Course II - Supervised ML (Regression)

2.1 Introduction to Supervised Machine Learning

- **Machine Learning:** Allows computers to learn from data without being explicitly programmed. It involves approximating equations or functions from data to make predictions or decisions.
- **Function Approximation:** In machine learning, the focus is on approximating functions from data to enable prediction of future values.
- **Traditional Statistical Modeling vs. Machine Learning:** In traditional statistical modeling, we often know about the underlying process and choose models accordingly. In machine learning, we learn or approximate equations from data, especially when the underlying process is complex or unknown.

Artificial Intelligence Framework:

- Russell and Norvig's framework divides artificial intelligence into four quadrants based on whether machines can think or act and whether they replicate human or rational behavior.
- Machine learning focuses on the thought processes, whether human-like or rational, to learn from data and make decisions or predictions.
- Learning is a fundamental aspect of artificial intelligence and machine learning, enabling machines to make decisions, solve problems, perceive, reason, and act accordingly.

Models in Machine Learning:

- A model is a simplified representation that captures important features or relationships of a larger phenomenon.
- A good model omits unimportant details while retaining essential relationships, reducing the complexity of the real-world phenomenon.

Applications of Machine Learning:

- Machine learning has numerous applications in our daily lives, including spam filtering, web search ranking, route optimization, fraud detection, movie recommendations, and more.
- These applications demonstrate the pervasiveness of machine learning in various domains and its continuous advancement.

Parameters vs. Hyperparameters in Supervised Machine Learning:

Understanding Fit Parameters:

- **Fit Parameters:** These are aspects of the model estimated using the data during the training process. Examples include coefficients in linear regression models.

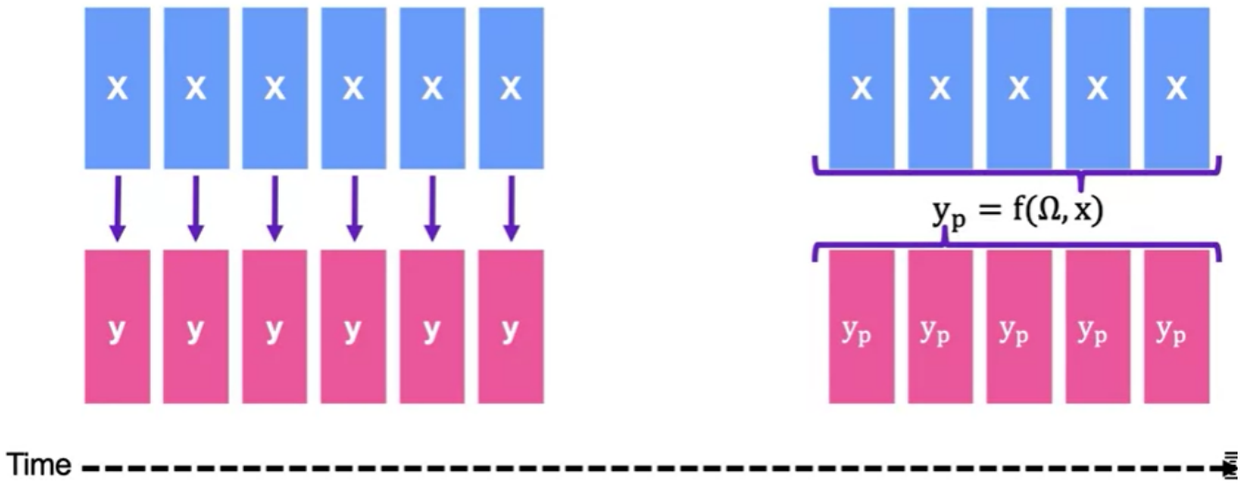
Introduction to Hyperparameters:

- **Model Decisions:** Before fitting the model to the data, decisions about the model itself need to be made. These decisions lead to the concept of hyperparameters.
- **Hyperparameters:** These are parameters that are not directly learned by the model but are instead set by the data scientist before the training process begins. Tweaking hyperparameters can optimize the model's performance.

Categories in Supervised Learning:

- **Regression:** Predicting numeric values, such as stock prices or coordinates of a location.
- **Classification:** Predicting categorical outcomes, such as face recognition or customer churn.

Supervised Machine Learning Framework:



- The supervised machine learning framework involves input X, predicted output $\hat{y}$ given a particular$X$, and a machine learning model that generates predictions by learning appropriate parameters Ω.

Learning Parameters from Data:

- Data scientists train the model to find the best parameters by examining past data, where each observation X relates to an outcome variable y.
- Training the model on past data allows it to learn the parameters that define the relationship between the features in X and the outcome variables y

Updating Parameters with New Data:

- Once trained on past data, the model can use past experience to make predictions for new observations.
- It's essential to avoid overfitting by training on a subset of old data and optimizing results on a holdout set before deploying the model into production.

Loss Function and Model Optimization:

- **Loss Function $J(y, \hat{y})$):** Defines a quantitative score for how good the predictions are compared to the actual values y.
- The update rule determines how to update the parameters to minimize the loss function, aiming to reduce the error between the predicted values and the actual values.

Interpretation vs. Prediction in Supervised Machine Learning:

Interpretation as the Primary Objective:

- **Focus:** Training the model to gain insights about the data.
- **Parameters:** Insight into the system is sought through understanding model parameters.
- **Workflow:** Gather X and Y data, train the model to learn parameters, prioritize understanding parameters over prediction accuracy.
- **Model Complexity:** Preference for less complex models with high interpretability.

- **Examples:** Understanding customer demographics' impact on sales, determining effective safety features to prevent accidents, assessing marketing budget's effect on movie revenue.

Prediction as the Primary Objective:

- **Focus:** Emphasis on the accuracy of predictions compared to actual values.
- **Performance Metrics:** Evaluation based on performance metrics measuring the closeness of predicted values to actual values.
- **Interpretability:** Less concern for interpretability, especially in complex models like deep learning.
- **Examples:** Predicting customer churn likelihood, forecasting default likelihood, predicting future purchases based on past history.

Trade-offs and Considerations:

- **Trade-off:** There's often a trade-off between interpretability and prediction accuracy.
- **Business Objective:** Decision-making depends on the specific business objective, weighing the importance of interpretability against prediction accuracy.
- **Ideal Scenario:** While interpretability and prediction accuracy are both desired, practical constraints may necessitate prioritizing one over the other based on the business context.

Regression Example (Movie Revenue Prediction):

- Data: Movie data with known revenue (features like marketing budget, cast budget).
- Model Fitting: Learn parameters to predict movie revenue based on features.
- Prediction: Use the fitted model to predict revenue for new, unlabeled movies.

Classification Example (Spam Email Detection):

- Data: Labeled emails (spam vs. not spam) with features (words).
- Model Fitting: Learn parameters to distinguish spam from non-spam based on word features.
- Prediction: Use the fitted model to classify new, unlabeled emails as spam or not spam.

Requirements for Classification:

- Quantifiable features (e.g., words encoded using one-hot encoding).
- Known labels for training data (e.g., human-labeled dataset).
- Similarity metric to compare new records with trained data.

2.2 Linear Regression

- Linear regression aims to predict a continuous outcome variable (e.g., box office revenue) using one or more input variables (e.g., marketing budget).
- The model is represented by a line equation: $Y = \beta_0 + \beta_1 X$, where:
  - β_0 is the intercept (predicted value when $X = 0$,
  - β_1 is the slope (effect of one unit increase in X on Y,
  - X is the input variable.

Model Fitting:

- The goal is to minimize the cost function, typically the mean squared error, by adjusting the parameters β_0 and β_1.

- The best-fit line minimizes the distance between predicted values and actual data points.

Interpreting Coefficients:

- β_0 represents the predicted value when $X = 0$.
- β_1 represents the change in Y for a one-unit change in X.

Measuring Errors:

- Error is calculated as the difference between predicted and actual values.
- Common error metrics include the mean absolute error (L1 norm) and the Euclidean distance (L2 norm).
- The mean squared error (MSE) is the average squared error, commonly used in linear regression.
- The cost function minimizes the squared error to optimize parameter values.

Best Practices in Modeling:

1. **Define Cost Function:** Establish the cost function to be minimized, providing a method for comparing different models' strengths.
2. **Develop Multiple Models:** Create various models with different hyperparameters or entirely different algorithms to determine the best predictions.
3. **Compare Results:** Evaluate model performance using the chosen cost function and select the best-performing model.

R-squared Metric:

- **Sum of Squared Error (SSE):** Measures the distance between predicted and actual values.
- **Total Squared Error (TSE):** Measures the distance between actual values and their mean.
- **Explained Variation:** The portion of total variation explained by the model.
- **R-squared:** Proportion of explained variation to total variation ($1 - \frac{SSE}{TSE}$)
  - Represents how well the model explains the variance in the data.
  - Higher values indicate better fit, closer to 1.

Considerations:

- Adding complexity to a model may improve fit to every data point but doesn't necessarily improve predictive power.
- R-squared can be used for any regression model to understand explained variance.

Implementation in Python (using sklearn):

1. **Import LinearRegression class:** `from sklearn.linear_model import LinearRegression`
2. **Create Model Instance:** `LR = LinearRegression()`
3. **Fit Model:** `LR.fit(X_train, y_train)`
4. **Make Predictions:** `predictions = LR.predict(X_test)`

2.3 Data Splits and Polynomial Regression

Importance of Data Splitting:

- Splitting data into training and testing sets allows for model evaluation on unseen data.

- Helps assess model generalization and performance in real-world scenarios.
- Prevents overfitting to training data and ensures model robustness.

Data Leakage Concerns:

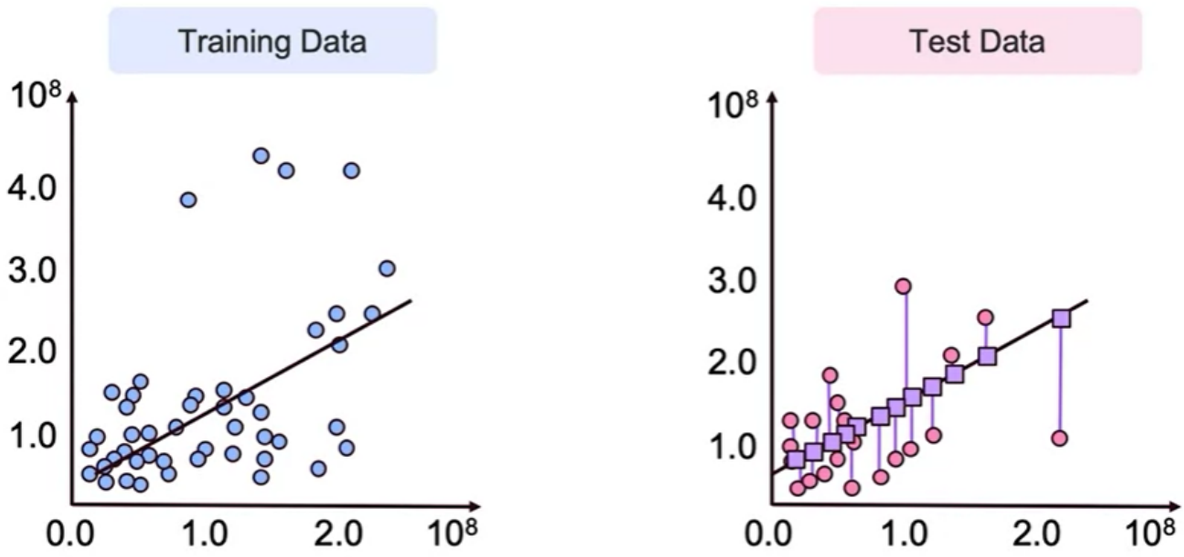
- Ensure independence between training and testing sets to avoid data leakage.
- Data leakage occurs when test data inadvertently influences the training process, leading to biased model performance.

Process:

1. **Training Data:** Used to fit the model and learn optimal parameters.
2. **Testing Data:** Unseen data used to evaluate model performance.
3. **Model Evaluation:** Predict values on the test data and compare with actual values to measure error.

Scenario Illustration:

- **Training Set:** Used to fit a regression line to explain the data.
- **Testing Set:** Unseen data where predictions are made using parameters from the training set.
- **Evaluation:** Measure the distance between actual and predicted values to assess model performance on unseen data.



Working with Training and Test Data:

1. **Training Data (X_train, Y_train):**
   - **Features (X_train):** Input variables used for model training.
   - **Outcome Variable (Y_train):** Target variable to be predicted.
   - **Model Fitting:** Fit a model to learn parameters using `fit()` method.
2. **Test Data (X_test):**
   - **Features (X_test):** Unseen data for evaluating model performance.
   - **Prediction:** Use the trained model to predict outcomes for X_test.
3. **Evaluation:**
   - **Comparison:** Compare predicted outcomes (Y_predict) with actual outcomes (Y_test).
   - **Error Metric:** Calculate error metric (e.g., Mean Squared Error) to assess model performance on test data.

Syntax for Train-Test Split in Python:

```
from sklearn.model_selection import train_test_split

# Split data into train and test sets
X_train, X_test, Y_train, Y_test = train_test_split(data, test_size=0.3)
```

- Specify the percentage of data to be allocated for the test set using the `test_size` parameter.
- Can also pass numerical values (e.g., 0.3) or other values to specify exact test size.
- If splitting features and outcome variable, provide X and Y as separate arguments.

Other Methods for Splitting Data:

- **ShuffleSplit:** Generate multiple random splits of data for cross-validation.
- **StratifiedShuffleSplit:** Ensure balanced representation of classes in train and test sets, useful for imbalanced datasets (e.g., medical diagnosis).

Polynomial Regression:

Pre-processing Trick: Creating New Features:

- Besides standard scaling, consider generating new features from existing ones.
- Example: Adding squared or cubed features to capture nonlinear relationships.

Functional Form and Feature Selection:

- Choose the correct functional form by evaluating correlations between manufactured features and the outcome variable.
- Explore all possible feature combinations and perform feature selection techniques to optimize model performance.

Balancing Prediction and Interpretation:

- Polynomial features improve prediction by capturing the curvature of data.
- Enhance interpretation by incorporating interaction terms, providing insights into coefficient relationships.

Framework for Model Evaluation:

- Apply the same framework across regression and classification problems.
- Focus on finding the balance between model complexity and generalization (bias-variance trade-off).

Utilizing Polynomial Features in Python:

- Import `PolynomialFeatures` from `sklearn.preprocessing`.
- Create an instance of the class with desired degree of polynomial features.
- Fit the object to the data and transform the features to obtain polynomial features.

Equations of Regression Models:

1. **Linear Regression Model:** $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \varepsilon$
2. **Polynomial Regression Model:** $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \ldots + \beta_n X^n + \varepsilon$
3. **Polynomial Features Transformation:** $\text{PolynomialFeatures}(X) = [1, X, X^2, X^3, \ldots, X^n]$

2.4 Cross-Validation and Bias-Variance Tradeoff

Importance of Data Splitting:

- Splitting data into training and test sets enables evaluating model performance on unseen data, crucial for real-world applicability.
- Avoid data leakage by ensuring independence between training and test sets.

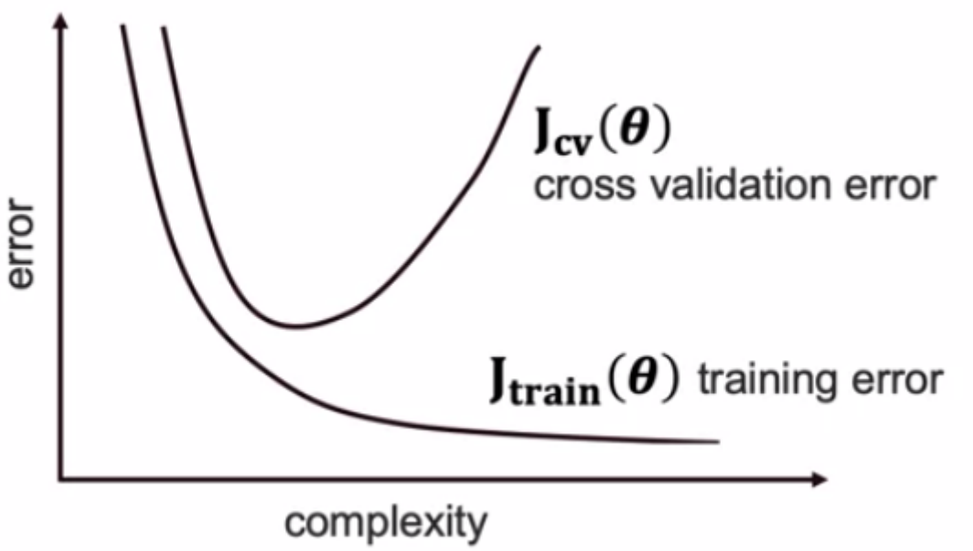
Process of Data Splitting:

1. **Training Data:** Consists of features (X_train) and the corresponding outcome variable (Y_train).
   - Fit a model to learn parameters using `fit()` method, typically in `scikit-learn`.
2. **Test Data:** Utilize features (X_test) from the test set to make predictions.
   - Predict outcome variable (Y_pred) using the trained model.
   - Compare predicted values with actual values to evaluate model performance using error metrics.

Syntax for Data Splitting in Python:

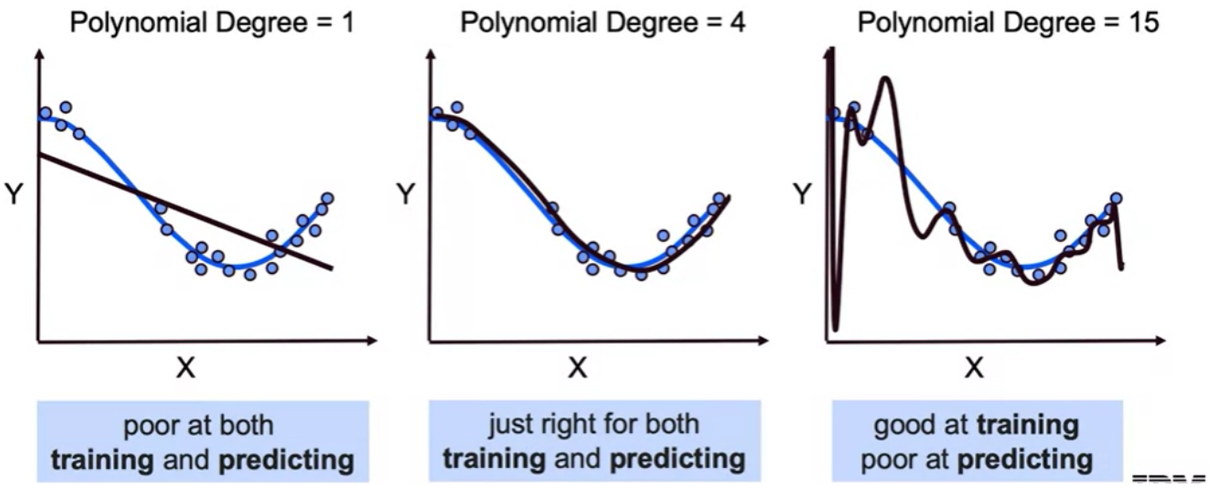
```
from sklearn.model_selection import train_test_split

# Splitting data into train and test sets (70-30 split)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
```

- `test_size` parameter specifies the proportion of data allocated to the test set.
- `X` represents features, and `Y` denotes the outcome variable.
- The output includes four sets: `X_train`, `X_test`, `Y_train`, and `Y_test`.
- **Model Complexity vs. Error:**
  - Ideal scenario: Small errors in both training and test data, indicating a good fit.
  - Too complex model: Small training error but large test error, indicating overfitting.
  - Too simple model: Large errors in both training and test data, indicating underfitting.
  - Regularization is a technique to handle overfitting.

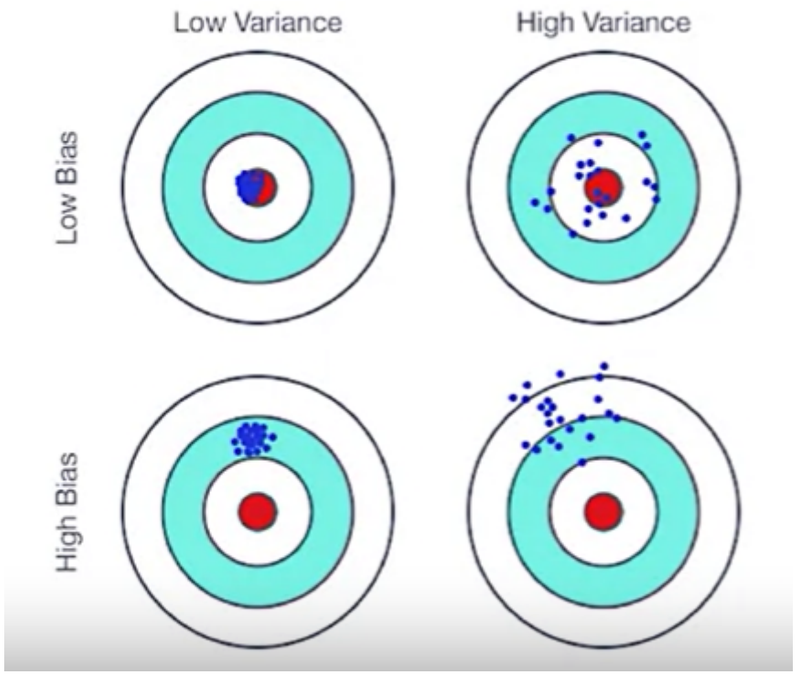


- **Choosing Model Complexity:**
  - Example: Fitting a polynomial function to data points.
  - Error decreases as model complexity increases, but this may deviate from the true underlying function.

- Critical to consider how well the model generalizes beyond the training data.

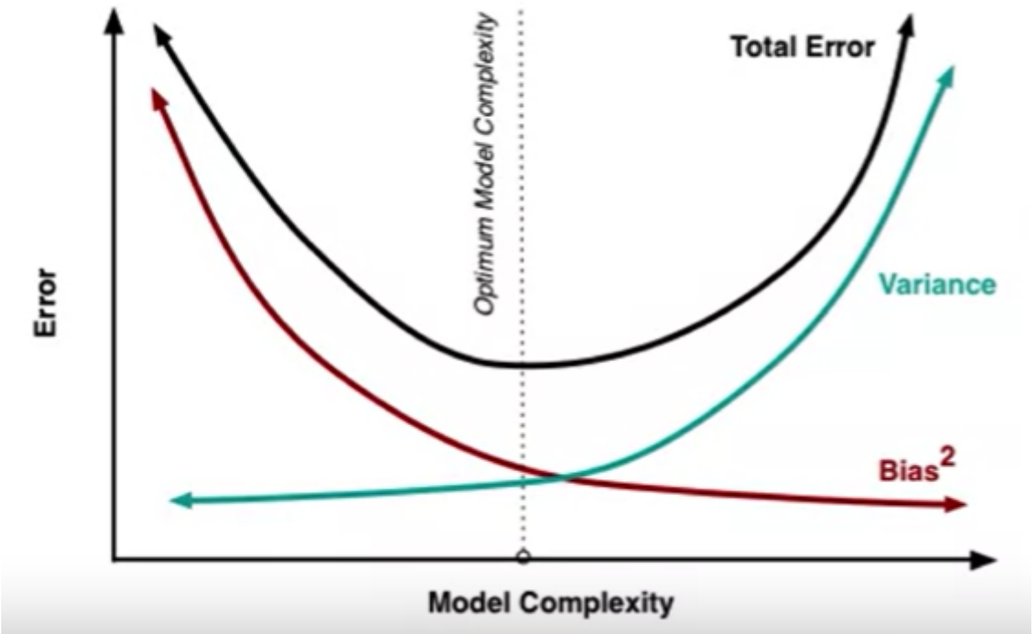


- **Bias and Variance:**
  - Bias: Tendency to consistently miss the target.
    - High bias, low variance: Consistently missing the target.
  - Variance: Tendency to be inconsistent.
    - Low bias, high variance: Inconsistent predictions, potentially overfitting.
  - Ideal: Low bias, low variance, consistent and accurate predictions.



- **Sources of Model Error:**
  - Incorrect modeling: Not capturing the true relationship between features and outcomes (biased model).
  - Instability: High variance due to overfitting, incorporating noise in the data.
  - Unavoidable randomness: Real-world data inherently includes randomness that cannot be perfectly predicted.
- **High Bias (Underfitting):**
  - Occurs when the model is too simplistic or lacks sufficient complexity.
  - Misses real patterns in the data.
  - Associated with underfitting or bias towards simplicity.
- **High Variance (Overfitting):**
  - Predictions fluctuate dramatically, sensitive to small changes in input.
  - Occurs when the model is too complex and captures noise in the data.

- Associated with overfitting or bias towards complexity.



2.5 Regularization

- Regularization modifies the cost function of the model to penalize complexity.
- Regularization parameter (lambda) controls the strength of the penalty.
- Adding regularization reduces the complexity of the model, preventing overfitting.
- **Regularization Techniques:**
  - Ridge and Lasso regression: Methods to add regularization to linear regression models.
  - Elastic Net: Balances Ridge and Lasso by combining their penalties.
  - Recursive feature elimination: Technique to eliminate irrelevant features, improving model generalization and interpretability.
- **Role of Regularization in Feature Selection:**
  - Regularization serves as a form of feature selection by reducing the contribution of less important features.
  - Lasso regression can drive coefficients to zero, effectively removing irrelevant features.
  - Feature elimination can be quantitatively performed to improve model performance and interpretability.
- **Benefits of Feature Elimination:**
  - Reduces overfitting and improves fitting time for models without built-in regularization.
  - Identifies the most important features, enhancing model interpretability and meeting business requirements.
- **Ridge Regression:**
  - Ridge regression adds a penalty term to the original linear regression cost function to prevent overfitting.
  - The penalty term is a function of the square of the coefficients, multiplied by a regularization parameter λ.
  - The penalty term penalizes larger coefficients more strongly, leading to reduced complexity.
  - Scaling of features becomes crucial as the penalty term is affected by the scale of coefficients.
  - Ridge regression imposes bias on the model but reduces variance, striking a balance between complexity and error.

    - The penalty term proportional to the square of coefficients shrinks them towards zero, reducing complexity.
    - Lambda controls the strength of regularization; cross-validation helps in selecting the optimal lambda.
    - Standardization of features ensures that penalties aren't influenced by variable scales.
- **Effect of Lambda on Coefficients:**
    - Larger lambda values result in lower coefficients, as they increase the penalty for larger weights.
    - The relationship between lambda and standardized coefficients typically shows a decrease in coefficients as lambda increases.
    - The complexity trade-off in Ridge regression aims to reduce variance without significantly increasing bias.
    - There may not be a linear trade-off between reducing complexity and increasing bias; optimal lambda minimizes mean squared error on the holdout set.
- **Application to Polynomial Graphs:**
    - Different lambda values affect the complexity of polynomial models differently.
    - Higher lambda values lead to simpler models with lower coefficients, reducing overfitting.
    - The goal is to find the optimal lambda that minimizes error on the holdout set while balancing bias and variance.
- **Equations for Ridge Regression:**

1. **Original Linear Regression Cost Function**:
   $\text{Cost}(w) = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2$
2. **Ridge Regression Cost Function**:
   $\text{Cost}(w) = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} w_j^2$
    - λ: Regularization parameter (controls the strength of regularization).
    - w_j: Coefficients of the model.
    - m: Number of training examples.
    - n: Number of features.
3. **Feature Standardization**:
   $x'_j = \frac{x_j - \mu_j}{\sigma_j}$
    - x'_j: Standardized feature.
    - x_j: Original feature.
    - μ_j: Mean of the feature x_j.
    - σ_j: Standard deviation of the feature x_j.

LASSO Regression:

LASSO (Least Absolute Shrinkage and Selection Operator) regression is another regularization technique used in linear regression models to prevent overfitting. Unlike Ridge regression, which penalizes the square of the coefficients (L2 norm), LASSO penalizes the absolute value of the coefficients (L1 norm). This difference in penalty terms affects the behavior of the model, particularly in terms of feature selection.

Comparison with Ridge Regression:

1. **Penalty Term**:
    - Ridge: Proportional to the square of the coefficients.
    - LASSO: Proportional to the absolute value of the coefficients.

2. **Outlier Sensitivity**:
    - LASSO is less sensitive to outliers compared to Ridge regression due to its use of the absolute value of coefficients.
3. **Feature Selection**:
    - LASSO is more likely than Ridge to perform feature selection by completely zeroing out certain coefficients. This is because the L1 penalty has a sparsity-inducing property, driving some coefficients to exactly zero.

Equations:

1. **LASSO Regression Cost Function**:

$\text{Cost}(w) = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |w_j|$

2. **Feature Standardization**: Same as in Ridge regression.
3. **Relationship between Lambda and Coefficients**:
    - As (\lambda) increases, coefficients tend to decrease in magnitude.
    - LASSO is more likely to drive some coefficients exactly to zero compared to Ridge.

Effects of Lambda Values:

- **High Lambda**: Strong regularization, many coefficients driven to zero, high bias, low variance.
- **Low Lambda**: Weak regularization, less feature selection, lower bias, higher variance.
- **Optimal Lambda**: Balanced trade-off between bias and variance, determined through cross-validation.

Elastic Net Regression:

Elastic Net is a hybrid regularization technique that combines the penalties of both Ridge and LASSO regression. It introduces a mixing parameter ((\alpha)) that determines the balance between L1 and L2 penalties. Elastic Net offers a compromise between the feature selection capabilities of LASSO and the computational efficiency of Ridge.

Equations:

1. **Elastic Net Cost Function**:
   $\text{Cost}(w) = \frac{1}{2m} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2 + \lambda_1 \sum_{j=1}^{n} |w_j| + \lambda_2 \sum_{j=1}^{n} w_j^2$
2. **Mixing Parameter ((\alpha))**:
    - Determines the proportion of L1 and L2 penalties.
    - When $\alpha = 0$, it reduces to Ridge regression.
    - When $alpha = 1$, it reduces to LASSO regression.

Recursive Feature Elimination (RFE):

Recursive Feature Elimination (RFE) is a feature selection technique provided by scikit-learn that automates the process of selecting relevant features for a given model. It works by recursively removing less important features based on their coefficients or feature importances.

How RFE Works:

1. Choose a Model: First, select a model that supports feature importances or coefficients, such as Linear Regression, Lasso, or RandomForest.
2. Define Number of Features: Specify the number of features you want to keep in the final model.

3. Recursive Elimination: RFE will repeatedly train the model, measure feature importances or coefficients, and eliminate the least important features until the desired number of features is reached.

- The chosen model must have either coefficients or feature importances available.
- Data should be scaled before applying RFE, especially for linear regression models, to ensure coefficients are measured on the same scale.

Using RFE in scikit-learn:

1. Import the RFE class: `from sklearn.feature_selection import RFE`.
2. Initialize RFE object: `rfeMod = RFE(estimator, n_features_to_select)`, where `estimator` is the chosen model and `n_features_to_select` is the desired number of features.
3. Fit the RFE object: `rfeMod.fit(X_train, y_train)`.
4. Predict using the trained model: `predictions = rfeMod.predict(X_test)`.

Additional Class - RFECV:

- RFECV incorporates cross-validation to evaluate the impact of feature elimination on model performance. It helps in selecting the optimal number of features based on cross-validation scores.

Choosing Between Ridge, LASSO, and Elastic Net:

When deciding between Ridge, LASSO, and Elastic Net regularization techniques for linear regression, several factors should be considered:

1. Prediction Accuracy:

- Use validation techniques to assess how well each model generalizes to new data.
- Optimize hyperparameters (e.g., λ for Ridge and LASSO, α for Elastic Net) using cross-validation to achieve the best prediction accuracy on holdout sets.

2. Interpretability:

- LASSO offers the advantage of feature selection by eliminating less important features, enhancing interpretability.
- Consider the importance of understanding which features contribute most to the model's predictions.

3. Model Complexity:

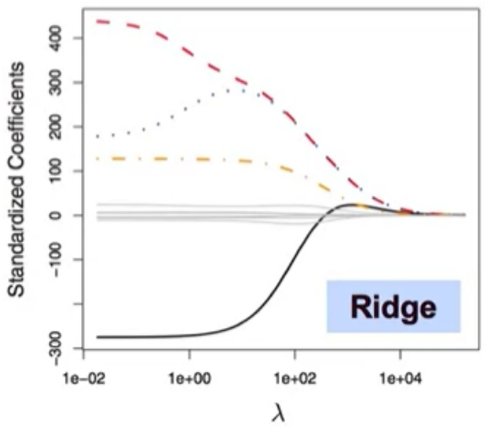
- Ridge regression tends to maintain all features in the model, potentially leading to higher complexity.
- LASSO and Elastic Net can help reduce model complexity by eliminating or shrinking less important features.

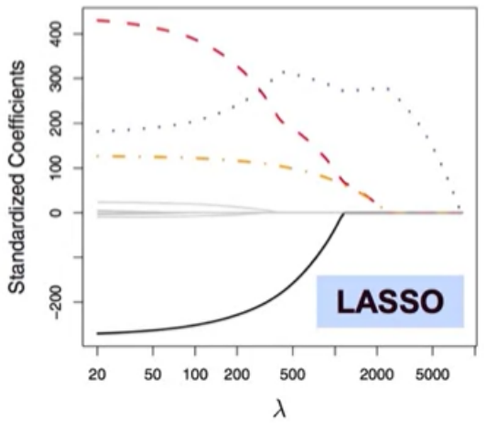
4. Computational Efficiency:

- Ridge regression is generally more computationally efficient than LASSO or Elastic Net.
- Consider the trade-off between computational resources and model performance when selecting a regularization technique.

Analytic View of Regularization:

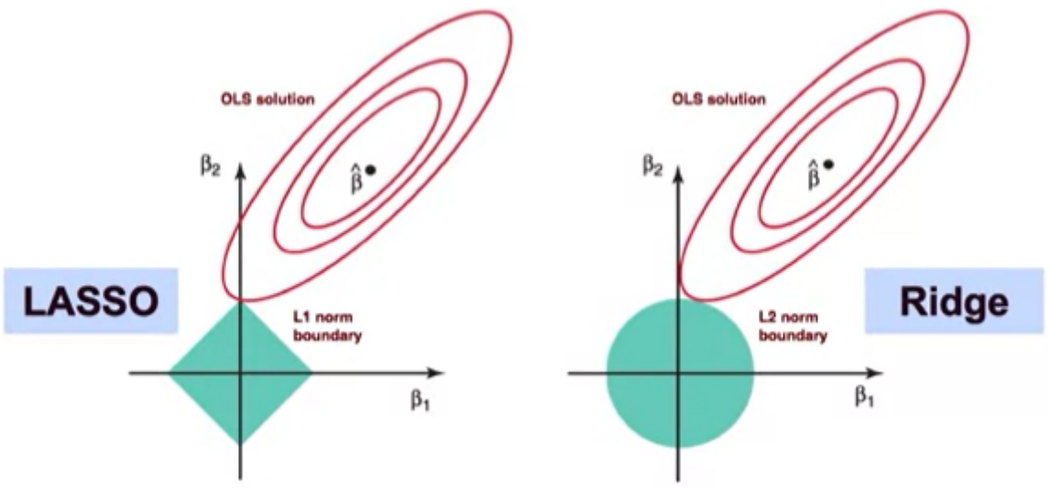
- Analytically, regularization imposes penalties (L1 or L2) on the coefficients of the model.
- By adding L2 and L1 penalties, we force the coefficients to be smaller, limiting their plausible range.
- Smaller coefficients imply a simpler model with lower variance, reducing complexity.
- Eliminating features quickly reduces variance, simplifying the model and enhancing generalization.





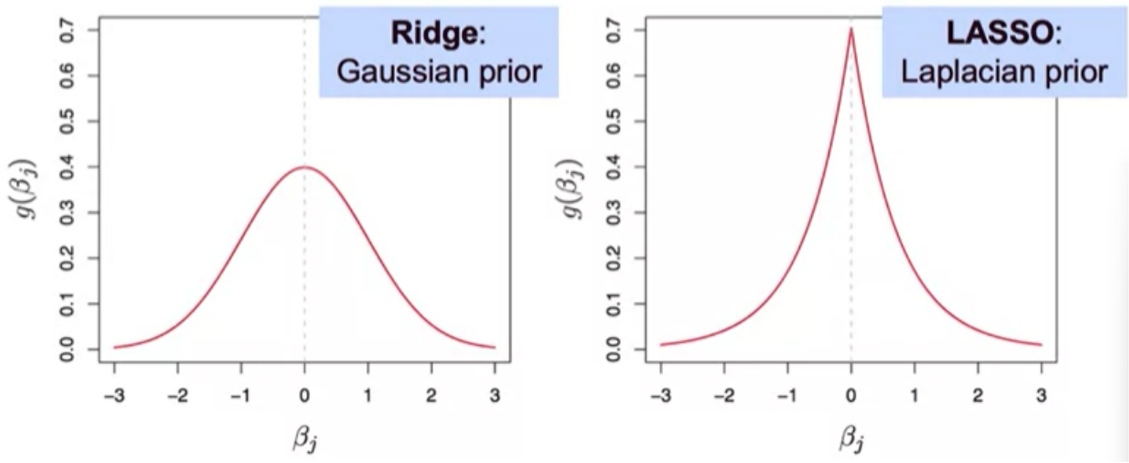
Geometric View of Regularization:

- From a geometric perspective, regularization formulations involve minimizing error subject to constraints on coefficient magnitude.
- For Ridge regression, we minimize error subject to coefficients being as small as possible (squared penalty).
- In LASSO, we minimize error subject to the absolute value of coefficients being less than a specified value.
- The intersection of the penalty boundary (diamond for LASSO, circle for Ridge) with the contour of the traditional OLS cost function determines the optimal solution.
- LASSO tends to zero out certain coefficients as its penalty shape (diamond) intersects with the contour, forcing coefficients to hit the edges and leading to feature elimination.
- In Ridge regression, the circular penalty shape allows the intersection to occur at any point, minimizing values without eliminating coefficients.



Probabilistic View of Regularization:

- The probabilistic view reframes regularization by considering coefficients as having particular prior distributions.
- Ridge and LASSO can be interpreted as adjusting the posterior distribution of coefficients based on their prior distributions and the observed data.
- Regularization techniques act as a form of Bayesian inference, updating our belief about coefficients' magnitudes.



By understanding regularization from these different perspectives, we gain insights into its role in controlling model complexity, improving generalization, and enhancing interpretability. These views provide a deeper understanding of regularization approaches and demystify their workings, making them less opaque and more accessible to practitioners.

Understanding Regularization from a Probabilistic Perspective

Regularization imposes prior distributions on regression coefficients, recalibrating our understanding of the coefficients' values before establishing the relationship between predictors (X) and the outcome variable (Y). By finding optimal coefficients given our data, we essentially update the posterior distribution of coefficients based on their prior distributions.

In Bayesian terms, this process can be expressed as the probability of Y given β (coefficients) multiplied by our prior understanding of what Beta should be. We recalibrate this probability using Bayes' formula to find the probability of obtaining our target given a set of coefficients multiplied by our prior distribution of Beta.

The choice of regularization method (Ridge or LASSO) determines the form of the prior distribution imposed on coefficients. For Ridge regression, a Gaussian prior is assumed, while for LASSO, a Laplacian prior is imposed. These distributions shape our prior belief about the coefficients' values.

Probabilistic View Insights:

- Regularization adjusts the posterior distribution of coefficients based on prior distributions and observed data.
- Ridge imposes a Gaussian prior, while LASSO imposes a Laplacian prior, influencing the coefficients' behavior differently.
- LASSO's sharper and higher peak at zero indicates a higher likelihood of zeroing out coefficients, leading to feature selection.
- The Lambda parameter in regularization controls the variance of the prior distribution, affecting the magnitude of coefficients. A higher Lambda shrinks the variance, leading to smaller coefficients.

Probabilistic View Equations:

Regularization adjusts the posterior distribution of coefficients based on prior distributions and observed data. In Bayesian terms, this process can be expressed as follows:

1. **Bayes' Formula:**

   $$P(\text{coefficients}|\text{data}) = \frac{P(\text{data}|\text{coefficients}) \times P(\text{coefficients})}{P(\text{data})}$$

   Where:

- $P(\text{coefficients}|\text{data})$ is the posterior distribution of coefficients given the data.
- $P(\text{data}|\text{coefficients})$ is the likelihood of observing the data given the coefficients.
- $P(\text{coefficients})$ is the prior distribution of coefficients.
- $P(\text{data})$ is the probability of observing the data.

2. **Reformulated as Probability of Y Given β:**

   $$P(Y|\text{Beta}) = P(Y|\text{X, Beta}) \times P(\text{Beta})$$

   Where:
   - $P(Y|\text{Beta})$ is the probability of the outcome variable Y given the coefficients Beta.
   - $P(Y|\text{X, Beta})$ is the likelihood of observing Y given X and Beta.
   - $P(\text{Beta})$ is the prior distribution of coefficients.

3. **Imposing Prior Distributions:**

   - For Ridge regression, a Gaussian prior distribution is assumed:

     $$P(\text{Beta}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\text{Beta}^2}{2\sigma^2}\right)$$
   - For LASSO, a Laplacian prior distribution is imposed:

     $$P(\text{Beta}) = \frac{\lambda}{2} \exp(-\lambda|\text{Beta}|)$$

   Where:
   - λ controls the sparsity of the coefficients in LASSO.
   - σ^2 is the variance parameter in Ridge regression.

These equations demonstrate how regularization methods impose specific prior distributions on coefficients, influencing the optimization process and model behavior.

Understanding regularization from these perspectives provides insights into its mechanisms and helps in making informed choices about model complexity and generalization. Regularization is a fundamental concept in machine learning, crucial for building robust and effective models.