Regression Tutorial

Stephen Coshatt[1] and Dr. WenZhan Song[1]

The University of Georgia[1]

Sensor Data Science and AI
Spring 2025

UNIVERSITY OF
GEORGIA

Outline

Table of Contents

Regression

"Regression analysis is a set of statistical methods used for the estimation of relationships between a dependent variable and one or more independent variables. It can be utilized to assess the strength of the relationship between variables and for modeling the future relationship between them." From: Regression Analysis, by Sebastian Taylor

Regression is a type of supervised learning

Variables in Regression

1. **Dependent Variables** - a variable whose value depends on another variable or variables.
2. **Independent Variables** - a variable whose value does not depend on another variable.

Types of Independent Variables

1. **Explanatory Variables** - a variable explains an outcome or event.
2. **Predictor Variables** - a variable whose value is used to predict another variable.
3. **Experimental Variables** - a variable whose value can be changed in an experiment to asses its impact.
4. **Subject Variables** - a variable that vary across a dataset but cannot be manipulated.

Cross-Validation

1. Cross-validation is a technique used to assess the performance and generalization ability of machine learning models.
2. It involves partitioning the dataset into multiple subsets or "folds" and training/testing the model multiple times using different folds as the training and testing sets.
3. The goal is to ensure that the model is evaluated on various parts of the data and isn't just overfitting to a particular subset.

Table of Contents

Generalized Linear Regression Models

Generalized Linear Models (GLM) are group of models that are based on the basic linear model of regression.

All linear regression models use a loss function to determine the best model. Basic linear regression models use Ordinary Least Squares (OLS). Others more sophisticated models add other terms to the loss function. All models in this section use OLS.

GLMs have three components:

1. **Linear predictor** - a linear combination of parameter **b** and explanatory value **x**
2. **Link function** - a function that links the linear predictor and the parameter for a specific probability distribution
3. **Probability distribution** - The data distribution that represents the observed variable **y**.

GLM Example

Example GLM using the **logit function** as the link function and a **Bernoulli distribution** as the probability distribution, which is called logistic regression:

$$z_i = b_0 + b_1 x_i$$

$$q_i = \frac{1}{1 + \exp(-z_i)}$$

$$y_i \sim \text{Bern}(q_i)$$

Figure: Logistic Regression

Fundamental Assumptions

Linear regression analysis is based on six fundamental assumptions:

1. Dependent and independent variables have a linear relationship between the slope and the intercept
2. The independent variable is not random
3. The value of the residual (error) is zero
4. The value of the residual is constant across all observations
5. The value of the residual is not correlated across all observations
6. The residual values follow the normal distribution

Reference: Regression Analysis, by Sebastian Taylor

Normal Distribution

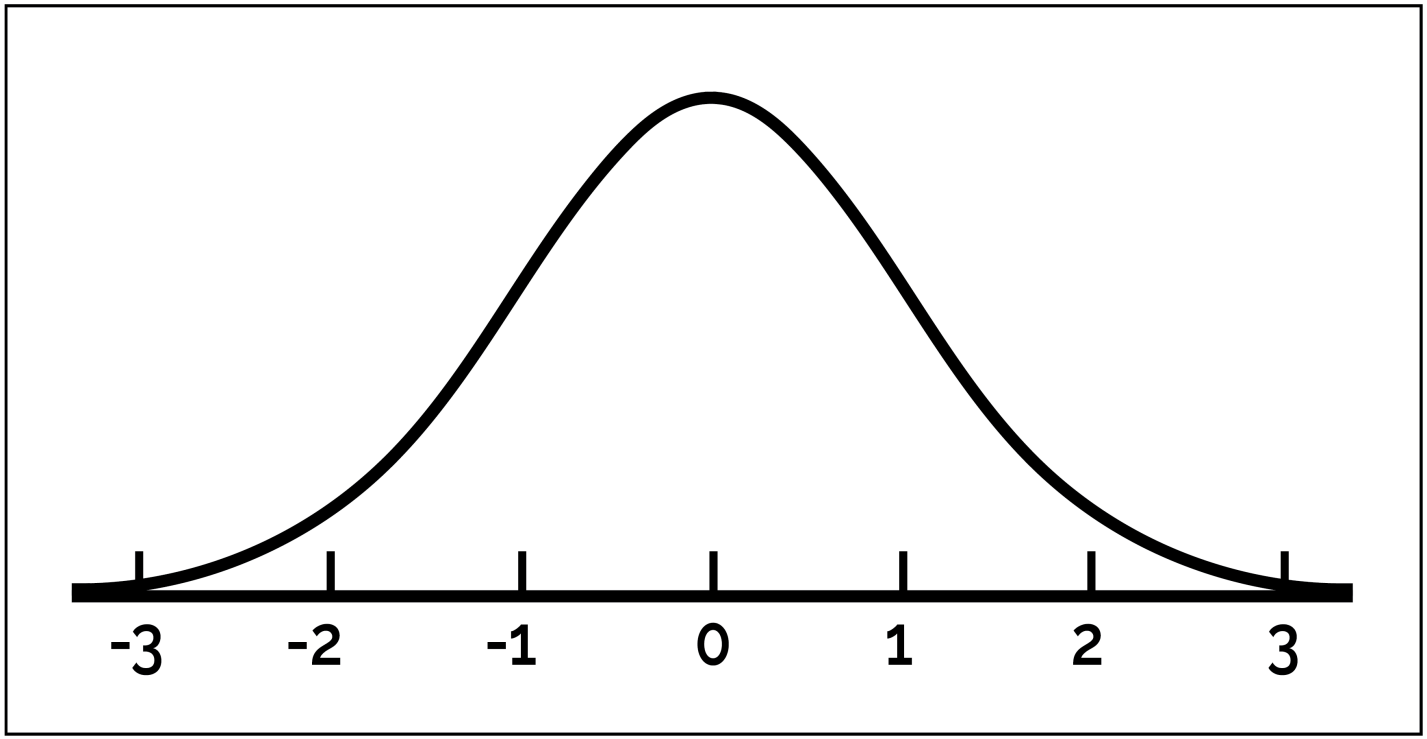


Figure: Normal Distribution

Linear Regression

Basic linear regression uses the identity link function, which means the linear predictor and the parameter for probability distribution are identical.

$$\mu_i = b_0 + b_1 x_i$$

$$y_i \sim \mathcal{N}(\mu_i, \varepsilon)$$

Figure: Linear Regression

Linear Regression

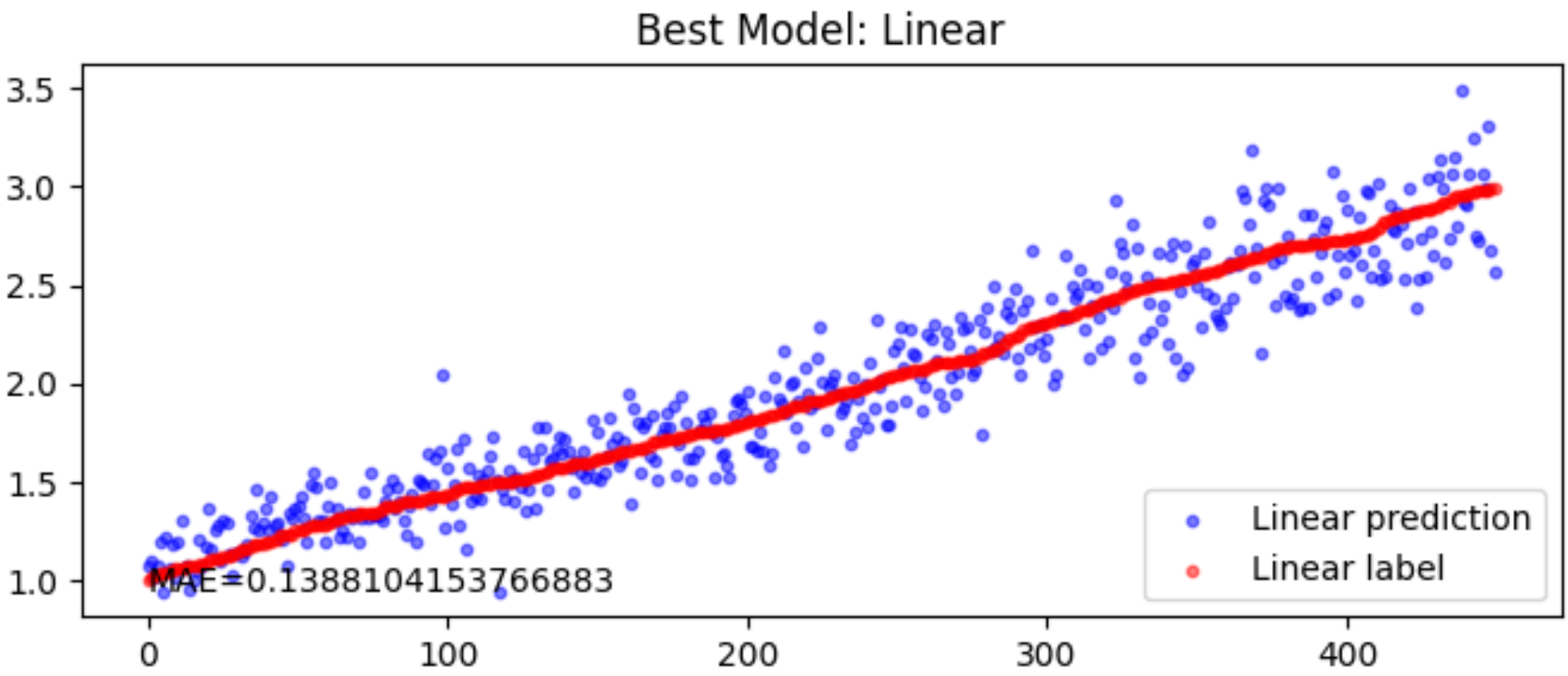


Figure: Example of Linear Regression

Gamma Regression

1. Similar to linear regression, but relies on the Gamma Distribution.
2. The gamma distribution is a right-skewed normal distribution

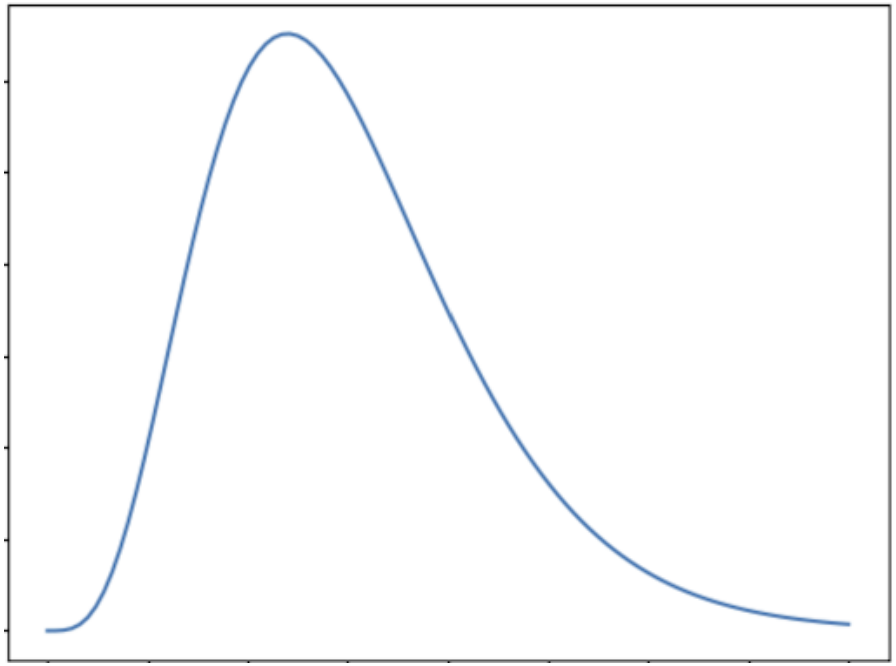


Figure: Gamma Distribution

Poisson Regression

A Poisson distribution is a probability distribution used to model the number of events that occur within a fixed interval of time or space, given that these events happen with a known constant mean rate and are independent of one another.

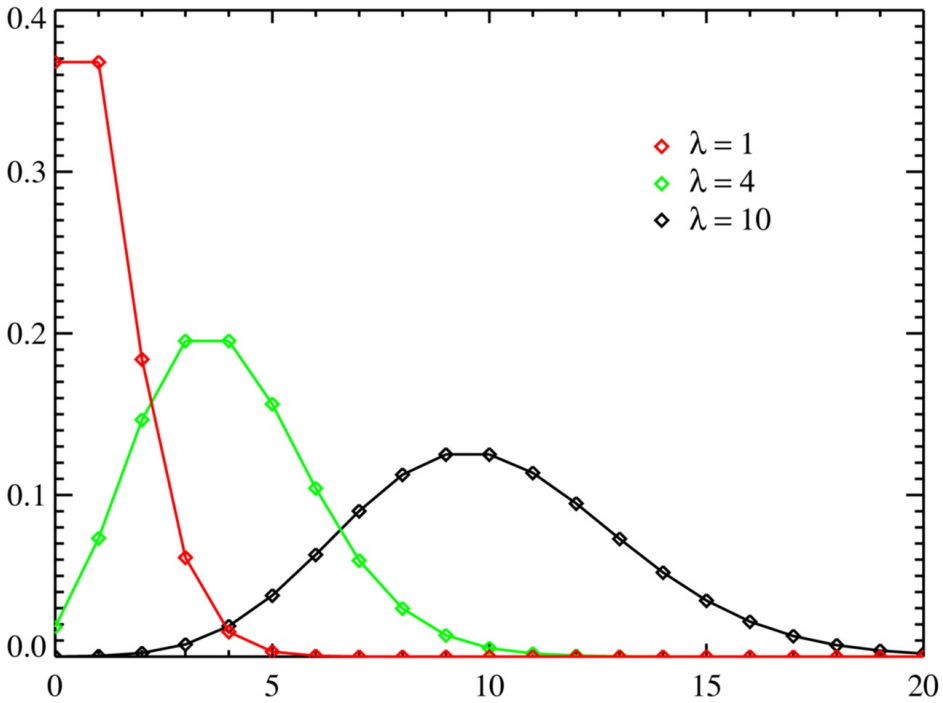


Figure: Poisson Distribution

Tweedie Regression

Tweedie distributions are special cases of other distributions. Tweedie's have a cluster of data at zero, but can otherwise follow other probability distributions.

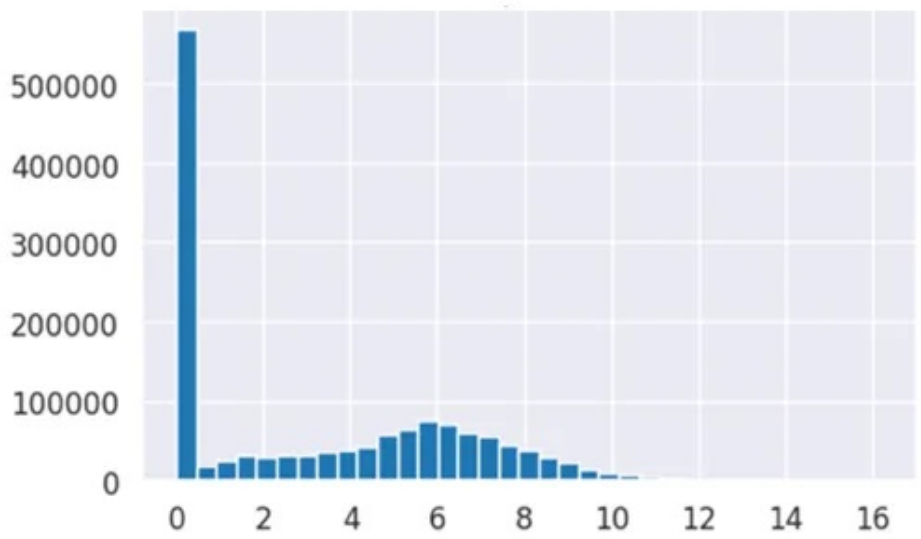


Figure: Tweedie Distribution

Reference: Tweedie Loss Function, by Sathesan Thavabalasingam

Linear Regression Pros

1. **Simple model:** Linear regression models utilize simple equations to relate features to the predicted variable.
2. **Computationally efficient:** Linear regression model's simplicity also makes the computationally efficient. They are fast and can handle large amounts of data.
3. **Interpretability of the Output:** Linear regression models are easy to interpret. Predictions based on a feature set can be easily understood. They can also easily show which features are most important.

Reference: Linear Regression -Pros & Cons, Satyavishnumolakala

Linear Regression Cons

1. **Overly-Simplistic:** Not useful for most real-world problems.
2. **Linearity Assumption:** "Linear regression makes strong assumptions that there is Predictor (independent) and Predicted (dependent) variables are linearly related which may not be the case."
3. **Severely affected by Outliers:** These models can be very sensitive to outliers, which can drastically alter predictions.
4. **Independence of variables:** Linear models assume the inputs are independent. This is rarely the case for real world data.
5. **Assumes Homoskedacity:** Linear regression assumes a constant variance around the mean, which is rare in practical problems.
6. **Inability to determine Feature importance:** Correlated features can cause issues with weights. Running the same algrothm over the same data can lead to changing feature weights.

Reference: Linear Regression -Pros & Cons, Satyavishnumolakala

Table of Contents

LARS & LASSO Regression

Least Angle Regression Shrikage (LARS) and **Least Absolute Shrinkage and Selection Operator (LASSO)** are two common types of regression. These two are discussed in this section because there is a version of regression called **LassoLars** that combines these two algorithms.

Least Angle Regression (LARS)

LARS "is an algorithm used in regression for high dimensional data (i.e., data with a large number of attributes). Least Angle Regression Shrinkage is somewhat similar to forward stepwise regression. Since it is used with data that have a lot of attributes, at each step, LARS finds the attribute which is most highly correlated to the target value. There may be more than one attribute that has the same correlation. In this scenario, LARS averages the attributes and proceeds in a direction that is at the same angle to the attributes. This is exactly why this algorithm is called Least Angle regression. Basically, LARS makes leaps in the most optimally calculated direction without over-fitting the model."

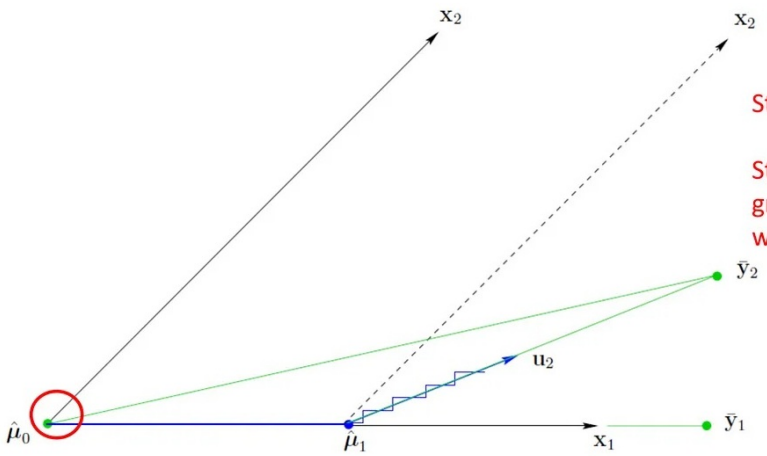
Reference: Geeks for Geeks

LARS

Algorithm:

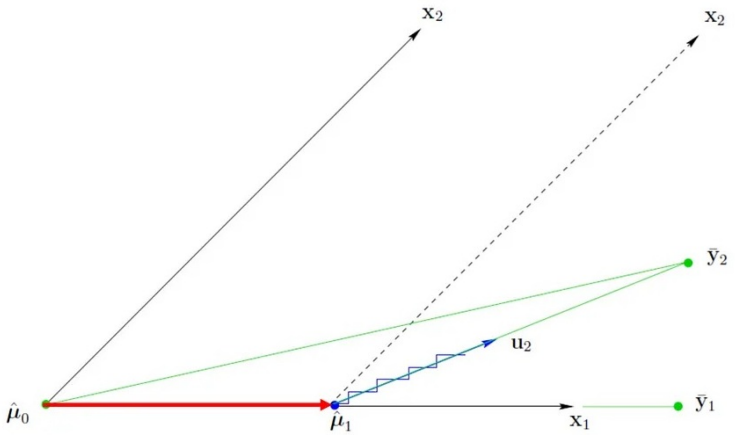
1. Normalize all values to have zero mean and unit variance.
2. Find a variable that is most highly correlated to the residual. Move the regression line in this direction until we reach another variable that has the same or higher correlation.
3. When we have two variables that have the same correlation, move the regression line at an angle that is in between (i.e., least angle between the two variables).
4. Continue this until all of our data is exhausted or until you think the model is big and 'general' enough.

Reference: Geeks for Geeks



Step 1: Start with $\hat{u}_0=0$

Step 2: The residual $\hat{y}_2-\hat{u}_0$ has a greater correlation with x_1 than with x_2



Step 3: Move in the direction of x_1

Figure: LARS

Reference: Geeks for Geeks

LARS

LARS works as follows:

1. All coefficients, **'B'** are set to **0**.
2. The predictor, x_j is found that is most correlated to y.
3. Increase the coefficient B_j in the direction that is most correlated with y and stop when you find some other predictor. x_k the has equal or higher correlation than x_j.
4. Extend (B_j, B_k) in a direction that is equiangular (has the same angle) to both x_j and x_k.
5. Continue and repeat until all predictors are in the model.

Reference: Geeks for Geeks

LASSO

LASSO regression, also known as L1 regularization, is a popular technique used in statistical modeling and machine learning to estimate the relationships between variables and make predictions.

LASSO regression attempts to balance model simplicity and accuracy, by adding a penalty term to the traditional linear regression model.

Lasso is often preferred over other regression methods for a more accurate prediction.

It utilizes shrinkage, which is where data values are shrunk towards a central point as the mean.

Reference: A Complete understanding of LASSO Regression

LASSO

1. **Linear regression model:** LASSO regression starts with the standard linear regression model, which assumes a linear relationship between the independent variables (features) and the dependent variable (target). The linear regression equation can be represented as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_x + ... + \beta_p x_p + \epsilon.$$

Where:

1. y is the dependent variable (target).
2. $\beta_0, \beta_1, \beta_2, ..., \beta_p$ are the coefficients (parameters) to be estimated.
3. $x_1, x_2, ..., x_p$ are the independent variables (features).
4. ϵ represents the error term.

Reference: A Complete understanding of LASSO Regression

LASSO

2. **L1 regularization:** LASSO regression introduces an additional penalty term based on the absolute values of the coefficients. The **L1 regularization** term is the sum of the absolute values of the coefficients multiplied by a tuning parameter $\lambda : L_1 = \lambda * (|\beta_1| + |\beta_2| + ... + |\beta_p|) = \beta_0 + \beta_1 x_1 + \beta_2 x_x + ... + \beta_p x_p + \epsilon$.

Where:

1. λ is the regularization parameter that controls the amount of regularization applied (called alpha in SK Learn).
2. $\beta_0, \beta_1, \beta_2, ..., \beta_p$ are the coefficients.

Reference: A Complete understanding of LASSO Regression

LASSO

3. **Objective function:** The objective of LASSO regression is to find the values of the coefficients that minimize the sum of the squared differences between the predicted values and the actual values, while also minimizing the L1 regularization term: Minimize: $RSS + L_1$

Where:

RSS is the residual sum of squares, which measures the

4. Shrinking coefficients: By adding the L1 regularization term, LASSO regression can shrink the coefficients towards zero. When λ is sufficiently large, some coefficients are driven to exactly zero. This property of LASSO makes it useful for feature selection, as the variables with zero coefficients are effectively removed from the model.

Reference: A Complete understanding of LASSO Regression

LASSO

5. **Tuning parameter λ:** The choice of the regularization parameter λ is crucial in LASSO regression. A larger λ value increases the amount of regularization, leading to more coefficients being pushed towards zero. Conversely, a smaller λ value reduces the regularization effect, allowing more variables to have non-zero coefficients.

6. **Model fitting:** To estimate the coefficients in LASSO regression, an optimization algorithm is used to minimize the objective function. Coordinate Descent is commonly employed, which iteratively updates each coefficient while holding the others fixed.

Reference: A Complete understanding of LASSO Regression

LassoLars

The LassoLars regressor is a Lasso model fit with Lars. It is essentially a linear model trained with an L1 prior as a regularizer.

LassoLars causes certain regression coefficients shrink to zero as a result of penalizing the absolute values of the regression coefficients.

Reference: Understanding LARS Lasso Regression

LassoLars Concepts

L1-Regularization: The linear regression objective function gains a penalty term from LassoLars depending on the absolute values of the coefficients. To encourage sparsity, this pushes some coefficients to be exactly zero.

Regularization strength (alpha): The LassoLars coefficients' intensity of penalty is determined by the regularization parameter alpha. The higher the alpha, the sparser the model.

Coefficient Path: As the regularization parameter alpha changes from 0 to 1 (or any other maximum value), LassoLars generates a path of coefficient estimates. Given that the coefficients of more significant characteristics tend to vary more gradually along the path, this provides insight on the relative value of the features.

Reference: Understanding LARS Lasso Regression

LassoLars Concepts

Forward Feature Selection: Forward feature selection means that the model adds predictors one at a time and advances in the direction of the predictor that has the highest correlation with the response at each step.

Orthogonal Active Sets: The predictors that have been added to the model are the orthogonal active set of predictors that LassoLarso keeps track of.

Reference: Understanding LARS Lasso Regression

LassoLars Concepts

1. **Efficiency:** For big datasets with plenty of characteristics, LassoLars is computationally more efficient than Lasso regression. This is because, instead of tackling a challenging optimization issue, it makes use of an effective method that repeatedly adds the most illuminating characteristic at each phase.
2. **Stability:** The LassoLars is renowned for its ability to choose features steadily. LassoLars offers a consistent feature selection procedure that is less vulnerable to fluctuations in the data, in contrast to Lasso regression, which might be sensitive to the sequence in which features are introduced to the model.
3. **Interpretability:** LassoLars's path of coefficient estimates offers important information about the relative significance of various characteristics. The characteristics that most substantially increase the model's prediction capacity may be found by tracking the coefficients' changes throughout the regularization process.

Reference: Understanding LARS Lasso Regression

LassoLars with Information-Criteria

This is a modification of the LassoLars model. It can use the **Akaiki information criterion (AIC)** or the **Bayes Information Criterion (BIC)**. Both of these criteria are used to better select the value of the regularization parameter. They make a trade-off between the goodness of fit and model complexity. In this approach, multiple models are created with differing combinations of the independent variables. Then either AIC or BIC is used to compare them to choose the best one.

Akaike information criterion (AIC)

The **Akaike information criterion (AIC)** is a mathematical method for evaluating how well a model fits the data it was generated from. In statistics, AIC is used to compare different possible models and determine which one is the best fit for the data. AIC is calculated from:

1. the number of independent variables used to build the model.
2. the maximum likelihood estimate of the model (how well the model reproduces the data).

Reference: Akaike Information Criterion - When & How to Use It (Example)

Bayesian information criterion (BIC)

The Bayesian information criterion (BIC) is a criterion for model selection among a finite set of models. It is based, in part, on the likelihood function, and it is closely related to AIC.

When fitting models, it is possible to increase the likelihood by adding parameters, but doing so may result in overfitting. The BIC resolves this problem by introducing a penalty term for the number of parameters in the model. The penalty term is larger in BIC than in AIC.

$$BIC = ln(n)k - 2ln(\hat{L})$$

1. $\hat{L}$ is the maximized value of the likelihood function of the model
2. n is the number of data points
3. k is the number of free parameters to be estimated

Reference: What is Bayesian Information Criterion (BIC)?

Table of Contents

Ridge Regression

"Ridge regression is a model-tuning method that is used to analyze any data that suffers from multicollinearity" [1].

It is also known as L2 regularization [1].

L2 regularization adds the sum of the squared values of the model's coefficients to the loss function [2].

L2 regularization does not force the coefficients to be exactly zero but instead encourages them to be small [3].

Ref 1: What is Ridge Regression?

Ref 2: What is Ridge Regression?

Ref 3: L1 And L2 Regularization Explained

Ridge Regression

Similar to L1 regularization,λ is the regularization parameter, and w_i represents the model coefficients. The sum is taken over all coefficients, and the squares of the coefficients are summed.

$L2 = \lambda * \sum(w_i^2)$ [3]

Ridge Regression Pros & Cons

Pros:

1. Trades variance for bias (i.e. in presence of co-linearity, it is worth to have biased results, in order to lower the variance.)
2. Prevents over fitting

Cons:

1. Increases bias
2. Need to select perfect λ
3. Model interpret-ability is low

Reference: Pros and cons of common Machine Learning algorithms

Bayesian Ridge

1. Bayesian Ridge works similar to the standard ridge.
2. Unlike the standard ridge where λ is set manually, Bayesian Ridge tunes the parameters based on prior data using Bayes Theorem.
3. It assumes a spherical Gaussian distribution.

Automatic Relevance Determination (ARD)

Automatic Relevance Determination Regression, which is also known as Relevance Vector Regression (RVR), uses a Bayesian framework to automatically select the most relevant features by regarding a few key features with high weights, effectively performing feature selection alongside regression. It assumes an elliptical Gaussian distribution instead of a spherical one.

Reference: Automatic Relevance Determination Regression: Unleashing the Power of Python for Enhanced Predictive Modeling

Advantages of ARD

1. **Feature Selection:** It automatically determines the relevance of features, reducing the risk of overfitting and eliminating the need for manual feature selection. This results in more robust models that capture the most informative aspects of the data.

2. **Model Interpretability:** By assigning relevance weights to each feature, ARD regression provides insights into the importance of different variables in the predictive process. This not only enhances the interpretability of the model but also aids in identifying key drivers behind the predictions.

3. **Computational Efficiency:** It effectively handles high-dimensional datasets by discarding irrelevant features, reducing the dimensionality of the problem. This leads to faster training and inference times, making it suitable for large-scale data analysis.

Reference: Automatic Relevance Determination Regression: Unleashing the Power of Python for Enhanced Predictive Modeling

Table of Contents

Elastic Nets

Elastic Nets are simply regression models that include both L1 and L2 regularization, a combination of Lasso and Ridge.

The Elastic Net regularization term is defined as:

$$\lambda_1 * \sum |w_i| + \lambda_2 + \sum (w_i^2)$$

Where:

1. λ_1 and λ_2 are the regularization parameters
2. w_i represent the individual model coefficients

Reference: 1.1.10.1. Bayesian Ridge Regression

Table of Contents

Support Vector Machines

Support Vector Machines (SVM) can be used for classification and regression.

SVMs construct a **hyperplane** in multidimensional space to separate different classes.

The **hyperplane** is iteratively constructed using error minimization to find the optimal hyperplane.

Support Vector Machine Terminology

1. **Support Vectors**

   A set of data points closest to the hyperplane.

2. **Hyperplane**

   A decision plane which separates objects by class.

3. **Margin**

   The gap between a hyperplane and support vectors. The distance is measured perpendicular from the plane to the vector.

4. **Maximum Marginal Hyperplane**

   The hyperplane that gives the best division of classes. The larger the margin, the better.

5. **Kernel**

   The kernel is the method by which an SVM transforms lower dimensional data into higher dimensional data. Hyperplanes are created in the higher dimensional space. There are multiple kernels available.

SVMs Illustrated

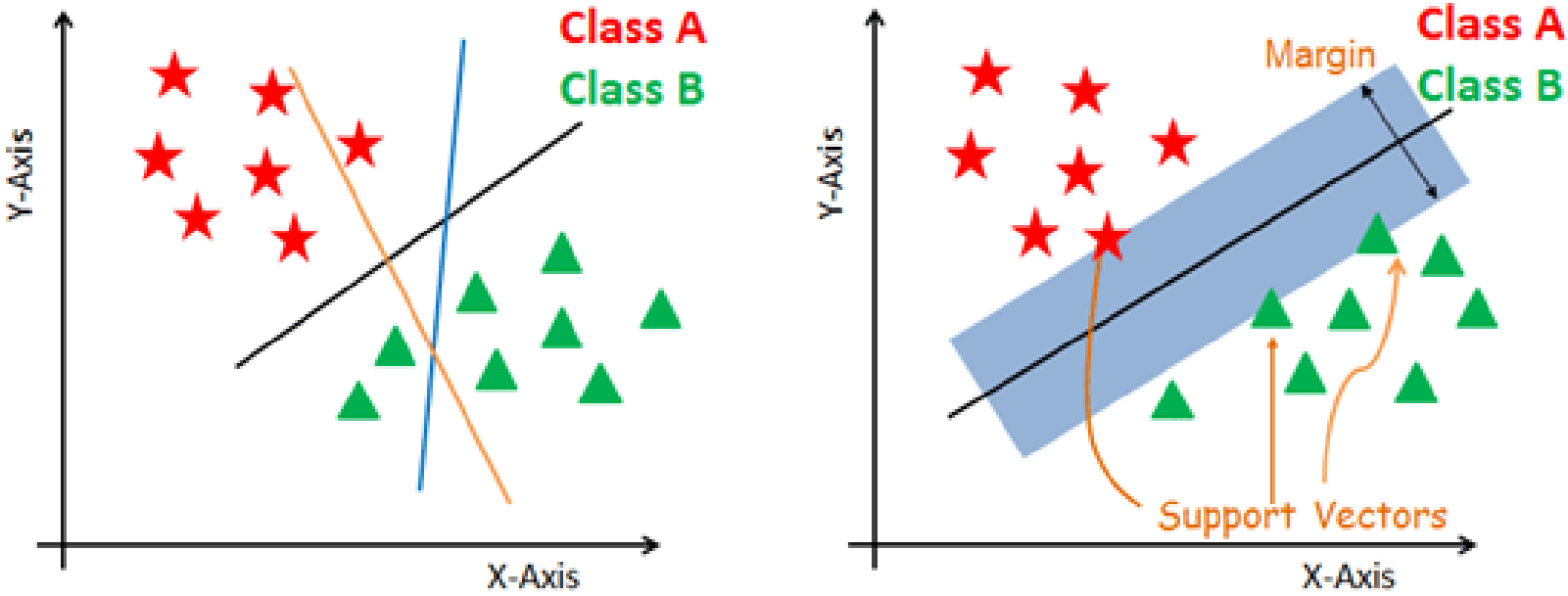


Figure: Simple SVM Illustration

SVMs Illustrated

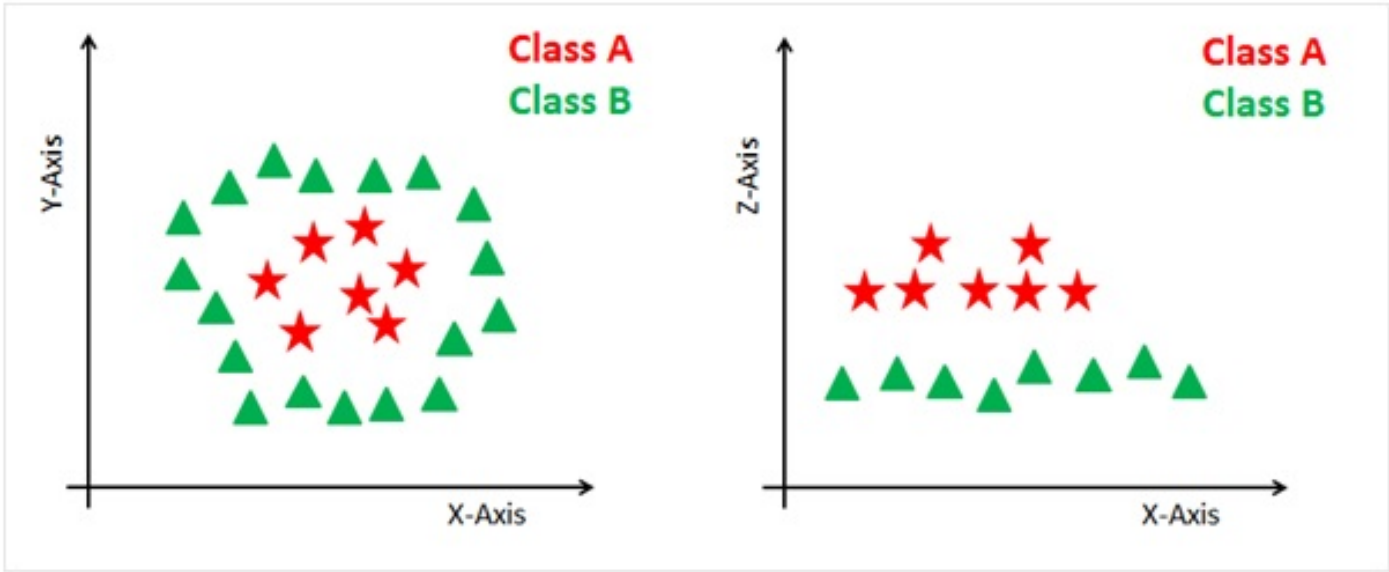


Figure: Simple Kernel Illustration

SVM Kernels

1. **Linear**

$$k(x, y) = x^T y$$

2. **Polynomial**

$$k(x, y) = (\gamma x^T y + c_0)^d$$

3. **Gaussian Radial Basis Function (RBF)**

$$k(x, y) = exp\left(-\gamma \| x - y \|^2\right)$$

SVM Kernels

1. **Sigmoid**

$$k(x, y) = tanh(\gamma x^T y + c_0)$$

2. **Global Alignment Kernel (GAK)**

$$k(x, y) = \sum_{\pi \in A(x,y)} \prod_{i=1}^{|\pi|} exp\left(-\frac{\|x_{1\pi_{1(i)}} - y_{\pi_{2j}}\|^2}{2\sigma^2}\right)$$

GAK Kernel

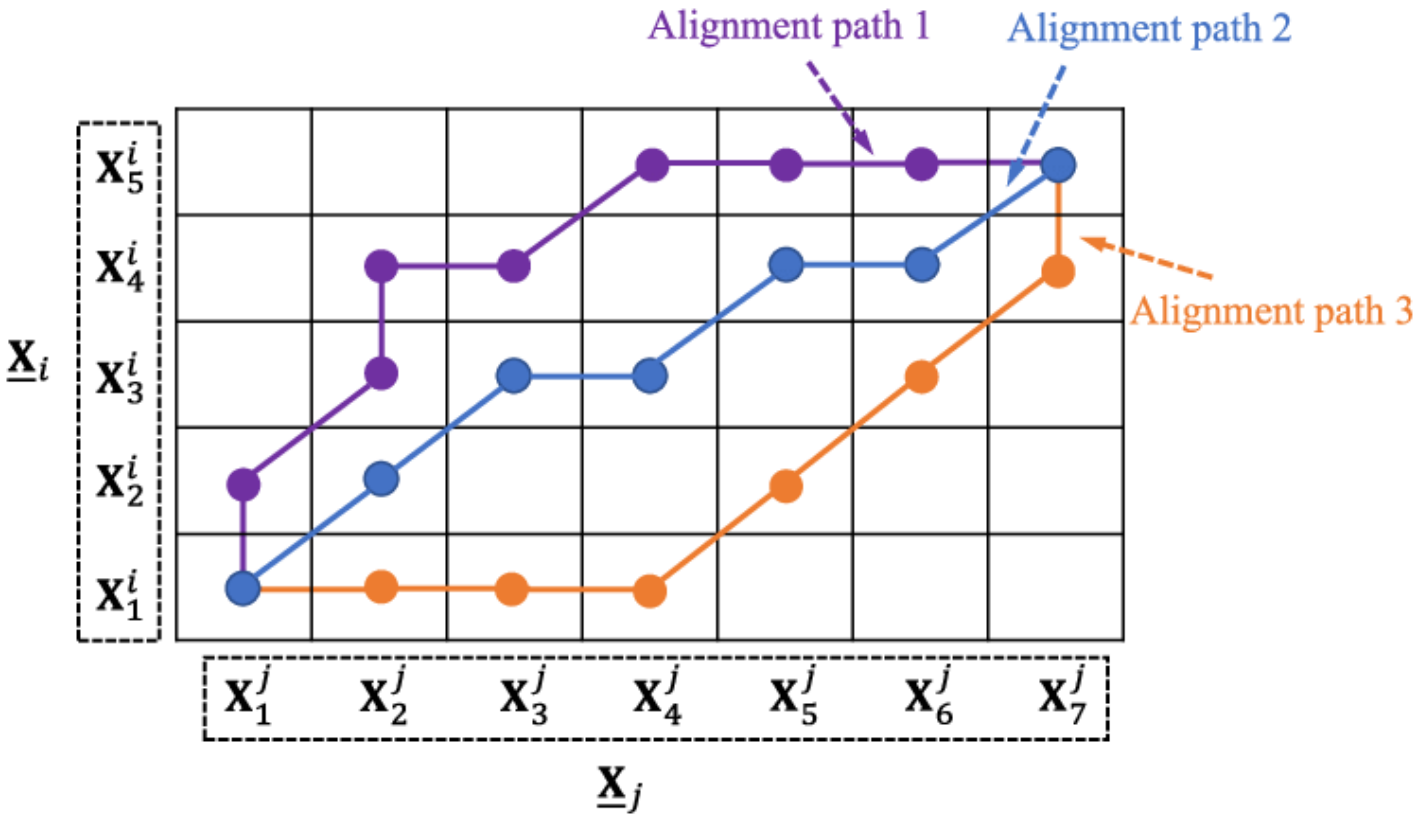


Figure: GAK Kernel Illustration

SVM Pros & Cons

1. **Pros**

   Work well when there is a clear margin of separation

   More effective in higher dimensional space

   Effective in cases where there are more dimensions than the number of samples

   Relatively memory efficient

2. **Cons**

   Doesn't work well with large data sets

   Doesn't perform well with a lot of noise

   Under-performs when the number of features exceeds the number of samples

   Difficult to explain decision process

SVM Regression Example

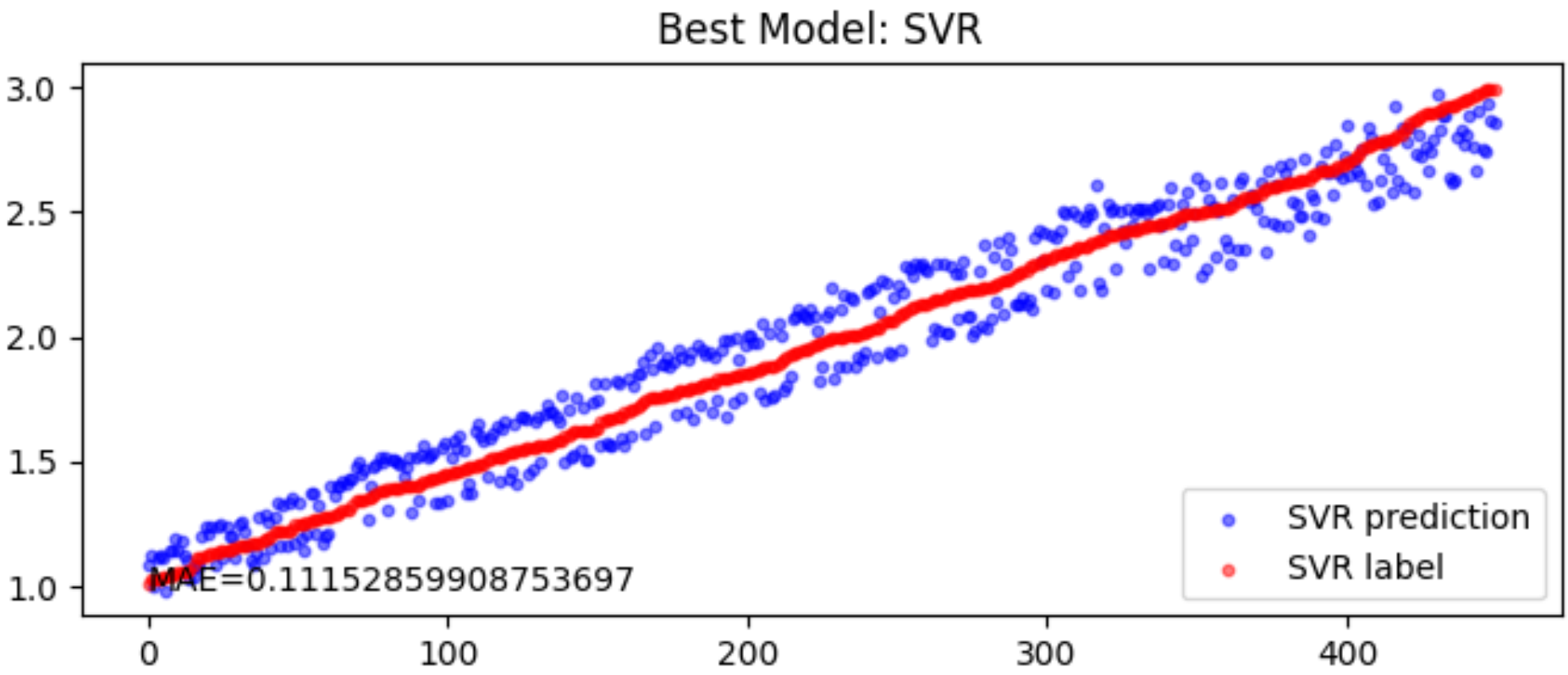


Figure: SVR Plot

Table of Contents

Nearest Neighbors

1. **Nearest Neighbors**

   Nearest Neighbors algorithms classify a data point by determining which class a majority of the points neighbors belong to

   There are several methods for determining neighbors

K Nearest Neighbors

1. **KNN**

   Assumption: "Similar things exist in close proximity"

   Estimates the likelihood of a data point being a member of a group or another depending on the data points nearest to it are in

   Note: There is a "Time Series" version of this algorithm available in the TS Learn package. It is the same as the standard KNN, but with distance metrics more suited for time series data, such as Dynamic Time Warping.

K Nearest Neighbors

1. **KNN Algorithm**

   1. Load data

   2. Initialize **K** to your chosen # of neighbors

   3. For each data point: Calculate the distance between query example & all current examples in the dataset

   4. Then add the distance & index of the example to a ordered collection

   5. Sort collection in ascending order

   6. Select the first **K** entries

   7. If regression, return the mean of the **K** labels

   7. If classification, return the mode of the **K** labels

Nearest Neighbors Pros & Cons

Pros:

Fast. In particular, KNN is a **Lazy Learner**, which has no training period.

Easy to implement

Easy to understand

Useful for both Classification & Regression

Cons:

Does not work well with large datasets

Does not work well with high dimensions

Requires feature scaling

Sensitive to noise

Table of Contents

RANSAC

1. One of the common issues in machine learning are outliers in the dataset, but we do not want the outliers to influence the prediction.
2. **Random Sample Consensus (RANSAC)** is an algorithm that can overcome this issue and a is a robust solution to replace linear regression if outliers are the concern of your prediction model.
3. RANSAC is an improved algorithm to encounter outliers in linear regression. It is an iterative and non-deterministic method to train the model to take out the outlier influence in the model.

Reference: RANSAC - Machine Learning Algorithm Review Note

Huber Regression

Huber regression is a technique that is robust to outliers [1]. It uses a different loss function than the traditional least-squares.

$$minimize\ \beta \quad \sum_{i=1}^{m} \phi(y_i - x_i^T \beta) \tag{1}$$

for variable $\beta \in R^n$, where the loss ϕ is the Huber function with threshold $M > 0$,

$$f(n) = \begin{cases} \mu^2 & if\ |\mu| \leq M \\ 2\mu - M^2 & if\ |\mu| > M \end{cases} \tag{2}$$

"This function is identical to the least squares penalty for small residuals, but on large residuals, its penalty is lower and increases linearly rather than quadratically. It is thus more forgiving of outliers."

Reference: Huber Regression

Huber Pros

1. **Robustness to Outliers**
2. **Balances Least Squares & Absolute Deviation**
3. **Continuity and Differentiability**
4. **Tunable Robustness**
5. **Predictive Performance**
6. **Computational Efficiency**

Reference: A Comprehensive Guide to Huber Regression: Balancing Efficiency and Robustness for Reliable Parameter Estimation

Huber Cons

1. **Parameter Tuning**
2. **Sensitivity to Outliers**
3. **Computational Complexity**
4. **Subjectivity in Loss Function Choice**

Reference: A Comprehensive Guide to Huber Regression: Balancing Efficiency and Robustness for Reliable Parameter Estimation

Orthogonal Marching Pursuit

Orthogonal Matching Pursuit (OMP) is a greedy algorithm used to solve the problem of sparse signal representation. The main goal is to approximate a signal by finding a sparse linear combination of a set of basis functions or dictionary elements (columns of a matrix). This method is widely used in signal processing, machine learning, and compressed sensing.

OMP Algorithm

ALGORITHM

1. Initialization

1. Start with an initial residual (usually the signal itself)
2. The solution is initialized as an empty set.

2. Selection Step

1. In each iteration, the algorithm finds the dictionary element (column of the dictionary matrix) that is most correlated with the current residual (using a similarity measure, often the dot product or inner product).
2. The selected element is added to the set of chosen elements.

OMP Algorithm

3. Update Step

1. Solve a least-squares problem to update the coefficients of the selected dictionary elements. This ensures the best approximation of the signal using the chosen dictionary elements.
2. The residual is updated by subtracting the projection of the signal onto the selected elements.

4. Repeat

1. The algorithm repeats steps 2 and 3 for a predefined number of iterations or until the residual is small enough (i.e., the signal has been approximated sufficiently well).

5. Termination

1. The process stops when a predefined stopping condition is met, such as reaching a certain number of iterations or the residual being sufficiently small.

OMP Key Features

1. **Greedy Algorithm:** It builds the approximation iteratively, always choosing the best match at each step.
2. **Sparsity:** The goal of OMP is to produce a sparse solution (i.e., using as few dictionary elements as possible).
3. **Orthogonality:** Each new selection of a dictionary element is orthogonal to the previous residuals. This means that each chosen element adds unique information to the approximation, reducing redundancy.

A Passive Aggressive Regressor is a type of machine learning model used for regression tasks, which means predicting a continuous value. It's part of the family of linear models but has a unique way of updating its parameters during training. It's called "passive-aggressive" because of its update mechanism.

How it works:

Passive: If the model makes a correct prediction (close to the target value), it doesn't make any changes to its parameters—it's passive.

Aggressive: If the model makes a wrong prediction (the predicted value is far from the target), it aggressively adjusts its parameters to minimize the error.

The model is designed to be efficient and scalable for large datasets, as it performs updates after each individual data point (online learning). This makes it suitable for problems with a large number of data points or when the data arrives in a stream.

Perceptron Regressor

A **Perceptron Regressor** is a type of machine learning model used for regression tasks. It's based on the perceptron, which is one of the simplest types of artificial neural networks. However, unlike the traditional perceptron that's used for classification, the perceptron regressor is designed for predicting continuous numerical values.

Basic Concept:

1. **Perceptron:** A perceptron is a type of linear classifier that makes predictions by calculating a weighted sum of the input features and passing it through an activation function (typically a step function in its simplest form).
2. **Regression:** In a regression task, the goal is to predict a continuous value (such as predicting the price of a house based on its features).

Perceptron Regressor

How the Perceptron Regressor Works:

1. The perceptron regressor model is trained to minimize the difference between its predicted output and the actual target value.
2. The model uses a linear function of the input features and learns the weights that minimize the error. The output is a continuous value, rather than a discrete class label.

Quantile Regressor

A **Quantile Regressor** is a type of regression model that estimates the conditional quantiles of the response variable, rather than its mean. In simpler terms, while traditional linear regression models estimate the expected (average) value of the dependent variable for a given set of input features, quantile regression provides a way to estimate any quantile (e.g., median, 90th percentile, etc.).

Key Concepts:

Quantiles: Quantiles divide data into intervals with equal probabilities. For example:

1. The **median** is the 50th percentile (the 0.5 quantile).
2. The **upper quartile** is the 75th percentile (the 0.75 quantile).
3. The **lower quartile** is the 25th percentile (the 0.25 quantile).

Quantile Regressor

Benefits:

1. **Robustness:** Quantile regression is more robust to outliers than mean regression. This is because instead of minimizing the sum of squared residuals (as in ordinary least squares), quantile regression minimizes a weighted sum of absolute residuals, which reduces the influence of extreme values.

2. **Flexible Modeling:** It allows you to model the relationship between independent variables and different quantiles of the dependent variable. This can be especially useful when the data has skewed distributions, heteroscedasticity (non-constant variance), or when you're interested in predicting something other than the mean (like the 90th percentile of future outcomes).

   Quantile regression uses a quantile loss function (also known as pinball loss) to estimate the desired quantile. For the τth quantile, the loss function is asymmetric, meaning the penalty for over- and under-predictions differs depending on whether the prediction is above or below the actual value.

Stochastic Gradient Descent

A **Stochastic Gradient Descent Regressor** (SGD Regressor) is a machine learning algorithm used for regression tasks. It's a variant of gradient descent that updates the model's parameters (weights) incrementally, rather than using the entire dataset at once. Here's a breakdown of the key components:

Gradient Descent:

1. In machine learning, gradient descent is an optimization algorithm used to minimize the error or loss function by iteratively updating the model's parameters in the direction that reduces the loss.
2. This is done by calculating the gradient (or derivative) of the loss function with respect to the model's parameters and moving in the opposite direction of the gradient.

SGD Regressor

Stochastic Gradient Descent (SGD):

1. Unlike batch gradient descent, which uses the entire dataset to calculate the gradient in each step, stochastic gradient descent uses a single randomly selected data point (or a small batch) to compute the gradient at each step.
2. Because it updates the model's parameters after looking at just one data point, SGD is faster than batch gradient descent and can handle large datasets more efficiently.

SGD Regressor:

1. The SGD Regressor specifically applies stochastic gradient descent to regression problems. It aims to minimize the loss function (e.g., mean squared error) between predicted values and true values.
2. It is typically used when you want to fit a model to predict continuous values based on input features (like predicting house prices or stock prices).

SGD Benefits

1. **Efficiency:** It's faster than traditional gradient descent methods, especially when dealing with large datasets.
2. **Flexibility**: You can tune the learning rate and other hyperparameters to control the behavior of the model.
3. **Convergence:** Since it's based on individual data points, it can be noisy, but with enough iterations, it usually converges to an optimal solution. Online Learning: SGD can be used for online learning, where the model can be updated incrementally as new data comes in.

Table of Contents

Thank you!

E-mail: stephen.coshatt@uga.edu