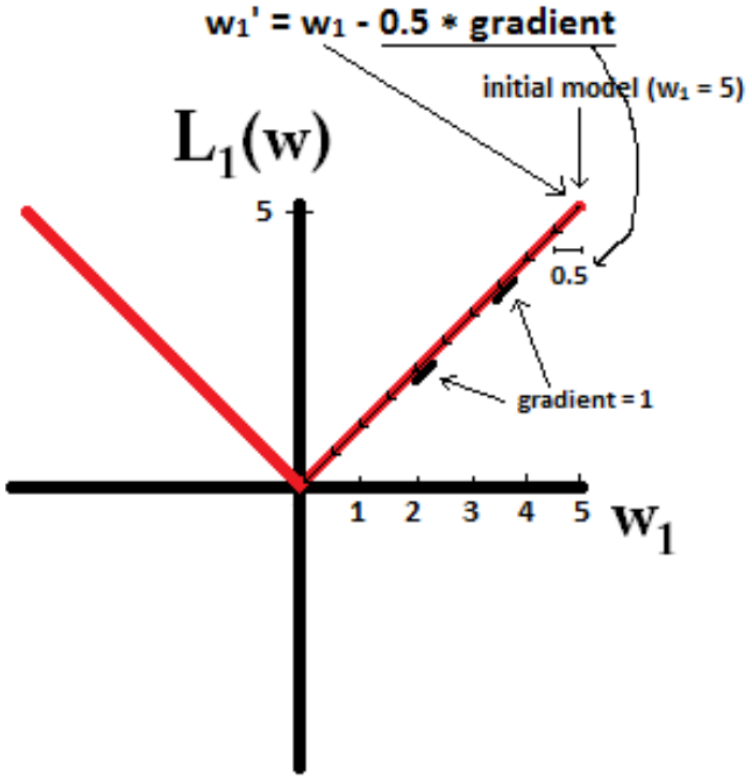
The choice between L1 and L2 regularization

L2 regularization (Ridge):

- **Smoothing and stability:** L2 regularization penalizes the squared magnitudes of the coefficients, which helps to reduce the impact of outliers and can lead to smoother and more stable models.
- **Overall performance:** L2 regularization typically improves the overall performance of the model by reducing overfitting and improving generalization. It is a widely used regularization technique that works well in many scenarios.

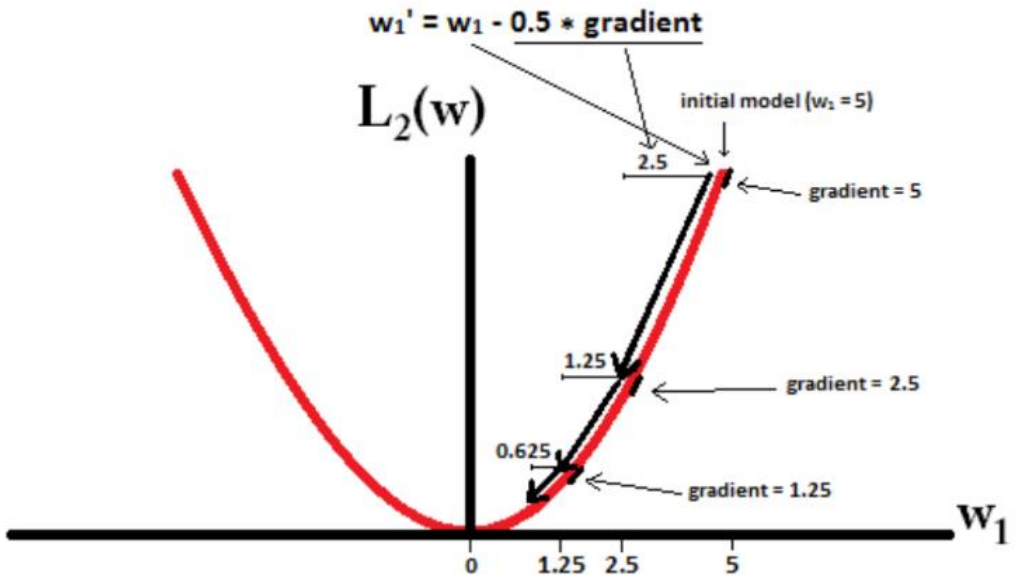
Why L1 norm for sparse models?

L1-regularization



$$w_1 := w_1 - \eta \cdot \frac{dL_1(w)}{dw} = w_1 - \frac{1}{2} \cdot 1,$$

L2-regularization



$$w_1 := w_1 - \eta \cdot \frac{dL_2(w)}{dw} = w_1 - \frac{1}{2} \cdot w_1$$

the gradient is w1, causing every step to be only halfway towards 0.

Credit: https://stats.stackexchange.com/questions/45643/why-l1-norm-for-sparse-models

The choice between L1 and L2 regularization

- In practice, it is often a good idea to try both L1 and L2 regularization and evaluate their impact on your specific problem. You can compare the performance, interpretability, and the sparsity of the resulting models to determine which regularization technique is more suitable.