UNIT 2
Model Assessment and Selection
Bias,Variance and Model Complexity

Bias, variance, and model complexity are fundamental concepts in machine learning and statistical modelling that help us understand the trade-offs and challenges involved in building predictive models.

Bias

Bias refers to the error introduced by approximating a real-world problem, which may be complex, by a simplified model.
It can be seen as the difference between the predicted values and the true values in a model.

Low Bias: A low-bias model fits the training data very well but may perform poorly on unseen data. It is often associated with complex, flexible models.

High Bias: A high-bias model is too simplistic and may underfit the data, failing to capture important patterns. It is often associated with overly simple models.

Variance

Variance refers to the model's sensitivity to small fluctuations or noise in the training data. It measures how much the model's predictions would vary if we trained it on different subsets of the data.

Low Variance: A low-variance model is stable and produces consistent predictions across different datasets. However, it may not adapt well to complex patterns and can underfit.

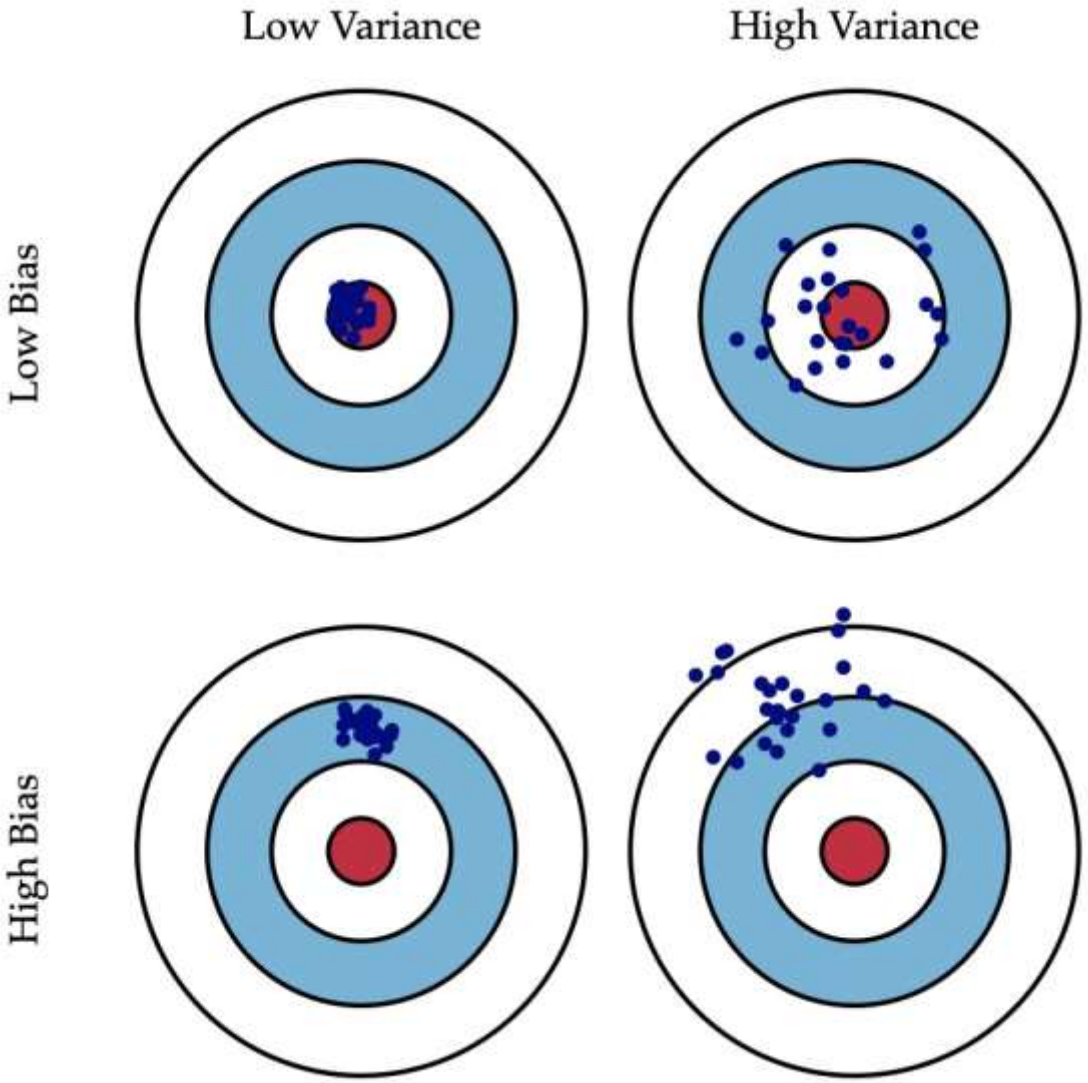
High Variance: A high-variance model is sensitive to the training data and can produce very different predictions for different datasets. It is often associated with overfitting, where the model captures noise in the data.

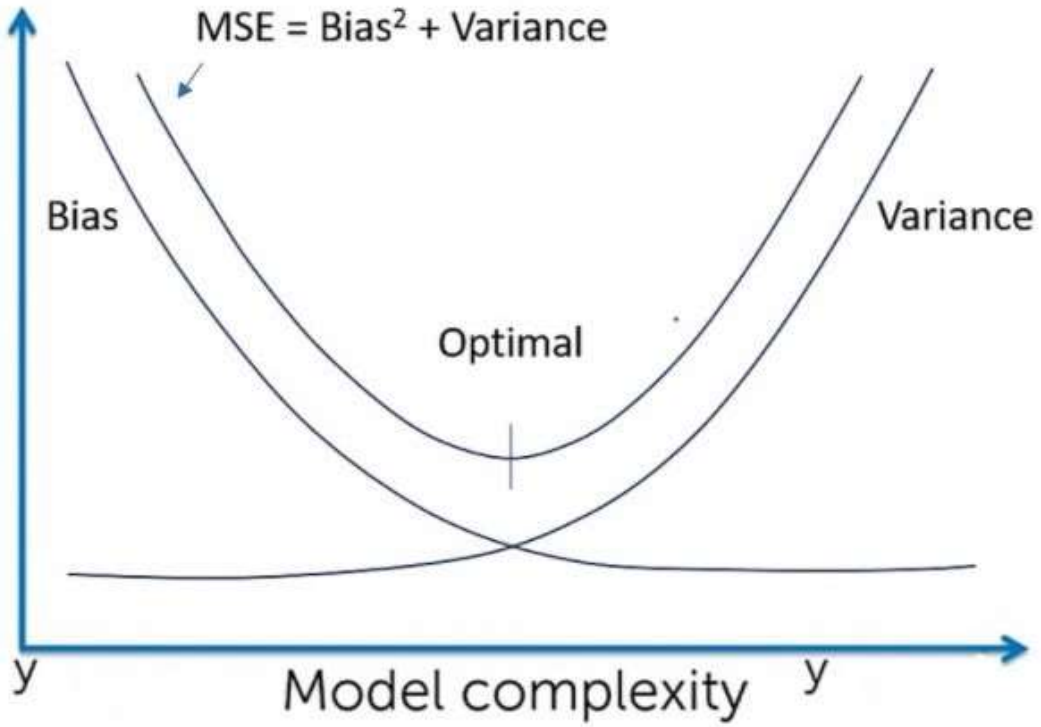
Model Complexity

Model complexity refers to the sophistication or flexibility of a model in capturing relationships within data. It's often controlled by the number of parameters or features in the model.

Low Model Complexity: Models with low complexity are simple and have fewer parameters. They are less likely to overfit but may underfit if the data is complex.

High Model Complexity: Models with high complexity are more sophisticated and have many parameters. They can capture intricate patterns in the data but are prone to overfitting when there's noise or insufficient data.




MSE = Bias² + Variance
Bias
Variance
Optimal
y
Model complexity
y

Bias-Variance tradeoff

The bias-variance trade-off is a fundamental concept in machine learning and statistics that deals with the balance between two types of errors that a predictive model can make: bias and variance.
Achieving the right balance between these two types of errors is crucial for building models that generalize well to unseen data.
Predicted Error: Predicted error generally refers to the difference between the actual (ground truth) values and the values predicted by a machine learning model. It can be measured using various metrics, such as mean squared error (MSE), mean absolute error (MAE), or root mean squared error (RMSE), depending on the problem type (e.g., regression or classification). Predicted error helps assess how well the model is performing on a specific set of data.

Training error is the average loss over the training sample

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i)).$$

Test Error: Test error, also known as out-of-sample error or generalization error, measures how well a machine learning model performs on data that it has never seen during training. Typically, a dataset is split into a training set and a test set. The model is trained on the training set and then evaluated on the test set. Test error provides an estimate of how well the model is expected to perform on new, unseen data. It helps assess the model's ability to generalize beyond the training data.
Test error, also referred to as generalization error, is the prediction error over an independent test sample

$$\text{Err}_{\mathcal{T}} = \text{E}[L(Y, \hat{f}(X))|\mathcal{T}]$$

Validation Error: Validation error is closely related to test error but is often used during the model training process, especially in techniques like cross-validation. In k-fold cross-validation, for example, the dataset is divided into k subsets (folds), and the model is trained and validated k times. Validation error measures how well the model performs on the validation set (which is part of the original data but not used in training) during each iteration of cross-validation. It helps in hyperparameter tuning and model selection.

. Training error consistently decreases with model complexity, typically dropping to zero if we increase the model complexity enough. However, a model with zero training error is overfit to the training data and will typically generalize poorly

The trade-off between bias and variance can be visualized as follows:

High Bias, Low Variance: Models with high bias and low variance are simple and tend to underfit the data. They may not capture complex patterns and have a relatively high training error.

Low Bias, High Variance: Models with low bias and high variance are complex and can fit the training data very closely. However, they are prone to overfitting and have a high test error when applied to new data.

Balanced Trade-off: The goal is to strike a balance between bias and variance. You want a model that is complex enough to capture the underlying patterns in the data but not so complex that it overfits and becomes sensitive to noise.

Achieving this balance often involves techniques like:

Regularization: Adding constraints or penalties to the model to reduce its complexity and prevent overfitting.

Cross-Validation: Evaluating the model's performance on multiple validation sets to get a better estimate of its generalization error.

Feature Selection: Choosing relevant features and reducing dimensionality to simplify the model.

Ensemble Methods: Combining the predictions of multiple models to reduce variance while maintaining low bias.

The ultimate goal of the bias-variance trade-off is to build models that generalize well to new, unseen data, rather than just memorizing the training data or being too simplistic to be useful.

Optimism of the Training error rate

Given a training set T = {(x1,y1),(x2,y2),...(xN ,yN)} the generalization error of a model ˆf is

$$\text{ErrT} = \text{EX0,Y 0 [L(Y 0 , ˆf(X0))|T];}$$

Note that the training set T is fixed in expression . The point (X0 ,Y 0) is a new test data point, drawn from F, the joint distribution of the data. Averaging over training sets T yields the expected error

$$\text{Err} = \text{ET EX0,Y 0 [L(Y 0 , ˆf(X0))|T],}$$

which is more amenable to statistical analysis. As mentioned earlier, it turns out that most methods effectively estimate the expected error rather than ET t. Now typically, the training error

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

will be less than the true error ErrT , because the same data is being used to fit the method and assess its error.**A fitting method typically adapts to the training data, and hence the apparent or training error err will be an overly optimistic estimate of the generalization error ErrT .**

We define the optimism as the difference between Err in sample and the training error

$$\text{op} \equiv \text{Err}_{\text{in}} - \overline{\text{err}}.$$

.

This is typically positive since err is usually biased downward as an estimate of prediction error. Finally, the average optimism is the expectation of the optimism over training sets

$$\omega \equiv \text{Ey(op).}$$

we can estimate the expected error Err rather than the conditional error ErrT .

For squared error, 0–1, and other loss functions, one can show quite generally that

$$\omega = \frac{2}{N}\sum_{i=1}^{N}\mathrm{Cov}(\hat{y}_i, y_i),$$

here Cov indicates covariance. Thus the amount by which err underestimates the true error depends on how strongly yi affects its own prediction. The harder we fit the data, the greater Cov(ˆyi ,yi) will be, thereby increasing the optimism. This result for squared error loss where yˆi is the fitted value from the regression. For 0–1 loss, ˆyi ∈ {0, 1} is the classification at xi , and for entropy loss, ˆyi ∈ [0, 1] is the fitted probability of class 1 at xi .

$$\mathrm{E}_{\mathbf{y}}(\mathrm{Err}_{\mathrm{in}}) = \mathrm{E}_{\mathbf{y}}(\overline{\mathrm{err}}) + \frac{2}{N}\sum_{i=1}^{N}\mathrm{Cov}(\hat{y}_i, y_i).$$

This expression simplifies if ˆyi is obtained by a linear fit with d inputs or basis functions. For example,

$$\sum_{i=1}^{N}\mathrm{Cov}(\hat{y}_i, y_i) = d\sigma_\varepsilon^2$$

for the additive error model Y = f(X) + ε, and so

$$\mathrm{E}_{\mathbf{y}}(\mathrm{Err}_{\mathrm{in}}) = \mathrm{E}_{\mathbf{y}}(\overline{\mathrm{err}}) + 2\cdot\frac{d}{N}\sigma_\varepsilon^2.$$

The optimism increases linearly with the number d of inputs or basis functions we use, but decreases as the training sample size increases.

An obvious way to estimate prediction error is to estimate the optimism and then add it to the training error err. The methods described in the next section—Cp, AIC, BIC and others—work in this way, for a special class of estimates that are linear in their parameters.

Estimates of In-Sample prediction Error

In-sample error is not usually of direct interest since future values of the features are not likely to coincide with their training set values. But for comparison between models, in-sample error is convenient and often leads to effective model selection. The reason is that the relative (rather than absolute) size of the error is what matters.

The general form of the in-sample estimates is

$$\widehat{\mathrm{Err}}_{\mathrm{in}} = \overline{\mathrm{err}} + \hat{\omega},$$

where ˆω is an estimate of the average optimism

Using expression applicable when d parameters are fit under squared error loss, leads to a version of the so-called Cp statistic,

$$C_p = \overline{\mathrm{err}} + 2 \cdot \frac{d}{N} \hat{\sigma_\varepsilon}^2.$$

Here ˆσε 2 is an estimate of the noise variance, obtained from the meansquared error of a low-bias model. Using this criterion we adjust the training error by a factor proportional to the number of basis functions used.

The Akaike information criterion is a similar but more generally applicable estimate of Errin when a log-likelihood loss function is used. It relies on a relationship similar to (7.24) that holds asymptotically as N → ∞:

$$\mathrm{AIC} = -\frac{2}{N} \cdot \mathrm{loglik} + 2 \cdot \frac{d}{N}.$$

AIC statistic is equivalent to Cp, and so we refer to them collectively as AIC.To use AIC for model selection, we simply choose the model giving smallest AIC over the set of models

considered. For nonlinear and other complex models, we need to replace d by some measure of model complexity.

Cp (Mallow's Cp):

- Purpose: Cp was developed by Colin Mallows to assess the quality of linear regression models. It is primarily used in the context of regression analysis.
- Calculation: Cp measures the trade-off between model fit and model complexity. It is calculated as follows: Cp = (SSE_p / MSE) - (n - 2p) Where:
- SSE_p: The sum of squared errors for the model with p predictor variables.
- MSE: The mean squared error for the full model with all predictor variables.
- n: The number of data points.
- p: The number of predictor variables in the model.
- Interpretation: A smaller Cp value indicates a better trade-off between model fit and complexity. Cp is used to assess whether a model with a subset of predictor variables is competitive with the full model while penalizing for model complex

AIC (Akaike Information Criterion):

- Purpose: AIC is a general-purpose model selection criterion used in various statistical models, including linear regression, time series analysis, and more.
- Calculation: AIC is based on the likelihood function of the model and is calculated as follows: AIC = -2 * log(Likelihood) + 2 * k Where:
- Likelihood: The likelihood of the model given the data.
- k: The number of estimated parameters in the model.
- Interpretation: AIC balances the fit of the model and its complexity. A lower AIC value indicates a better model. It effectively penalizes models with more parameters, encouraging simplicity.

BIC (Bayesian Information Criterion):

- Purpose: BIC is similar to AIC but tends to penalize model complexity more heavily. It is also used for model selection in various statistical contexts.
- Calculation: BIC is calculated as follows: BIC = -2 * log(Likelihood) + k * log(n) Where:

- Likelihood: The likelihood of the model given the data.
- k: The number of estimated parameters in the model.
- n: The number of data points.
- Interpretation: BIC favors simpler models more strongly than AIC. A lower BIC value indicates a better model. Compared to AIC, BIC is more conservative when it comes to model selection, often resulting in a more parsimonious choice.

The choice between Cp, AIC, and BIC depends on the specific context and goals of your analysis. Cp is suitable for linear regression, while AIC and BIC are versatile and widely used in various statistical modeling scenarios. BIC is the most conservative in terms of model selection and favors simpler models the most, while AIC strikes a balance between model fit and complexity.

Effective number of parameters

The concept of "number of parameters" can be generalized, especially to models where regularization is used in the fitting.

The effective number of parameters in model evaluation, assessing the complexity of a model and understanding its generalization performance.

Overfitting and Generalization:

- Models with too many parameters relative to the amount of training data are prone to overfitting. They may capture noise in the training data rather than learning the underlying patterns.
- The effective number of parameters helps in evaluating whether a model is likely to generalize well to new, unseen data.

Model Complexity:

- Effective parameters provide a more nuanced measure of a model's complexity compared to the total number of parameters.
- During evaluation, understanding the effective number helps in assessing whether the model is unnecessarily complex or if it strikes a good balance between complexity and generalization.

Regularization Impact:

- Regularization techniques, such as L1 and L2 regularization, directly influence the effective number of parameters.
- Evaluation involves assessing the impact of regularization on the model's performance and understanding how it helps control overfitting.

Interpretability:

- Models with a high effective number of parameters may be harder to interpret or explain. During evaluation, it's important to consider not just predictive performance but also the interpretability of the model.

Model Compression:

- Evaluation might involve exploring techniques like model pruning or compression to reduce the effective number of parameters.
- Compression methods aim to simplify the model without sacrificing performance, making it more efficient for deployment.

Trade-off Analysis:

- Evaluating the effective number of parameters is often part of a broader trade-off analysis between model complexity and performance.
- This analysis helps in selecting models that achieve a good balance between fitting the training data and generalizing to new data.

Cross-Validation Considerations

- During cross-validation, understanding the effective number of parameters helps in assessing the consistency of model performance across different folds and ensures that the model is not overfitting to a

specific subset of the data.

Cross Validation

This method directly estimates the expected extra-sample error Err = E[L(Y, ˆf(X))], the average generalization error when the method ˆf(X) is applied to an independent test sample from the joint distribution of X and Y.

The basic idea is to split your dataset into multiple subsets or folds. The model is trained on some of these folds and tested on the remaining fold. This process is repeated multiple times,

with different folds used as the test set in each iteration. The most common form is k-fold cross-validation.

K-Fold Cross-Validation:

- Split your dataset into k equal-sized folds.
- Train the model on k-1 folds and test it on the remaining fold.
- Repeat this process k times, each time using a different fold as the test set.
- Average the performance metrics from all iterations to get a more robust evaluation.

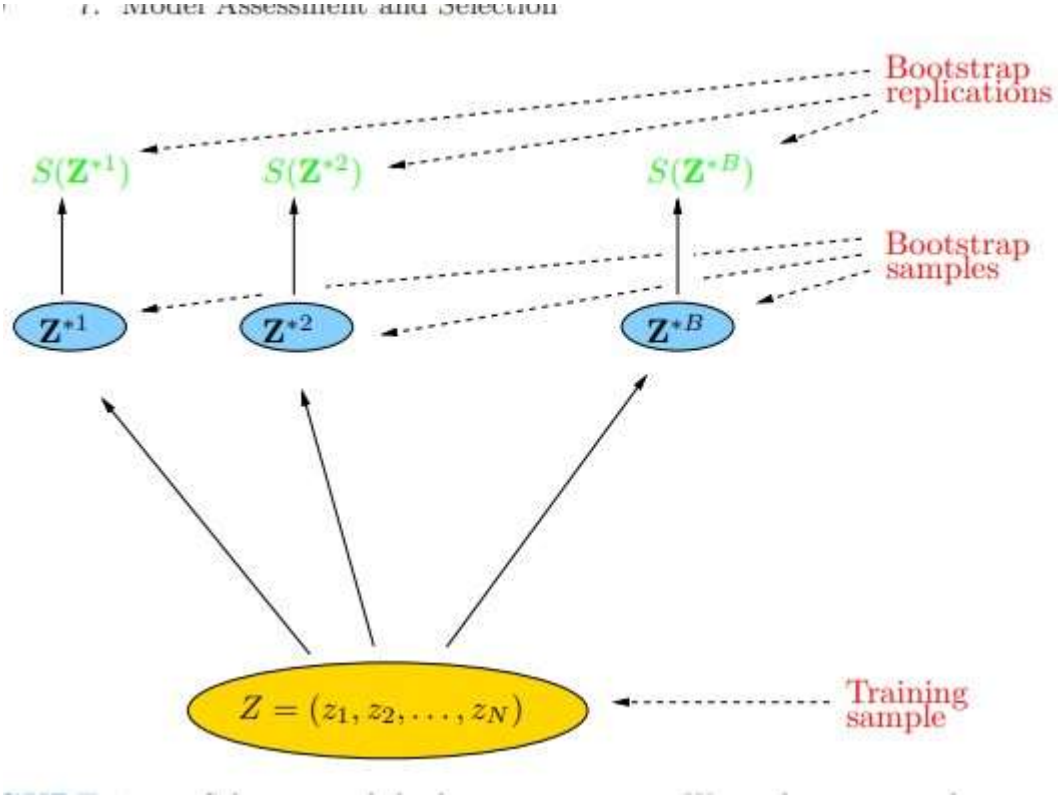
K-fold cross-validation provides a more reliable estimate of a model's performance compared to a single train-test split. It helps to ensure that your model's performance is consistent across different subsets of the data.
the cross-validation estimate of prediction error is

$$\mathrm{CV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i)).$$

BootStrapping

Bootstrapping is a resampling technique that involves generating multiple subsets (samples) from a single dataset by randomly selecting data points with replacement. Each bootstrap sample is of the same size as the original dataset, but it's likely to contain some duplicate data points.

The bootstrap is a general tool for assessing statistical accuracy. First we describe the bootstrap in general, and then show how it can be used to estimate extra-sample prediction error. As with cross-validation, the bootstrap seeks to estimate the conditional error ErrT , but typically estimates well only the expected prediction error Err.



Steps in Bootstrapping:

- Randomly select data points from the original dataset with replacement to form a new bootstrap sample.
- Repeat this process to create multiple bootstrap samples.
- Perform the analysis (e.g., train a model) on each bootstrap sample.
- Aggregate the results (e.g., average predictions) over all bootstrap samples.

.

$$\widehat{\mathrm{Err}}_{\mathrm{boot}} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L(y_i, \hat{f}^{*b}(x_i)).$$

However, it is easy to see that Err boot does not provide a good estimate in general. The reason is that the bootstrap datasets are acting as the training samples, while the original training set is acting as the test sample, and these two samples have observations in common. This overlap can make overfit predictions look unrealistically good, and is the reason that crossvalidation explicitly uses non-overlapping data for the training and test samples.

Conditional or Expected Test Error?

The conditional or expected test error is like the crystal ball of machine learning—it gives you a glimpse into how well your model is likely to perform on new, unseen data. Essentially, it's the anticipated error your model will make on future predictions.

Imagine you're training a model to identify cats and dogs. During training, your model gets really good at recognizing the particular cats and dogs in your dataset. But the true test comes when you unleash it on a new set of pictures with different cats and dogs. The conditional test error tells you how well your model is expected to generalize to these fresh examples.

It's like predicting the future performance of your model. The lower the conditional test error, the better your model is at handling unseen data. So, when you hear someone talking about minimizing the expected test error, they're essentially fine-tuning their crystal ball to make better predictions in the real world!
We conclude that estimation of test error for a particular training set is not easy in general, given just the data from that same training set. Instead, cross-validation and related methods may provide reasonable estimates of the expected error Err.