Unit 2. Supervised Learning (10 Hrs.)

Objective:

- ✓ *Design and implement supervised learning algorithms to solve real world problems.*

2.1. Types of Supervised Learning

Supervised machine learning involves training a model on labeled data to learn patterns and relationships, which it then uses to make accurate predictions on new data.

Supervised learning is a category of machine learning that uses labeled datasets to train algorithms to predict outcomes and recognize patterns.

- Supervised learning algorithms are given labeled training to learn the relationship between the input and the outputs.
- The goal of supervised learning is to make accurate predictions when given new, unseen data.
- Supervised machine learning algorithms make it easier for organizations to create complex models that can make accurate predictions.
- Results of Supervised Machine Learning are widely used across various industries and fields, including healthcare, marketing, financial services, and more.

How does Supervised Learning Work?

- The data used in supervised learning are labeled (i.e. it contains examples of both inputs (called features) and correct outputs (labels or target variables)).
- The algorithms analyze a large dataset of training pairs to infer what a desired output value would be when asked to make a prediction on new data. This is achieved by adjusting the model's parameters to minimize the difference between its predictions and the actual labels.
- Over time, it adjusts itself to minimize errors and improve accuracy.

By: Er. Mukunda Paudel, M.Sc. CE
Email: paudelmuku@gmail.com

Example Case-2.1

Suppose you want to teach a model to identify pictures of animal. You provide a labeled dataset that contains many different examples of types of animals and the names of each species. You let the algorithm try to define what set of characteristics belongs to each animal based on the labeled outputs. You can then test the model by showing it an animal picture and asking it to guess what species it is. If the model provides an incorrect answer, you can continue training it and adjusting its parameters with more examples to improve its accuracy and minimize errors.

Once the model has been trained and tested, you can use it to make predictions on unknown data based on the previous knowledge it has learned.

Common supervised learning examples includes,

- Image Classification: For example, an algorithm might be used to recognize a person in an image and automatically tag them on a social media platform.
- Fraud Detection: For example, Models are trained on datasets that contain both fraudulent and non-fraudulent activity so they can be used to flag suspicious activity in real time.
- Recommendation systems: For example, online platforms and streaming services to power recommendations based on previous customer behavior or shopping history.
- Risk assessment: For example, banks and other financial services companies determine whether customers are likely to default loans, helping to minimize risk in their portfolios.

Supervised Learning can be applied to mainly **two types** of problems:

2.1.1. Regression

2.1.2. Classification

2.2. Regression

- Regression is supervised learning techniques where the goal is to predict a continuous numerical value as an output based on one or more independent features in which machine learning algorithm is trained with both input features and output labels.
- Regression finds the relationships between variables by estimating how one variable affects the other so that predictions can be made.

- Regression algorithms are used to predict a real or continuous value, where the algorithm detects a relationship between two or more variables.
- There are two types of variables used in regression
    1. ***Dependent Variable or Target Variable (Y)***: variable we are trying to predict (for e.g. price of house)
    2. ***Independent Variable or Features or predictors (X):*** variables that influence the prediction (in above example 2.3, area, bedrooms, age and location are independent variables)

Mathematically: Regression is statistical methods that allow to predict a continuous outcome (y) based on the value of one or more predictor variables (x).

Example-2.2

Let's predict a salary of person based on work experience. For instance, a supervised learning algorithm would be fed inputs related to work experience (e.g., duration of time, the industry or field, location, etc.) and the corresponding assigned salary amount. After the model is trained, it could be used to predict the average salary based on work experience.

Example-2.3

Imagine you are trying to predict the price of a house. You might consider various factors like its size, location, age and number of bedrooms. In a house price dataset, the independent variables are columns used to predict, such as the "Area", "Bedrooms", "Age", and "Location". The Dependent variable will be the "Price" column – the feature to be predicted.

In Summary: Regression is a statistical method used in machine learning to model and analyze the relationships between a dependent variable (target) and one or more independent variables (predictors). The primary purpose of regression is to predict a continuous outcome based on input features.

Regression can be ***classified into different types*** based on the number of predictor variables and the nature of the relationship between variables.

2.2.1. Linear Regression

- Linear regression is a statistical technique used to find the relationship between a dependent variable and one or two or more than two independent variables.
- In an ML context, linear regression finds the relationship between features and a label. And can be used for both prediction and understanding the nature of the relationship.
- Also called simple regression.
- Regression allows you to estimate how a dependent variable changes as the independent variable(s) change.
- Can be used in predicting house prices, stock prices, or sales based on multiple factors like location, interest rates, and marketing spend.

2.2.1.1 Simple and multiple regression

Simple Linear Regression

- Simple linear regression is a statistical method used to model the relationship between two variables: a dependent variable and an independent variable.
- The dependent variable is the variable we want to predict, while the independent variable is the variable that we use to make the prediction.
- In simple linear regression, we assume that there is a linear relationship between the two variables i.e. the change in the independent variable is directly proportional to the change in the dependent variable.

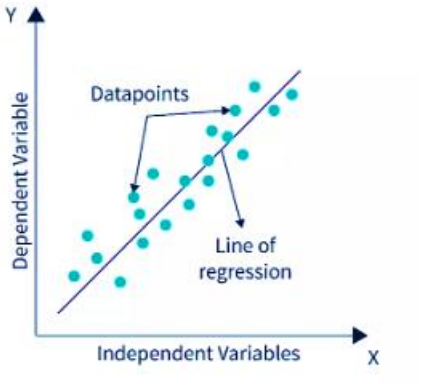


Figure: Linear Regression (best fit line)

In algebraic terms (or you can say mathematically),

Simple linear regression can be expressed as y = mx + b

where, y is the dependent variable / predicted value

x is the independent variable / input variable

m is the slope of line (how much y changes when x changes

b is the intercept (the value of y when x=0)

For example, in the linear regression formula of y = 4x + 7, there is only one possible outcome of "y" if "x" is defined as 2.

In Machine Learning,

We write the equation for a linear regression model as follows:

$y' = b + w_1x_1 + e$

Where,

y′ is the predicted label / output.

b is the **bias** of the model. In ML, bias is sometimes referred to as w_0.

w_1 is the **weight** of the feature.

x_1 is a **feature /** input. And e is error of the estimation

During training, the model calculates the weight and bias that produce the best model.

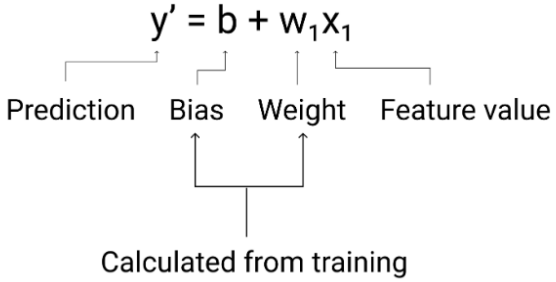


Figure: Mathematical Representation of linear model

By: Er. Mukunda Paudel, M.Sc. CE
Email: paudelmuku@gmail.com

Multiple Linear Regression

Multiple linear regression is a statistical method used to model the relationship between two variables: a dependent variable and two or more predictors or independent variable.

The idea behind multiple linear regression is similar to simple linear regression, except that we now have multiple independent variables that we use to make our prediction.

Example:

let's say we want to predict a person's salary based on their age, education, and years of experience. In this case, salary is the dependent variable, while age, education, and years of experience are the independent variables. We would collect data on these variables for a sample of individuals and use this data to create a regression model. The model would allow us to predict a person's salary based on their age, education, and years of experience.

$$y' = b + w_1x_1 + w_2x_2 + w_3x_3 \ldots\ldots\ldots\ldots + w_nx_n + e$$

where y', b w and x have similar meaning as in simple linear regression

e = model error (i.e. how much variation there is in your estimation of y')

in above example there are 3 independent variables so equation is up to w_3x_3**.** when the input variables are increased then the equation adjusts accordingly**.**

What is the difference between Simple linear and multiple linear regression?

The main difference between simple linear regression and multiple linear regression is the number of independent variables used in the model.

Another difference is the complexity of the model. Simple linear regression models are relatively simple and easy to interpret, as they involve only two variables. Multiple linear regression models, on the other hand, are more complex and require more computational power. They also require more careful interpretation, as the relationships between the independent variables and the dependent variable can be more difficult to understand.

By: Er. Mukunda Paudel, M.Sc. CE
Email: paudelmuku@gmail.com

2.2.1.2. Polynomial Regression

Polynomial regression is used when the relationship between the dependent and independent variables is nonlinear. It fits a polynomial equation (e.g., quadratic, cubic) to the data points, making it suitable for datasets with curves where the relationship between the variables is better represented by a curve rather than a straight line.

- Polynomial regression is an essential extension of linear regression used to model non-linear relationships in data.
- In many real-world scenarios, the relationship between variables isn't linear, making polynomial regression a suitable alternative for achieving better predictive accuracy.
- This technique allows machine learning models to capture curved patterns in data by fitting polynomial equations of higher degrees.
- Understanding polynomial regression equips data scientists with the tools needed to handle complex datasets effectively.
- There is no limit to the degree in a polynomial equation, and it can go up to the nth value, numerous kinds of polynomial regression exist.
- Generally, second degree of a polynomial equation is used.

As a key method in supervised learning, it can be used in Modeling complex relationships in data such as,

- predicting the progression of diseases or analyzing the impact of multiple factors on temperature changes.
- predict a non-linear trend like population growth over time etc.
- Predicting Tissue Growth Rates
- Estimating Mortality Rates based on factors like age, lifestyle, and environmental conditions
- Speed Control in Automated Systems such as those used in self-driving cars, speed adjustments are often influenced by factors like terrain, traffic, and weather conditions
- Finance, biology, physics where patterns are often non-linear.

Example:

Imagine you want to predict how many likes your new social media post will have at any given point after the publication. There is no linear correlation between the number of likes and the time that passes. Your new post will probably get many likes in the first 24 hours after publication, and then its popularity will decrease. Such kind of problems can be modeled by using polynomial regression model as linear regression.

Mathematically,

Polynomial Regression is a form of linear regression in which the relationship between the independent variable x and dependent variable y is modelled as an *nth-degree* polynomial.

When the relationship is non-linear, a polynomial regression model introduces higher-degree polynomial terms.

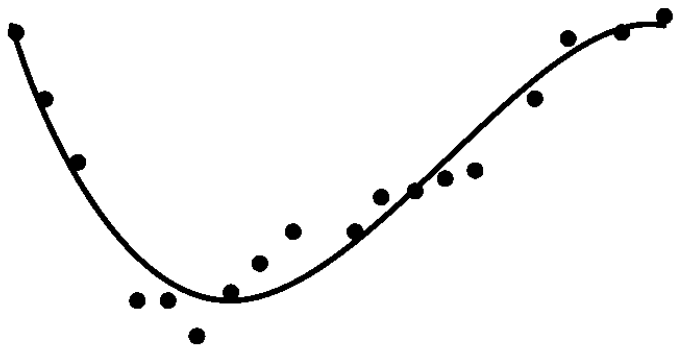


Figure: Polynomial Regression

The general form of a polynomial regression equation of degree n is:

$$y' = b + w_1x_1 + w_2x_2^2 + w_3x_3^3 \ldots\ldots\ldots\ldots + w_n x_n^n + e$$

where,

y is the dependent variable.

x is the independent variable.

w_1, w_2, w_3… w_n are the weight / coefficients of the polynomial terms.

n is the degree of the polynomial.

e- represents the error term.

Advantages of Polynomial Linear Regression

- Flexibility for Non-Linear Datasets
- Applicability Across Various Fields: Biology, Finance.
- Better Performance in Specific Scenarios

Dis-advantages of Polynomial Linear Regression

- Overfitting Risk with High-Degree Polynomials
- Computational Complexity
- Selecting the Optimal Polynomial Degree

2.2.2. Regularization Techniques

Imagine you're an architect, standing before a majestic blueprint of a future skyscraper. It's meant to be tall, imposing, and graceful. But there's one problem. It's at risk of becoming unstable. To prevent this, you need to ensure that its structural design doesn't get overloaded with unnecessary materials. Every beam, every bolt, must serve a purpose no more, no less.

Similarly, in machine learning, building a model that fits the data is much like building that skyscraper. Too much complexity and the model, like your building, will collapse under its own weight.

But, ***how do we manage the complexity of our machine learning models?***

- The answer is regularization.

In multiple and polynomial regression, if we rely too much on numerous features, the machine learning model works perfectly on training data but fail miserably on new / unseen data (overfitting). In such cases, model is not only learning the core patterns but also the noise in the data.

Regularization removes or shrinks unnecessary coefficients in the model, helping you build something strong and efficient.

Regularization is a method of adding a penalty term to the cost function of a linear regression model, which reduces the magnitude of the coefficients or weights. The penalty term can be either L1 or L2 norm, which correspond to different types of regularization: Lasso and Ridge,

respectively. By adding regularization, you can prevent the model from fitting too closely to the training data and reduce the variance of the model. However, regularization also introduces some bias to the model, as it shrinks the coefficients towards zero or a constant.

What is the bias-variance trade-off?

The bias-variance trade-off is a fundamental concept in machine learning, which describes the trade-off between the error due to the model's complexity and the error due to the model's sensitivity to the data. A high-bias model is one that is too simple and cannot capture the underlying patterns of the data, resulting in a large error on both the training and the test data. A high-variance model is one that is too complex and fits the training data too well, resulting in a small error on the training data but a large error on the test data. Ideally, you want to find a model that has low bias and low variance, which means that it can generalize well to new data.

How does regularization affect the bias-variance trade-off?

Regularization is a way of controlling the complexity of a linear regression model, by penalizing the coefficients that are not important or relevant for the prediction. By doing so, regularization can reduce the variance of the model, as it prevents overfitting and makes the model more robust to noise and outliers. However, regularization also increases the bias of the model, as it forces the coefficients to be smaller or zero, which means that the model may not capture some of the true relationships between the variables. Therefore, regularization involves a trade-off between bias and variance, and you need to find the optimal level of regularization that minimizes the total error of the model.

How do you choose the optimal level of regularization?

The optimal level of regularization depends on the data and the problem you are trying to solve. One way to choose the optimal level of regularization is to use cross-validation, which is a technique of splitting the data into multiple subsets and using some of them for training and some of them for validation. By varying the regularization parameter, you can evaluate how well the model performs on the validation data, and choose the value that minimizes the validation error. Another way to choose the optimal level of regularization is to use grid search or random search, which are methods of exploring a range of possible values for the regularization parameter and finding the best one based on a scoring metric.

How do you implement regularization in linear regression?

There are different ways to implement regularization in linear regression, depending on the type of regularization and the programming language or tool you are using. For example, if you are using Python and scikit-learn, you can use the classes Linear Regression, Lasso, or Ridge to create and fit linear regression models with or without regularization. The Lasso class implements L1 regularization, which tends to select a subset of features and set the rest of the coefficients to zero. The Ridge class implements L2 regularization, which tends to shrink all the coefficients towards zero but not exactly zero. You can also use the ElasticNet class, which combines both L1 and L2 regularization, and allows you to control the balance between them. To specify the regularization parameter, you can use the alpha argument, which controls the strength of the penalty. A higher alpha means more regularization and more bias, while a lower alpha means less regularization and more variance.

What are the benefits and limitations of regularization in linear regression?

Regularization in linear regression can have several benefits and limitations, depending on the data and the problem you are attempting to solve. It can reduce overfitting and improve generalization, as well as simplify the model and make it more interpretable. Regularization can also handle multicollinearity, which is a situation where some of the explanatory variables are highly correlated with each other, by reducing the effect of redundant or irrelevant variables. However, regularization can also introduce underfitting and reduce accuracy, by ignoring some of the important or relevant features or relationships in the data. Additionally, it can increase bias and make the model less flexible, by constraining the coefficients to be smaller or zero. Finally, regularization can require tuning and validation, by adding an extra parameter that needs to be chosen carefully and evaluated on different data sets.

Technically,

Regularization is a machine learning technique designed to prevent overfitting by adding a penalty term to the model's loss function. This penalty reduces the model's tendency to over-rely on specific features by shrinking the magnitude of the coefficients, leading to better generalization on unseen data. By controlling model complexity, regularization ensures the model captures true patterns rather than fitting noise in the training data, improving performance on new datasets.

By: Er. Mukunda Paudel, M.Sc. CE
Email: paudelmuku@gmail.com

In **linear regression**, we typically minimize the sum of squared errors (also known as the **Mean Squared Error**, or MSE) between the predicted and actual outputs. Regularization modifies this loss function / cost function by adding a penalty term based on the size of the coefficients (weights) associated with the features in the model.

$$\text{MSE}=\frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2$$

This equation represents the Mean Square Error for Linear Regression which measures how well the model predicts the target variable. Where,

y_i Actual value for the i^{th} sample.

Y'_i Predicted value for the i^{th} sample.

n Total number of training samples.

This is the traditional error term used in linear regression.

In practice, regularization involves adjusting the coefficients so that the model doesn't place too much importance on any one feature. There are two main types of regularization in linear regression:

2.2.2.1. L2 Regularization (Ridge Regression):

Shrinks the coefficients by adding the sum of their squares to the loss function.

In Ridge Regression, we add the squared values of the coefficients to the loss function. This penalizes larger coefficients, forcing the model to maintain smaller weights. In essence, Ridge discourages the model from becoming overly dependent on any one feature.

The loss function / Cost function for Ridge Regression looks like this:

$$L_{ridge}=\frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2 + \lambda \sum_{j=1}^{p}{\beta_j}^2$$

Where, $\frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2$ is the MSE and $\lambda \sum_{j=1}^{p}{\beta_j}^2$ is the L2 Regularization term (penalty). This term Adds a penalty based on the size of the model coefficients.

Where,

Bj-The j-th model parameter (weight).

p: Total number of features (predictors).

λ: Regularization strength (hyperparameter). If

λ=0: No regularization → Equivalent to standard linear regression.

Λ large: More regularization → Coefficients are penalized more heavily → Model becomes simpler and possibly underfits.

The right value of λ\lambdaλ is typically chosen using **cross-validation**.

MSE ensures the model fits the data well and L2 Penalty prevents overfitting by shrinking the coefficient Bj towards zero (but not exactely zero).

Here, **λ (lambda)** is the regularization parameter that controls how much we penalize the model for large coefficients. A higher value of λ means more regularization, i.e., more emphasis on keeping the coefficients small.

2.2.2.2. L1 Regularization (Lasso Regression):

Shrinks the coefficients by adding the sum of their absolute values to the loss function.

Lasso Regression is similar to Ridge, but instead of adding the squared values of the coefficients, it adds their absolute values. This has the effect of shrinking some coefficients all the way to zero, effectively removing irrelevant features from the model.

The loss function for Lasso looks like this:

$$L_{lasso}= \frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2 + \lambda \sum_{j=1}^{p}|\beta_j|$$

Lasso performs **feature selection** by eliminating features that don't contribute much to the model. This is particularly useful when you have a large number of features and want to identify the most important ones.

2.2.2.3. Bias-variance tradeoff

How do we pick the best model? To address this question, we must first comprehend the trade-off between bias and variance.

The mistake is due to the model's simple assumptions in fitting the data is referred to as bias. A high bias indicates that the model is unable to capture data patterns, resulting in under-fitting.

The mistake caused by the complicated model trying to match the data is referred to as variance. When a model has a high variance, it passes over the majority of the data points, causing the data to overfit.

As the model complexity grows, the bias reduces while the variance increases, and vice versa. A machine learning model should, in theory, have minimal variance and bias. However, having both is nearly impossible. As a result, a trade-off must be made in order to build a strong model that performs well on both train and unseen data

2.2.3. Support Vector Regression

- Support Vector Regression (SVR) is a type of Support Vector Machine (SVM) algorithms, commonly used for regression analysis.
- SVMs are powerful supervised learning algorithms that are primarily used for classification problems.
- SVM aim to find the optimal *hyperplane* that best separates the two classes in the input data.
- A hyperplane is a flat subspace of dimension p-1 in a p-dimensional space, where p is the number of input features.
- The optimal hyperplane is the one that maximizes the *margin*, which is the distance between the hyperplane and the closest data points from each class, known as *support vectors* (for detailed explanation see on classification section below)

Similar to SVMs, SVR uses the concept of a hyperplane and margin but there are differences in their definitions.

- In SVR, the margin is defined as the error tolerance of the model, which is also called as the ε-*insensitive tube*.

- This tube allows some deviation of the data points from the hyperplane without being counted as errors.

- The hyperplane is the *best fit* possible to the data that fall within the ϵ-insensitive tube. The difference of SVM and SVR is summarized in the figure below.

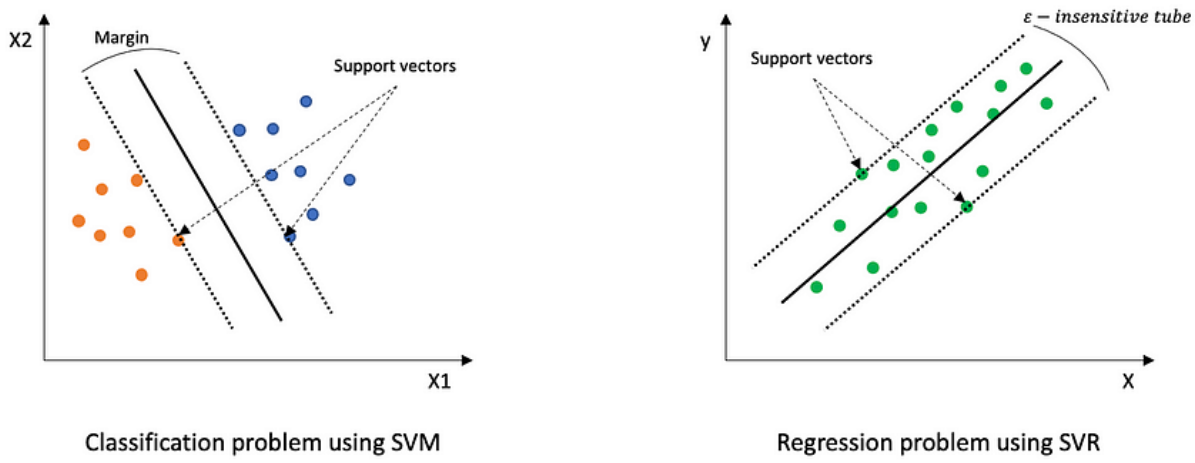


Figure: difference of SVM and SVR [IS: Medium]

In SVR, the goal is finding the best fit that accurately predicts the target variable while reducing complexity to avoid **overfitting**.

- To achieve this, an ε-insensitive tube around the hyperplane is defined, which permits some level of deviation between the actual and predicted target values, creating a balance between the complexity of the model and its generalization power.
- SVR can be mathematically formulated as a convex optimization problem.
- The objective of problem is to find a function *f(x)* that is as flat as possible while having a maximum deviation of ε from the actual targets for all the training data.
- Flatness of the function implies that it is less sensitive to small changes in the input data, which reduces the risk of overfitting.

For linear functions, flatness means having a small value of *w* in the best fit function $f(x) = wx + b$.

By: Er. Mukunda Paudel, M.Sc. CE
Email: paudelmuku@gmail.com

Mathematical Formulation in depth

SVR aims to find a function f(x)=w·ϕ(x)+b that minimizes prediction error while remaining as "flat" as possible. The optimization problem is defined as:

Minimize $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$

Subjected to $\begin{cases} y_i - (w.\phi(x_i) + b) \leq \in + \xi_i \,, \\ (w.\phi(x_i) + b) - y_i \leq \in + \xi_i \,, \\ \leq \in + \xi_i \geq 0 \; \forall_i \end{cases}$

where, w: Weight vector.

ϕ(x): Kernel function mapping input x**x** to a higher-dimensional space.

C: Regularization parameter balancing flatness and training error.

ε: Maximum allowable deviation (defines the ε-tube).

ξ_i, ξ_i^*: Slack variables for points outside the ε-tube.

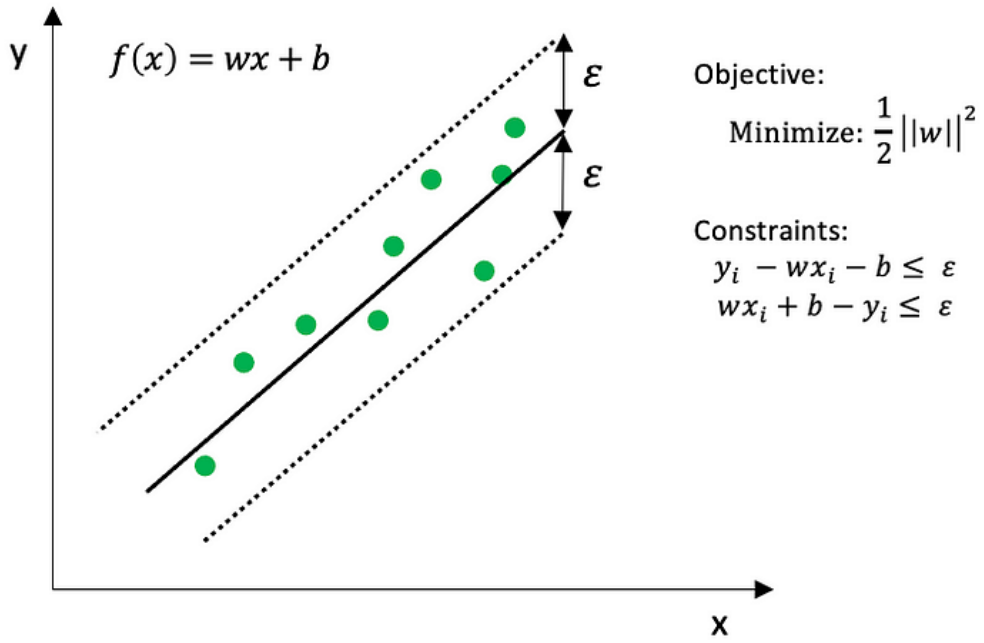
The solution depends on **support vectors** training points where errors exceed ε. The dual formulation uses Lagrange multipliers α_i, α_i^* , and the final prediction function is:

$f(x)=\sum_{i=0}^{n}(\alpha_i - \alpha_i^*)\, K(x_i, x) + b$

where, $K(x_i, x)= \phi(x_i)\cdot\phi(x)$ *is the kernel function*

Simply SVR can be represented by:

By: Er. Mukunda Paudel, M.Sc. CE
Email: paudelmuku@gmail.com

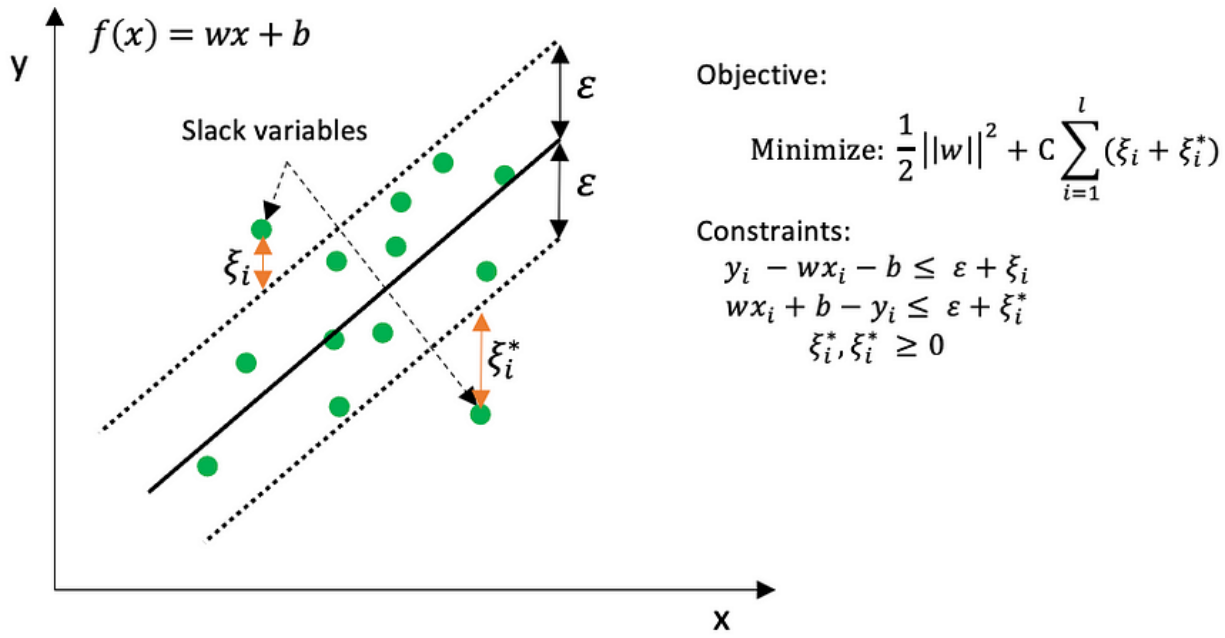


Univariate linear SVR (with zero tolerance for error)

- Sometimes optimization problem is not feasible or we may want to allow for some errors. In that case, we introduce *slack variables*, which are the data points that fall outside of the ε-insensitive tube.
- The distance of slack variables from ε-insensitive tube boundary is represented as ξ.
- To balance the trade-off between the model complexity (i.e., flatness of *f(x)*) and total deviations beyond ε-insensitive tube, we utilize a regularization parameter $C > 0$.
- The strength of the regularization is inversely proportional to *C*.

SVR assigns zero prediction error to the points that lie inside the ε-insensitive tube, whereas it penalizes the slack variables proportionally to their ξ. This feature of SVR enables it to handle overfitting more effectively than ordinary regression models.

By: Er. Mukunda Paudel, M.Sc. CE
Email: paudelmuku@gmail.com



Univariate linear SVR (allowing for errors)

SVR is ideal for scenarios where data exhibits nonlinear patterns, dimensionality is high, or robustness to outliers is critical. However, its computational demands and parameter-tuning requirements make it less suitable for large-scale or real-time applications. SVR can be used for both linear and non-linear regression problems by using various kernel functions. In cases where the data is non-linearly separable, kernel helps in finding function *f(x)* in higher dimensional space where a linear regression problem can be solved. Common kernel functions used in SVR include linear, polynomial, radial basis function (RBF), and sigmoid. SVR provides a powerful alternative to traditional regression methods in complex predictive modeling tasks.

2.3. Classification

Classification is a supervised machine learning method where the model tries to predict the correct label (class) of a given input data. Those classes can be targets, labels or categories. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.

For example, a spam detection machine learning algorithm would aim to classify emails as either "spam" or "not spam."

Types of learners in classification:

There are **two types** of learners in machine learning classification.

1. **Eager learners** are machine learning algorithms that first build a model from the training dataset before making any prediction on future datasets. They spend more time during the training process because of their eagerness to have a better generalization during the training from learning the weights, but they require less time to make predictions.

   Most machine learning algorithms are eager learners, and below are some examples:

   - Logistic Regression.
   - Support Vector Machine.
   - Decision Trees.
   - Artificial Neural Networks.

2. **Lazy learners or instance-based learners**, on the other hand, do not create any model immediately from the training data, and this is where the lazy aspect comes from. They just memorize the training data, and each time there is a need to make a prediction, they search for the nearest neighbor from the whole training data, which makes them very slow during prediction. Some examples of this kind are:

   - K-Nearest Neighbor.
   - Case-based reasoning.

Types of Classification:

Classification-based predictive modeling tasks are distinguished from each other based on the number of categories and the degree to which the categories are exclusive:

- **Binary classification** sorts data into two exclusive categories.
- **Multiclass classification** sorts data into more than two exclusive categories.
- **Multilabel classification** sorts data into nonexclusive categories.
- **Imbalanced** classification has an unequal distribution of data points across categories.

2.3.1 Logistic Regression

- Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation.
- The model delivers a binary or dichotomous outcome limited to two possible outcomes: it can be either yes/no, 0/1, or true/false but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- Logical regression analyzes the relationship between one or more independent variables and classifies data into discrete classes.
- It is extensively used in predictive modeling, where the model estimates the mathematical probability of whether an instance belongs to a specific category or not.
- Can be used in fraud detection (e.g. spam email detection), disease prediction (e.g. determine the probability of heart attacks), and many more like possibility of enrolling into a university etc.

For example, we have two classes Class A and Class B if the value of the logistic function for an input is greater than 0.5 (threshold value) then it belongs to Class A otherwise it belongs to Class B. It's referred to as regression because it is the extension of linear regression but is mainly used for classification problems.

Logistic Regression uses a special function called the **sigmoid function / activation function / logistic function** to map predictions and their probabilities. The sigmoid function refers to an S-shaped curve that converts any real value to a range between 0 and 1.

Moreover, if the output of the sigmoid function (estimated probability) is greater than a predefined threshold on the graph, the model predicts that the instance belongs to that class. If the estimated probability is less than the predefined threshold, the model predicts that the instance does not belong to the class.

For example, if the output of the sigmoid function is above 0.5, the output is considered as 1. On the other hand, if the output is less than 0.5, the output is classified as 0. Also, if the graph goes further to the negative end, the predicted value of y will be 0 and vice versa. In other words, if the

output of the sigmoid function is 0.65, it implies that there are 65% chances of the event occurring; a coin toss, for example.

The ***sigmoid function is referred to as an activation function for logistic regression*** and is defined as: $\sigma(z) = \frac{1}{1+e^{-z}}$

Where, e is the base of natural logarithms and $z = b+w_1x_1+w_2x_2+\cdots+w_nx_n$ is the linear combination of input features and coefficients. i.e. output of the linear equation, also called **log odds**.

This equation $\sigma(z) = \frac{1}{1+e^{-z}}$ becomes $y=\frac{e^{(\beta 0+\beta 1x)}}{1+e^{(\beta 0+\beta 1x)}}$ in logistic regression. Where, x is input value y is predicted output, b0 is bias or intercept term and b1 is coefficient for input (x)

Log Odds: it refers to the ways of expressing probabilities. Log odds are different from probabilities. Odds refer to the ratio of success to failure, while probability refers to the ratio of success to everything that can occur.

For example, consider that you play twelve tennis games with your friend. you won 5 games and you lost 7 games. Here, the odds of you winning are 5 to 7 (or 5/7), while the probability of you winning is 5 to 12 (as the total games played = 12).

Probability = no of wins / total games = 5/12 = 0.417 i.e. you have a 41.7 % chance of winning.

Odds = no of wins / no of losses = 5/7 = 0.7142 i.e. odds of your winnings are 71.42 %

Now, convert odds to log odds,

We have odds of winning = 0.714

Then convert odds to log odds,

log odds (logit) = log (odds) = log (0.714) = -0.336 (negative log odds means that the chance of winning is less than 50 %.

Again, convert odds to probability:

Probability (p) = $\frac{odds}{1+odds}$ = $\frac{0.714}{1+0.714}$ = 0.417 i.e. chances of win are 41.7 %

If we convert log odds to back to probability, probability = $\frac{1}{1+e^{-z}}$ = $\frac{1}{1+e^{-0.336}}$ = 0.417

By: Er. Mukunda Paudel, M.Sc. CE
Email: paudelmuku@gmail.com

Now, the question is why logistic regression uses log odds (logit) instead of probability directly?

Suppose you try to use linear regression to predict probability: $p=\beta_0+\beta_1 x$

This looks fine, but it has a serious problem i.e. Probability must be between 0 and 1, but a linear equation like $\beta_0+\beta_1 x$ can output any real number (e.g., –3, 2.5, etc.). So, the model might predict values less than 0 or greater than 1, which are invalid probabilities.

Solution is: use log odds.

Instead of modeling probability p directly, logistic regression models the mog of the odds: $\log\left(\frac{p}{1-p}\right) = \beta_0+\beta_1 x$

why this works better? Because Log Odds are Unbounded,

- While probability (p) is between 0 and 1,
- Odds can range from 0 to ∞,
- And log(odds) (logit) can range from –∞ to +∞.

So now, the linear function on the right-hand side can safely take any value, and we can convert it back to a valid probability using the sigmoid function: $p = \frac{1}{1+ e^{-(\beta 0+\beta 1 x)}}$

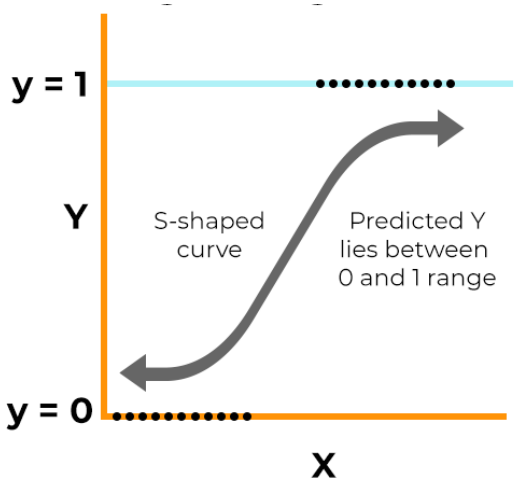


Figure: Logistic Regression curve (S-Shape curve / sigmoid curve)

By: Er. Mukunda Paudel, M.Sc. CE
Email: paudelmuku@gmail.com

There are four main classification tasks in Machine learning: binary, multi-class, multi-label, and imbalanced classifications.

1. binary classification
2. multi class classification
3. multi label classification
4. imbalance classification

2.3.1.1. Binary classification

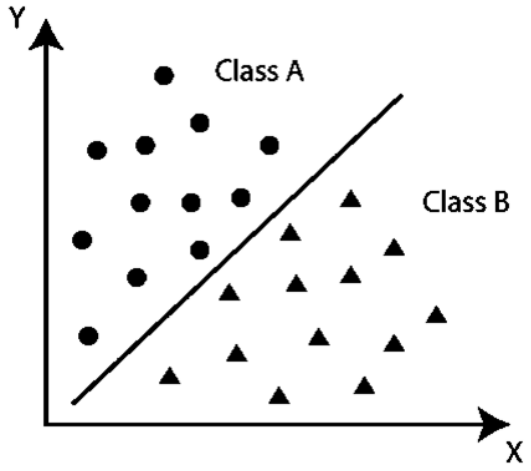


Figure: Binary Classification

Binary classification is a type of machine learning task where the goal is to categorize data into one of two distinct types. This classification problem is fundamental in supervised learning, where a model is trained on labeled data to predict categorical outcomes.

In binary classification, the output is binary, meaning there are only two possible outcomes. For example:

- A coin flip resulting in head or tail: Outcome= Head or Tail
- Categorizing an email as spam or not spam: Outcome= Spam or ham
- Deciding on whether or not to offer a loan to a bank customer: Outcome = yes or no.
- Evaluating the risk of cancer: Outcome = high or low.
- Predicting a team's win in a football match: Outcome = yes or no. etc.

Popular Algorithms for Binary Classification

- K-Nearest Neighbors (KNN)
- Logistic Regression *
- Support Vector Machine *
- Decision Trees
- Naive Bayes

Mathematically,

Given an input feature vector x= (x_1, x_2 ..., x_p), binary classification aims to learn a function f(x) that predicts the class label y∈ {0,1}.

Decision Rule

A general form of the decision function is:

$$f(x)=\begin{cases}1 \; if \; g(x) \geq T \\ 0 \; if \; g(x) < T\end{cases}$$

where g(x)$g(\mathbf{x})$ is a scoring function (e.g., probability or score) and 'T' is a threshold, often set to 0.5.

Example: Linear Classifier

A common example is a linear classifier where g(x) = $\frac{1}{1+e^{-z}}$, with z = $\beta_0 + \sum_{j=0}^{p} \beta_j \, x_j$

Here, σ(z) is the sigmoid function converting the linear combination z into a probability between 0 and 1 (logistic regression). The predicted class is 1 if σ(z)≥0.5, else 0.

2.3.1.2. Multi-class classification

For Example:

It can also be used when analyzing an individual's mood into more than positive or negative. Instead, it will use multiple categories such as, happy, sad, depressed, excited, etc. There is no limit to the number of classes used in multiclass classification.

The most common algorithms which are used for Multi-Class Classification are:

- K-Nearest Neighbors (KNN)
- Naive Bayes
- Decision trees

- Gradient Boosting
- Random Forest
- SVM with One Vs One or One Vs Rest strategies

One Vs Rest – The main task here is to fit one model for each class which will be versus all the other classes

One Vs One – The main task here is to define a binary model for every pair of classes.

Mathematically,

Let, an input feature vector x= (x_1, x_2,…… xp), the task is to predict the class label y∈{1, 2,...,K}, where K≥3 is the number of classes.

Model Output

A multiclass classifier typically outputs a vector of scores or probabilities for each class:

p= (p_1, p_2 ,..., p_K), where p_k=P (y=k |x, $\sum_{k=1}^{k} p_{k=1}$

The predicted class is the one with the highest probability:

y'= argmax p_k

Example:

A common approach to produce these probabilities is the softmax function, which generalizes the sigmoid function used in binary logistic regression:

$$P_k = \frac{e^{z_k}}{\sum_{i=0}^{n} e^{z_j}} \text{ where } z_k = \beta_{k0} + \sum_{i=0}^{n} \beta_{kj} x_j$$

Here, z_k is the linear score for class k, and β_{kj} are the model parameters for class k.

By: Er. Mukunda Paudel, M.Sc. CE
Email: paudelmuku@gmail.com