A Beginner's Guide to Machine Learning

Dr. Anirban Dasgupta

February 11, 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Machine Learning

The areas of machine learning (ML), artificial intelligence (AI), and data science are often used synonymously. However, these terms, although closely related, have different meanings. Let us first ponder over the term data science. Data science, in simple terms, is the application of scientific techniques to obtain meaningful information from the data. Now, when a human looks at some data and tries to draw some inference out of it, we call it some form of intelligence. Similarly, if a computer system can look at the data to draw some inference of it, this form of intelligence is not an inherent form of intelligence, rather it is programmed to do so. Hence, we call this property as artificial intelligence.

This intelligence can be explicitly programmed by a human programmer, where the machine responds to a set of rules. Such kinds of AI systems were prevalent in the early days of AI, and are termed as rule-based systems. When the nature of data became more complex, there was a necessity for the machine to automatically learn the rules from the data and the problem, instead of being explicitly programmed. This form of AI came to be popular as machine learning.

Hence, machine learning (ML) is a type of artificial intelligence (AI) that provides computers or computing systems the ability to automatically learn and improve from experience without being explicitly programmed. Applications of ML include self-driving cars, face recognition, speech recognition, effective web search, and a vastly improved understanding of the human genome.

ML may be broadly divided into the following three main areas, based on the nature of the problem.

- supervised learning
- unsupervised learning
- reinforcement learning

1.1 Supervised Learning

In supervised learning, the data samples (X_i) are trained along with their class labels (y_i). Here the subscript i is the i^{th} data point, and hence i varies from 1 to M, where M is the total number of data samples. As an examples, X_4 is a data-sample which

has a corresponding output label y_4. The X_4 may have a single value, but is usually a vector of values.
So, in general, the training data fed to a supervised learning algorithm is of the form (X_i, y_i). The objective is to find a function f such that

$$f(X_i) = y_i \tag{1.1}$$

This function implies that if a data point X_i is applied as an input to a supervised learning algorithm, it should output its corresponsing y_i. This set of values X_i for any given i is often called features or attributes or feature vector.
The data points for a given i may be written as $X_i = [x_1, x_2, \ldots x_N]$, where N is the number of features. This equation is almost always an ill-posed problem. This means it is difficult to find such an analytic function $f(.)$ which can fully satisfy the equation for all the data points. Hence, supervised learning algorithms tries to model the function with the least possible discrepancy among the training examples. Supervised learning problems may be divided as

- **Classification** $[y_i \, \epsilon \{\text{finite set}\}]$: Classification is the problem of identifying to which category the data point belongs to, among a finite available categories. The set y_i is hence discrete. The classification problem may be binary or multi-class. Examples of supervised learning problems are facial expression classification, gender classification, handwritten digit classification, etc. An example can be, given a set of features X_i, classify it as good $(y_i = 0)$, average $(y_i = 1)$ or bad $(y_i = 2)$.

- **Regression** $[y_i \ \epsilon \ \Re^D]$: Regression is a statistical process for estimating the relationships among variables. Hence, y_i is not discrete. Examples include predicting weight based on height, marks based on hours of study, etc.

The data obtained for training can be obtained through several means such as using sensors, or some survey, or any other standard means. In all such cases, the labels for the data y_i is called ground truth. The task of the supervised ML algorithms is to map the features to the ground truth. In many cases, the ground truth may not be available for training. In such cases, supervised learning cannot be performed. Some such problems fall under the category of unsupervised learning.

1.2 Unsupervised Learning

Unsupervised learning problems have data without any labels. The objective is to find any pattern from the data. Some well-known unsupervised learning problems are

- **Clustering**: Clustering is the task of dividing the data points into a number of groups such that data points in the same groups are more similar to each other, while dissimilar to the data points in other groups. Given a set of data points, a clustering algorithm attempts to assign each data point into a specific group. One example of clustering can be to cluster a set of 100 individuals based on their heights and weights as features. One possible solution can be

to form four clusters *viz.* tall-fat people, tall-slim people, short-fat people, and short-slim people. However, the number of clusters are dependent on the actual data.
Some popular clustering algorithms are K-Means Clustering, Mean-Shift Clustering, Expectation–Maximization (EM) Clustering, Hierarchical Clustering.

- **Density Estimation**: Density refers to the probability density function (pdf) of a random variable. Density estimation is the use of statistical models to find an underlying probability distribution that gives rise to the observed variables. For example, given a specific feature, such as weight of 100 individuals, finding which is the best possible pdf of the population.
Some popular examples of density estimation algorithms are Kernel Density Estimation, Mixture models, etc.

- **Dimensionality Reduction**: In many cases, the number of features is huge, and often do not contribute to the ML problem. This requires reducing the number of features to a few useful discriminative features. The task of dimensionality reduction is to find a smaller set of features that captures the essential variations or patterns of the observed data. This smaller set of features may be just a subset of the original features, or it may be a set of new features that has better performance for a given ML problem. Some popular examples of dimensionality reduction include principal component analysis, autoencoders, etc.

1.3 Reinforcement Learning

There are certain problems, where the learning is based on the actions of the ML model. For example, humans interact with the environment and learn new things based on experience. In such cases, the human evolves rather being trained on some labelled data. This category of ML problems falls under reinforcement learning. Here, an agent takes actions in an environment and based on its action, there is either a reward or a punishment, which is fed back into the agent. There are two types of reinforcement learning *viz.* positive and negative.

1.4 Other Types

There are other variations of ML problem categories which may be

- **Semi-supervised:** In such problems, the label may be missing for some samples, while the remaining may be labelled.

- **Active Learning:** It is like semi-supervised in which a learning algorithm is able to interactively query the user to obtain the desired labels at new data points.

Table 1.1: Example of a dataset illustrating features and instances

Name	**Height (cm)**	**Weight (kgf)**	**Age (yrs)**	**Gender**
Piyush	160	78	32	M
Puja	172	80	31	F
Ramesh	164	74	19	M
Meenakshi	158	66	22	F

1.5 Features

A feature is a representation of the data. Suppose, a dataset contains the information of certain individuals such as name, height, weight, age, etc. In this case, the name, height, weight, etc. can be taken as features of the dataset. Hence, one can say that are features are the basic building blocks of datasets. Features are not necessarily unique for a given dataset. It depends on the programmer to select the features based on the ML problem.

Features are also called attributes. The basis of which an ML algorithm makes a decision is the feature or attribute of the dataset. All features of the same dataset may not be useful in different ML problems. For example, if a dataset contains data of some individuals, and the features are name, height, weight, age, sex, marital status, income, intelligence quotient, etc., then attributes such as height, weight, age may be useful for a medical diagnosis problem, while income, intelligence quotient, age may be useful for a job selection problem.

1.5.1 Feature Types

Features can be either numerical or strings. Examples of numerical features are Height, Weight, Age, Income, Intelligence Quotient, etc., while features such as Name, Gender form examples of string features. Numerical features can also be categorical or continuous. For example, weight of a person can be provided as a continuous value in a range such as 40-100 kgf, or as categorical such as [40-60], [60-80], [80-100].

1.5.2 Feature Representation

Since ML algorithms require a lot of numerical computations, such operations cannot be performed on string-valued or categorical features. Hence, these features need to be converted into a numerical representation. One example can be the mapping of gender values from string to numerical as: Male→ 0, Female→ 1. Similarly, all non-numerical or categorical features can be mapped to numerical representation through such suitable mappings.

1.5.3 Feature Scaling

The numerical features may not have the same ranges, and some algorithms based on gradient descent or distances may not work properly unless the features are scaled to a common range, such as 0 to 1. Feature scaling is performed to bring features with different ranges to a common scale. The common methods of feature scaling are:

- Standardization: Standardization is another scaling technique where the feature values are centred around the mean with a unit standard deviation. The scaled features x_s are obtained by subtracting each feature value x by the mean value, μ and divinding with the standard deviation σ, as

$$x_s = \frac{x - \mu}{\sigma} \tag{1.2}$$

- Normalization: Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. This is obtained by subtracting the feature values from the minimum value $\min x$, and dividing the result by the range, $\max x - \min x$.

$$x_s = \frac{x - \min x}{\max x - \min x} \tag{1.3}$$

  Hence, it is also known as Min-Max scaling.

1.5.4 Feature Vector

Once the numerical features are obtained, they can be grouped together to form a vector, known as the feature vector. All such possible feature vectors form a vector space called the feature vector space. For N features, the dimension of the feature vector space is N, which is also the length of the feature vector.

1.5.5 Features and Instances

An instance is a sample in the training data. An instance is described by a number of attributes or features. An instance for the earlier example can be a specific person together with his or her attributes.

An instance may be labelled or unlabelled. For a labelled instance, a special attribute is the class label that defines the class this instance belongs to.

The features and instances can be arranged in the form of an $M \times N$ matrix called the feature matrix, with M instances and N features, as shown in Table 1.1. In this table, we have five features ($N = 5$) *viz.* Name, Height, Weight, Age, and Gender, denoted by the columns of the table. We also have data of four candidates, hence, representing four instances ($M = 4$). So, the rows of the matrix are the instances, while the columns represent the features.

1.5.6 Feature Reduction

The choice of features hugely impacts the performance of the model. Increasing the number of features may not lead to an improvement in the performance of the model. Sometimes, more number of features may reduce the model performance. This is known as the curse of dimensionality. Hence feature reduction is essential, not only for better model performance but also to reduce the computational burden. Feature reduction can be either feature selection or feature extraction.

Feature Selection

Feature selection involves selecting a subset of the available feature set to be used in learning and classification. Mathematically, it can be expressed as the transformation of initial feature set X to a new feature set Y $X \rightarrow Y$, where Y is a subset of X, $Y \subset X$.
Irrelevant or partially relevant features can negatively impact model performance. So the objective of feature selections is to remove

- irrelevant features: which are not useful for the specific problem. Considering the earlier example, the feature selection will only select the relevant features such as height, weight, age may be useful for a medical diagnosis problem, while income, intelligence quotient, age for a job selection problem.

- correlated features: if two or more features are highly correlated, then only a single feature can be used, as the other features which have high correlation with this single feature, are irrelevant. For example, if an instance has three features, and one of them is a linear combination of the other two, then only two features should be used instead of three.

It is important to minimize the number of features to increase the performance of the learning and future classifications, in terms of processing time and memory required. It is also vital to remove irrelevant features for increasing the classification accuracy, as some ML algorithms cannot cope well with irrelevant features. Hence, feature selection and data cleaning should be the first and most important step. The main advantages of feature selection are:

- **Reduces Overfitting:** Overfitting is a concept where the ML model tries to memorize the training data, instead of trying to generalize. Hence, redundant features would force the model to learn irrelevant things along with useful things. This is like learning weight and height as features, along with percentage score, for selecting a software engineering job. So, less redundant data implies less opportunity to make decisions based on noise or irrelevant information.

- **Improves Accuracy:** Less misleading data will enhance the model accuracy during testing.

- **Reduces Training Time:** Fewer features reduce algorithm complexity, as there will be less number of computations, and algorithms train faster.

Thus, it is evident that selecting specific useful features from the original feature set is beneficial. But what if the original features set is large and is not directly providing sufficient discriminating property. For example, suppose an image of size 50×50 is an instance, where each individual pixel is treated as a feature. In such a case, the feature set is large having a size of 2500, and the individual pixel intensity may not have sufficient discriminating power. In such cases, feature selection may not be the best choice. This problem can be solved by transforming the original feature set into a new discriminating feature set. This process is called feature

extraction.
So, determining a subset of the initial features is called feature selection, while transforming into a new feature space is feature extraction.

Feature Selection Methods The two most common feature selection techniques are:

- Filter Methods - unlabelled data
- Wrapper Methods - labelled data

Filter Methods Filter methods find the relevance of features, when the data is unlabelled. Some popular filter methods used are as follows.

- High Correlation Filters: Correlation is a measure of how close two vectors are. A positive correlation between vectors indicates that an increase in one vector implies proportional increase in the other vector, and vice versa. A negative correlation indicates that an increase in one vector implies a proportional decrease in the other and vice versa. A correlation close to zero denotes the vectors are uncorrelated and they are independent. It is not useful to select two or more correlated features, as they provide similar information. Hence correlated features are discarded in this method, by just keeping one feature which is less correlated with the other features.
- Low Variance Filters: If the values in a feature does not have much variations, then it is less likely to contribute to the ML problem. For example, if a dataset of individuals has very similar values of age, it is less likely that age would be of significant importance for any ML problem. Hence low variance filters remove such features with less variance.
- Missing Value Ratio: In many cases while obtaining a dataset, certain features have missing values, and if the ratio of missing values is high, it is unlikely that the feature will be useful, and can be discarded.

Wrapper Methods Wrapper methods are used when the dataset is labelled. In these methods, statistical tests are used to select those features that have the strongest relationship with the output label. Examples of such statistical tests include chi-squared (χ^2) test. Such tests provide a sorting of features which signifies the feature importance, specific for the labels. The higher the feature importance score, the more important or relevant is the feature towards the output label. The following are the popular wrapper methods.

- Forward Feature Selection: This method starts with zero features, and gradually adds the most relevant feature successively in an iterative manner. This process continues until addition of a new feature does not improve the performance of the model.

- Backward Feature Elimination: This method starts with all the features, and gradually eliminates the most irrelevant feature in each iteration. This process continues until removal of a new feature does not improve the model performance.

Feature Extraction

Feature extraction transforms an initial set of features X into a new set of derived features Y, intended to be more informative and non-redundant. This transformation $X \to Y$ is often linear and can be given as

$$Y = W^T X \tag{1.4}$$

Here the matrix W is the transformation matrix of size $N \times P$, where $P < N$. As such, the initial feature space is reduced from N dimension to P.
Some common algorithms for feature extraction are as follows.

- Principal Component Analysis (PCA): This method transforms the feature space onto a new space such that one axis in this transformed space represents the maximum variance in the data. This axis forms the first feature known as the first principal component. The second principal component is another axis which is orthogonal to the first principal component, and has the second largest variance in the data. In this manner, all the principal components are formed. The axes with minimal variances are discarded, and hence the transformed feature space is a reduced space.

- Independent Component Analysis (ICA): The ICA method focuses on independence, unlike PCA which focuses on maximizing the variance of the data points, .

- Linear Discriminant Analysis (LDA): The PCA and ICA techniques are useful for unlabelled data. For labelled data, LDA is superior, as it reduces the feature space based on the labels instead of merely the features. As such, LDA is also used as a classification method. The LDA technique attempts to maximize the between-class variance and minimize the within-class variance using a linear discriminant function.

- t-distributed Stochastic Neighbor Embedding (t-SNE): The t-SNE works on unlabelled feature space to project data into two dimensions primarily for visualization purposes. The transformation is often non-linear. The aim is to keep similar data points close together in the lower dimensional space.

- Autoencoders: These are neural network based models which tries to learn the input itself at the output, by passing from a high-dimensional input layer to a low-dimensional latent layer. The latent layer forms the new feature space.

1.6 Data Visualization

In implementing most of the ML algorithms, each data point having N features can be represented as a point in the N-dimensional feature space. If the number of features are less than four, data visualization is possible by using Cartesian co-ordinate system, with the features forming the axes. For visualization, the features need to be numerical and scaled, although in certain cases, non-numerical features can also be visualized.

Data visualization is important as it can help to know the nature of the data, and ease the selection of the algorithm for the task. The most common visualization tool is a scatter plot. A scatter plot is a plot in the Cartesian co-ordinate system to display values upto three features for a dataset. A scatter plot can suggest various kinds of correlations between variables with a certain confidence interval. A sample plot is shown in Fig. 1.1, which reveal that weights and heights of most individuals have positive correlation. A 3-dimensional plot is shown in Fig. 1.2, with age as another feature. The visualization also helps in observing the clusters such as short-thin people and tall-heavy people.

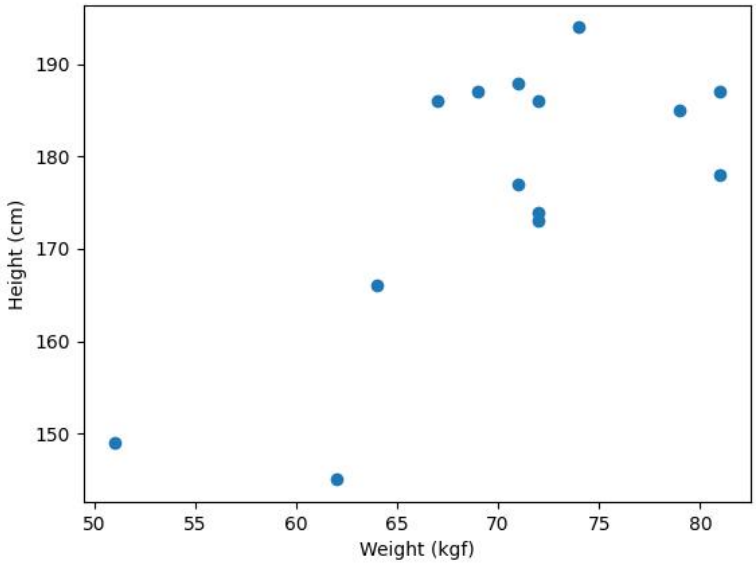


Figure 1.1: A sample 2D scatter plot showing weights and heights of 14 individuals

1.7 Model Validation

The data visualizations provide a reasonable insight into what kind of ML model can be better for the given dataset. The next step is to select the appropriate model and train it. Training is the process of using the data to get the best estimation of the model parameters. Once a model is trained, it needs to be validated. Usually, the data is divided into three sets *viz.* training, validation and testing. Sometimes the model trains well on the training set, but do not perform well on the validation

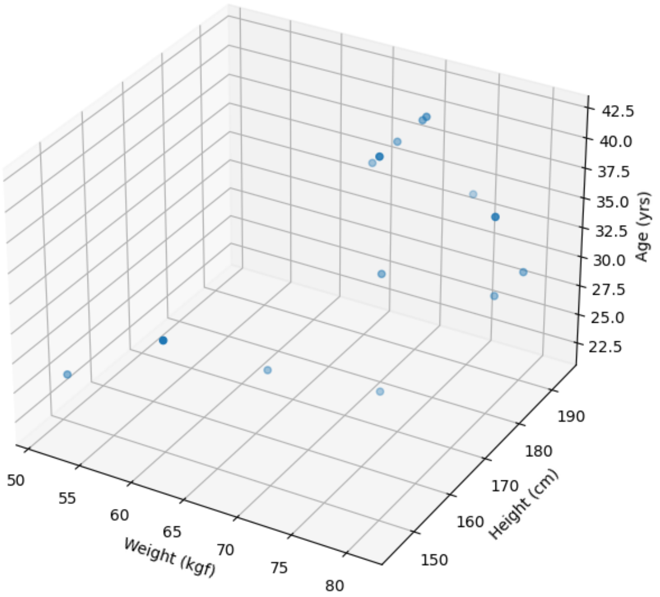


Figure 1.2: A sample 3D scatter plot showing weights, heights and ages of 14 individuals

set. This phenomenon is referred to as overfitting. The following are the popular validation methods used.

1.7.1 Holdout Validation

In hold-out validation, the data is split up into 'train' and 'validation' sets.

1.7.2 Cross-Validation

It is a resampling procedure used to evaluate the model.

K-Fold Cross-Validation

In K-Fold cross-validation, the data is divided into k subsets. One of the k subsets is used as the validation set and the other k-1 subsets as the training set.

Stratified K-Fold Cross-Validation K-Fold Cross Validation technique will not work as expected for an Imbalanced Data set. The data is restricted such that each fold contains approximately the same strata of samples of each output class as the complete.

Leave-P-Out Cross-Validation

In this approach we leave out p data points out of training data out of a total n data points, then n-p samples are used to train the model and p points are used as the validation set. This is repeated for all combinations, and then the error is averaged.

Leave-One-Out Cross-Validation In this case, P=1. It has zero randomness. The bias will be lower. However, this method is exhaustive and computationally expensive.

Monte-Carlo Cross-Validation

In this method, a random subset of data is selected as test data without replacement, while the remaining is used for training. This process is repeated.

Rolling Cross-Validation

The method is useful for cross-validating the time-series models on a rolling basis. The method starts with a small subset of data for training purpose, followed by forecasting for the later data points and then checking the performance for the forecasted data points. The same forecasted data points are then included as part of the next training dataset and subsequent data points are forecasted again.

1.8 Performance Metrics

The validation process requires certain metrics to evaluate how well the model is performing. There are various metrics which are used for the purpose. The performance of a specific algorithm on a dataset is highly dependent on the choice of metric. Certain metrics have better suitability over specific problems than other.

1.8.1 Classification Problems

The following performance metrics are used to evaluate predictions of classification problems.

Confusion Matrix

A confusion matrix is a table with two dimensions *viz.* “Actual” and “Predicted”. The diagonal entries show how much of a given class is correctly predicted, while off-diagonal entries show incorrect predictions. Hence, a diagonal confusion matrix shows perfect classification, and is desirable.

Binary Classification Problem For these problems, the confusion matrix is a 2×2 matrix, with the entries being “True Positives (TP)”, “True Negatives (TN)”, “False Positives (FP)”, “False Negatives (FN)”, as shown in Table 1.2. In this table, $\mathbf{p'}$ and $\mathbf{n'}$ denote the actual positive and negative classes respectively, while $\mathbf{p}$ and $\mathbf{n}$ denote the predicted positive and negative classes respectively. These terms are defined as follows for a two-class classification problem, with a positive class assigned ‘1’ and a negative class assigned a ‘0’.

- True Positives (TP): It is the number of cases when both actual class and predicted class of data point is 1.

Table 1.2: Confusion matrix format for binary classification problems

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

- True Negatives (TN): It is the number of cases when both actual class and predicted class of data point is 0.

- False Positives (FP): It is the number of cases when actual class of data point is 0 and predicted class of data point is 1.

- False Negatives (FN): It is the number of cases when actual class of data point is 1 and predicted class of data point is 0.

These values result in two more important metrics known as the true positive rate (tpr) and false positive rate (fpr), defined as

- true positive rate (tpr): is the probability that an actual positive will test positive. It is computed as

$$tpr = \frac{TP}{TP + FN} \tag{1.5}$$

- false positive rate (fpr): is the probability of a negative being wrongly classified as a positive. It is computed as

$$fpr = \frac{FP}{FP + TN} \tag{1.6}$$

Multi-class Classification Problem In these problems, the confusion matrix is a C×C matrix, where C is the number of classes. The higher the values in the diagonal entries and lower the values of the off-diagonal entries, the better is the classification method, and vice versa.

Accuracy

It is most common performance metric for classification algorithms. The accuracy may be defined as the percentage of correct predictions made. We can easily calculate the percentage accuracy α from the confusion matrix with the help of following formula as follows.

$$\alpha = \frac{TP + TN}{TP + FP + FN + TN} \times 100\% \tag{1.7}$$

For multi-class problems, the accuracy is obtained as the fraction of diagonal entries. Accuracy is not always a very good measure. For example, if class 0 has 100 members and class 1 has 900 members, a classifier which labels every data as class 1 still has an accuracy of 90%.

This issue is solved using metrics known as the classification report (CR). The CR consists of the scores of Precisions, Recall, F1 score. The Precision and Recall are very important in information retrieval search.

Precision

Precision is the ratio of correctly predicted positive observations TP to the total predicted positive observations P'. High precision relates to the low false positive rate.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{P'} \tag{1.8}$$

Recall

Recall is the ratio of correctly predicted positive observations TP to the all observations in actual positive class P.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{1.9}$$

F1 score

F1 score is the harmonic mean of Precision and Recall. This score will give us the harmonic mean of precision and recall. The harmonic mean is used instead of arithmetic mean because it punishes the extreme values. This score takes both false positives and false negatives into account. F1 score is usually more useful than accuracy, especially if class distribution is uneven. The best value of F1 would be 1 and worst would be 0. The F1 score is calculated with the help of following formula.

$$F1score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{1.10}$$

Receiver operating characteristic

The receiver operating characteristic (ROC) curve is a plot between the *tpr* and the *fpr* of a binary classifier, with varying discrimination thresholds. The Area Under Curve (AUC) of the ROC curve is a performance metric, varying from 0 to 1. The AUC measures the separability, with a value closer to 1 indicates higher separability.

1.8.2 Regression Problems

The various performance metrics that can be used to evaluate predictions for regression problems are as follow.

Mean Absolute Error (MAE)

It is the simplest error metric used in regression problems. It is the average of the sum of the absolute differences between the predicted and actual values, across all the N test instances. The following is the formula to calculate MAE.

$$MAE = \frac{1}{N} \sum |Y - \hat{Y}| \tag{1.11}$$

Here, Y = Actual Output Values and $\hat{Y}$ = Predicted Output Values. The MAE does not indicate the direction of the model i.e. no indication about under-performance or over-performance of the model.

Mean Squared Error (MSE)

MSE is like the MAE, but the only difference is that the it squares the difference of actual and predicted output values before summing them all instead of using the absolute values.

$$MAE = \frac{1}{N} \sum (Y - \hat{Y})^2 \tag{1.12}$$

Coefficient of Determination (R^2)

The coefficient of determination (R^2) is a statistical measure of how close the data are to the fitted regression line. It provides an indication of the goodness or fit of a set of predicted output values $\hat{Y}$ to the actual output values Y. It is defined as the ratio of explained variation to the total variation in the data, with $\bar{Y}$ being the data mean.

$$R^2 = 1 - \frac{\sum (Y - \hat{Y})^2}{\sum (Y - \bar{Y})^2} \tag{1.13}$$

R-squared cannot determine whether the coefficient estimates and predictions are biased.

1.8.3 Clustering Problems

The performance of a clustering algorithm is examined through cluster validity indices. They are required

- to compare clustering algorithms
- to compare two sets of clusters
- to determine whether random structure exists in the data due to noise

The evaluation is performed with respect to compactness and separation of clusters. A measure of compactness is within-class variance. A measure of separation is between-class variance.

Silhouette Coefficient

Silhouette coefficient or silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1. The silhouette coefficient, ρ is computed as

$$\rho = \frac{(b-a)}{\max(a,b)} \tag{1.14}$$

Here, a is the average distance between each point within a cluster, while b is the average distance between all clusters. A coefficient close to 1 indicates that the clusters are well apart from each other and clearly distinguished. A score of 0 means the clusters are indifferent, or the distance between clusters is not significant, while a score close to -1 indicates that the clusters are assigned in the wrong way.

Dunn index (DI)

DI is based on the clustered data itself. The DI identifies sets of clusters that are compact, with a small variance between members of the cluster, and well separated, where the means of different clusters are sufficiently far apart, as compared to the within cluster variance.

$$DI = \frac{\min b_{ij}}{\max a_k} \tag{1.15}$$

Here, b_{ij} is the mean cluster distance of the cluster i from cluster j, while a_k is the mean cluster distance of cluster k. Higher the DI value, better is the clustering. The number of clusters that maximizes DI is taken as the optimal number of clusters k.

1.8.4 Density Estimation

The performance of density estimation algorithms can be estimated if the true density of the data is known. The true density is compared with the estimated density to find the goodness of estimation.

Kullback–Leibler (KL) Divergence

The Kullback–Leibler divergence (KLD) measures the distance between two distributions. It is also called relative entropy. The KLD is defined as

$$KLD(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \tag{1.16}$$

Here, the probability distributions P and Q denote the true and estimated densities, on the same probability space, $\mathcal{X}$.

Jensen–Shannon Divergence

Jensen–Shannon divergence (JSD) is another method of measuring the similarity between two probability distributions. JSD is a symmetrized and smoothed version of the KLD. It is computed as

$$JSD = \frac{1}{2}KLD(P \parallel M) + \frac{1}{2}KLD(Q \parallel M) \tag{1.17}$$

Here, $M = \frac{1}{2}(P + Q)$.

1.8.5 Dimensionality Reduction

Reconstruction Error

The assessment of dimensionality reduction of a data X is found by reconstructing the original data X from the new data of the reduced dimensional space. The reconstruction error is computed as

$$\epsilon = \sum ||X - \hat{X}||^2 \tag{1.18}$$

Here, $\hat{X}$ is the reconstructed data.

1.9 Loss Functions

A loss function is an objective function, which is to be minimized during the training of the model. The model parameters are finally selected which minimizes the loss function. The loss value implies how poorly or well a model behaves after each iteration of optimization.

1.9.1 Loss Functions for Classification

Binary Cross Entropy Loss

This is the most common loss function used for classification problems with two classes. Binary cross entropy is a measure of the difference of the randomness between two random variables. If the probability of being in the class y is p and not being in class y is $1 - p$, then the loss function $\mathcal{L}$ is defined as

$$\mathcal{L} = -(y \log(p) + (1 - y) \log(1 - p)) \tag{1.19}$$

Categorical Cross Entropy Loss

For a multi-class problem, the loss function is defined as

$$\mathcal{L} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1.20}$$

Here, $p_{o,c}$ is the model's predicted probability that observation o is of class c.

Hinge Loss

Hinge loss was primarily developed for support vector machines for calculating the maximum margin from the hyperplane to the classes. This function penalizes wrong predictions and does not do so for the right predictions.

$$\mathcal{L} = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \tag{1.21}$$

Here, s_j is the true value and s_{y_i} is the predicted value.

Kullback Leibler Divergence Loss

Kullback Leibler Divergence Loss is a measure of how a distribution varies from a reference distribution. Kullback Leibler Divergence Loss of zero means that both the probability distributions are identical. The KL Divergence of a distribution P(x) from Q(x) is given by:

$$\mathcal{L} = \sum P(x) \log \frac{P(x)}{Q(x)} \tag{1.22}$$

1.9.2 Loss Functions for Regression

L1 loss

L1 loss is the same as MAE.

L2 loss

L2 loss is the same as MSE.

Huber Loss

Huber Loss combines the robustness of L1 with the stability of L2, essentially the best of L1 and L2 losses. For huge errors, it is linear and for small errors, it is quadratic in nature.

1.10 Challenges in Machine Learning

It is essential to understand that ML is only suitable for specific problems. Although most of the theories in ML are well developed, ML has many challenges which are being researched. Some of them are as follows.

1.10.1 Bias-variance trade off

It is important to understand the terms bias, variance, overfitting and underfitting, to understand the bias-variance trade-off.

Bias Bias is the average difference between the predictions of a model and the predicted values across the training data. Low bias suggests fewer assumptions about the form of the target function, which often implies memorization. High bias suggests more assumptions about the form of the target function. Examples of low-bias ML algorithms include Decision Trees, k-Nearest Neighbors, and Support Vector Machines, while examples of high-bias ML algorithms include Linear Regression, Linear Discriminant Analysis, and Logistic Regression.

Variance Variance is the average error of the predictions for the validation or test data. Low variance suggests that the training of the model is satisfactory. Low variance ML algorithms include Linear Regression, Linear Discriminant Analysis, and Logistic Regression, while algorithms like Decision Trees, k-Nearest Neighbors, and Support Vector Machines have high variance.

Overfitting

Overfitting is the phenomenon which implies that the bias is low, but the variance is high. This condition implies that the training data is fit too well, unlike the test or validation data. Hence an overfitted model fails to generalize the data. It rather memorizes the data. Overfitting usually happens when a model learns the outliers and noise in the training data, or sometimes the training data is insufficient to generalize the model. This is the reason why overfitting negatively impacts the performance of the model on new data. There are two important techniques that is used to check overfitting, as follows.

- **Resampling technique:** The resampling technique is used to train the model using multiple training sets, like the k-fold cross-validation method. As such, the model is trained and tested k-times on different subsets of training data. This brings a lot of variations in the training data to generalize the model on unseen data.

- **Validation dataset:** As stated earlier, there should be a validation dataset apart from training and test datasets. All three datasets must be mutually exclusive. This validation data provides insight into overfitting and can be used to fine-tune the model before testing.

Under-fitting

Underfitting refers to the condition, when a model that can neither fit the training data nor the test data well.

Good Fit

A good fit should have low bias and variance. However, sometimes increasing the bias helps to decrease the variance, thereby generalizing the model. Hence, a good fit can be achieved by finding a sweet spot between underfitting and overfitting. This can be accomplished by observing the performance of the ML model over time as it learns training data. This is a challenge in ML, which is to have an optimal trade-off between bias and variance.

1.10.2 Natural Language Processing (NLP)

Natural language is the language commonly used for speaking. Natural language processing (NLP) aims to process and analyze large amounts of natural language data using ML. Some common problems in NLP are provided as follows.

Speech-to-text Conversion

In this problem, a speech is input to the model, and the model needs to identify the spoken words and subsequently transform those into text. Such models must be speaker invariant and not prone to background noise. Speech-to-text can be used to aid people with hearing impairment.

Grammar Correction

The problem is to check the correctness of grammar in a specific language from an input text paragraph.

Language Translation

This problem involves the conversion of a paragraph of text from one language to another, without altering the meaning.

Optical Character Recognition

The problem of Optical Character Recognition (OCR) is to determine the corresponding text where the texts appear in printed form in an image. This is specifically challenging for recognizing handwritten texts.

1.10.3 One-shot Learning

In certain cases, few training data is available, which can be even a single training instance. One-shot learning is a classification task where one or a very few examples are used for training.

1.10.4 Object Localization

Object localization is the task of locating the instances of a specific object in an image, by obtaining rectangular bounding boxes covering the instances. The task has challenges such as occlusion, variations in illumination, rotations, etc.

1.10.5 No-Reference Assessment

In certain cases, no references are available to evaluate the performance of an ML algorithm. Usually, two kinds of problems fall in this category.

Quality Assessment

Assessment of quality of a signal or an image is challenging, specifically without any reference signal or image respectively.

Noise Removal

The estimation and subsequent removal of noise level of a signal or an image is challenging, and the assessment of performances of such algorithms is difficult, specifically without any reference signal or image respectively.

1.10.6 Explainability

Explainable AI is the collection of approaches aimed at the understanding and interpretation of the predictions made by ML models. This is to find the step-by-step approach to how the model arrives at a decision. This study also helps one to know the randomness in a classifier if it just arrives at a prediction by mere chance.

These challenges are just the tip of the iceberg. There are plenty more challenges, and ML researchers are working hard on solving these problems specific to the data. It is essential to understand all the fundamental concepts and the tools available in ML to further explore the area and challenges.

Chapter 2

Classification

Supervised learning problems fall under two primary categories - classification and regression, as mentioned earlier. The task of classification is to assign a label to test data among a set of available labels used during the training. A classifier is an algorithm that labels a test data, given a feature vector of the instance of the test data. In this chapter, some popular classification methods are introduced.

The task of training a classifier is to build a model which can translate the feature vector X to an output label y, where the labels can have specific discrete values. The model is usually defined using parameters and hyperparameters. The parameters of an ML model are the trainable values, based on which the class prediction is performed. Hence, parameter values are internal to the model. For example, if the classifier is based on a decision boundary which is a line, the slope and intercept values form the parameters. Hyperparameters, on the other hand, are explicitly specified parameters to control the training process, such as the stopping criteria in an iterative training algorithm or value of the learning rate in a gradient descent algorithm. In neural networks, parameters include the weights and biases, while hyperparameters may include the learning rates, size of convolution filters, etc. Hence, parameters are essential for making predictions, while hyperparameters are useful for optimizing the model.

Classifiers can be generative or discriminative. A generative classifier attempts to learn an ML model that generates the data. This is usually performed by estimating the assumptions and distributions of the model. A discriminative classifier, on the other hand, trains the model parameters based on the observed data. As such, fewer assumptions are made on the data distributions, and such classifiers are heavily dependent on the quality of the data.

There are a huge number of classifiers available in the literature. Most of these classifiers are developed from some base classification methods, and finally customized to suit specific problems. It is a rigorous task to know about all the possible classification methods. However, understanding of the basic classifiers is necessary to apprehend the variants of these classification methods, and also to develop a new classification method, depending on the problem. The following section discusses some of the most important classification methods.

2.1 Logistic Regression

Logistic regression is a classification model which first performs a regression first, followed by classification. The regression is performed with the features as the independent variables and yields a temporary output variable. This temporary output is thresholded accordingly to take one of the discrete class labels. This classification method works best for a binary classification problem. In such problems, the regression function is usually the sigmoid function. This is the reason why the sigmoid is also known as the logistic function. One example where this method can be used is to classify the gender of an individual as male or female, using height and weight as features.
Logistic regression can be either linear or non-linear depending on the nature of regression function.

Linear Logistic Regression In linear logistic regression, each feature $x_i \in X$ is given a weight w_i for $i \in [1, N]$, and a bias w_0 for a dataset having N features. If the problem has C classes, then the weighted sum is thresholded accordingly, such that the mapping $f(\sum_0^{N-1} w_i x_i + w_0) \to y$ happens, where y is the class label for the specific instance. The model parameters are the weights w_i for $i \in [0, N-1]$, and the bias w_0.

Non-linear Logistic Regression In non-linear logistic regression, the function does not input a linear combination of features and weights. Hence the function $f(W, X)$ can be any arbitrary function of the weight vector W and the feature vector X, to obtain the mapping $f(W, X) \to y$.

2.1.1 Training

Let us understand the training using a linear model for a binary classification task. Suppose, there are M training instances, with N features. The first step is to scale the features. The second step is to assign arbitrary weights to the model. The next step is to form the weighted sum of features as $\sum_0^{N-1} w_i x_i + w_0$, where the weights are unknown. This weighted sum is passed through the sigmoid function $f(.)$. The task is to find the weights, which will minimize the absolute error between the predicted value from the function $f(.)$ and the actual label y. This is achieved using an iterative process like gradient descent, as the error is passed through each training instance.

2.1.2 Testing

Once the model is trained and the weights are fixed, the value of the function $f(.)$ with the trained weights is obtained. Since the function $f(.)$ is sigmoid, the value is in the range of 0 to 1. The threshold is usually taken to be 0.5, where values less than 0.5 are assigned class 0, while the remaining as class 1.

2.1.3 Advantages and Limitations

The advantages of using logistic regression are as follows.

- Logistic regression is easy to implement, interpret, and very efficient to train.
- It makes no assumptions about distributions of classes in feature space.
- It performs reasonably well when the dataset is linearly separable.
- Logistic regression is less inclined to over-fitting, but it may overfit in high dimensional datasets.

The main disadvantages or limitations are as follows.

- If the number of observations is much less than the number of features, logistic regression should not be used, otherwise, it may lead to overfitting.
- The linear logistic regression assumes linearity between the dependent variable and the independent variables.

2.2 k-Nearest Neighbors (kNN)

The k-nearest neighbors algorithm (kNN) is one of the simplest ML algorithms. This algorithm can perform both classification and regression, however, its use as a classifier is more popular. It is a non-parametric method of classification, and does not involve any training process. Hence, it is also a form of lazy learning or instance-based learning, as there is no training process. Most of the computation happens during the testing phase. As the name suggests, the classification uses information from k nearest neighbors. A majority vote of its neighbors, decides the assignment of the class label. The value of k is usually an odd number, for binary classification problems, to discard cases of a tie. The value of k highly influences the performance.

In this method, the first step is to define a labeled data as of the form (X_i, y_i), where $X_i \in \mathbb{R}^D$, which is properly scaled. Now given an unknown data X_j, it has to be labelled based on the labels of its nearest neighbors. The algorithm performs a voting of the labels of k-NN and the winning label is assigned to the unknown data. A limitation of the k-NN algorithm is that it is highly sensitive to the local structure of the data. This issue can be solved by assigning weights of $\frac{1}{d}$ to each neighbor, where d is the distance to the neighbor.

k-NN classifiers inherently do not output probabilities. Rather they directly output the class label. The conversion into a probability can be carried out using suitable distance transforms. The output of a k-NN classifier is in terms of distance of x to nearest member, e.g. $f(x) = d \in \mathbb{R}^+$.

2.2.1 Testing

The KNN testing algorithm is presented as follows.

- The number of neighbors k is initialized.
- For each data point
  - The distance is calculated from the test data point is computed.
- The computed distances are sorted in ascending order.
- The first k entries are picked from the sorted collection.
- The class labels of these selected k entries are obtained.
- The class label with highest number of entries is selected as the final predicted label.

2.2.2 Advantages and Limitations

The advantages of using kNN are:

- It can learn non-linear decision boundaries when used for classification.
- No training time is required.
- There is only one hyperparameter, which is the value of k. This makes hyperparameter tuning easy.
- Addition of new data to the dataset does not consume extra training time.

The disadvantages of kNN are:

- The testing time increases with the size of the dataset, and hence is not suitable in real-time applications with large data size.
- It is not great for large datasets, since the entire training data is processed for every prediction. Time complexity for each prediction is $O(MN \log k)$ where M is the dimension of the data, N is the size or the number of instances in the training data.
- The kNN assumes equal importance to all the features.
- The kNN is sensitive to outliers.

2.3 Linear Discriminant Analysis (LDA)

Linear discriminant analysis (LDA) transforms the original feature set onto a new set of features using a linear combination of original features. In this newly transformed feature space, the data has higher between class variability and lower within class variability. Hence, LDA is useful in separating two classes. The cost function J can be represented as follows.

$$J = \frac{\mu_1 - \mu_2}{s_1^2 + s_2^2} \tag{2.1}$$

Here, μ_1 and μ_2 are the means of the two classes, while s_1 and s_2 are the standard deviations of the two classes. The cost function, hence, tires to maximize the difference in means, while minimise the the sum of within class variance.
LDA is also used for dimensionality reduction in the case of labelled data. A fundamental assumption of the LDA method is that the features are normally distributed to minimize the expected error.

2.3.1 Illustration

Suppose there are two classes of data points which are not linearly separable as shown in Fig. 2.1. This means, there are no straight lines that can divide the plane and separate the two classes of the data points completely.
LDA attempts to reduce the 2D feature space into a 1D feature space in order to maximize the separability between the two classes. Here, LDA uses both the axes to create a new axis, and projects data onto this new axis. There are two criteria used by LDA to create this new axis, as follows.

- Between class: Maximize the distance between means of the two classes.
- Within class: Minimize the variation within each class.

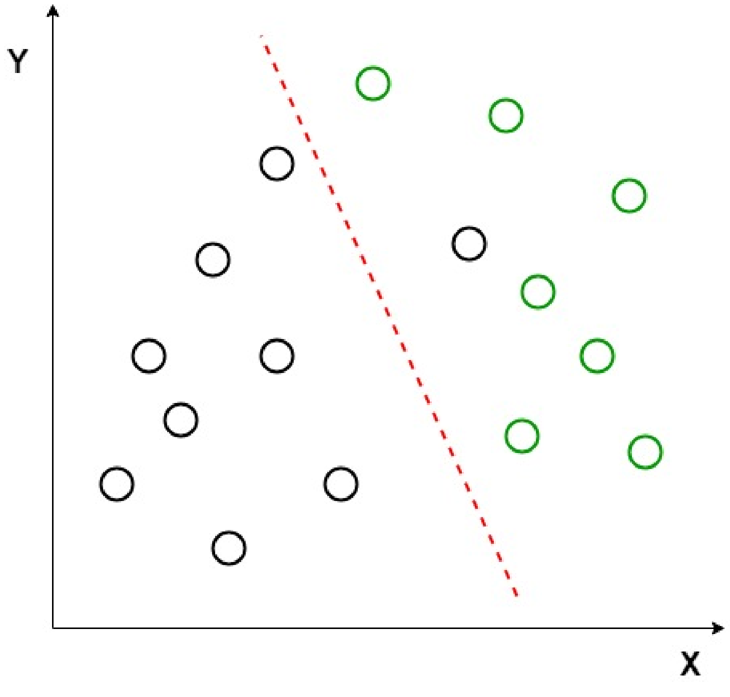


Figure 2.1: A sample 2D plot with two classes which are not linearly separable

In Fig. 2.2, it can be seen that a new axis (in red) is generated and plotted in the 2D graph such that it maximizes the distance between the means of the two classes and minimizes the variation within each class. After generating this new axis using the above-mentioned criteria, all the data points of the classes are plotted on this new axis and are shown in the Fig. 2.3.

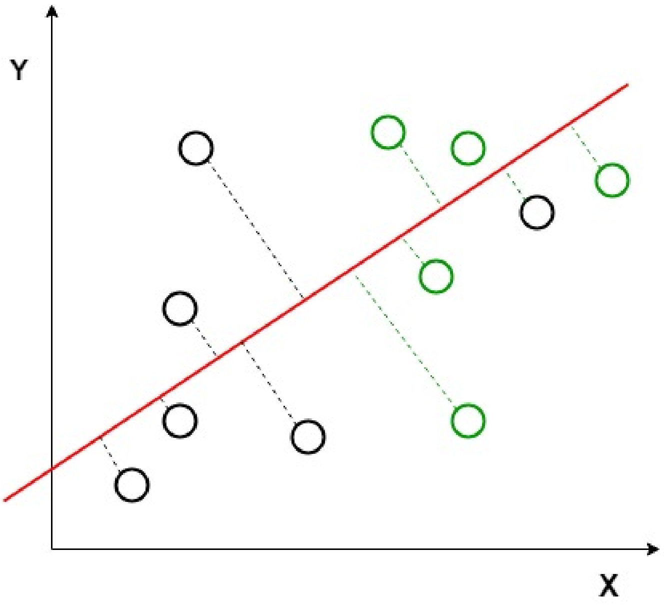


Figure 2.2: Projecting the 2D plot onto a line

2.3.2 Advantages and Limitations

LDA is simple, and fast. However, LDA assumes normal distribution of features, which is not always the case. LDA fails when the mean of the distributions are shared, as it becomes impossible for LDA to find a new axis that makes both the classes linearly separable. In such cases, non-linear discriminant analysis is used.

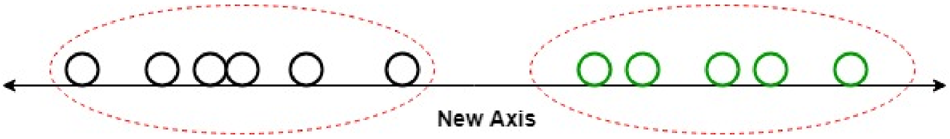


Figure 2.3: Projected point in the new axis

2.4 Decision Trees

A decision tree is a flow-chart-like structure. The method of classification using a decision tree is a hierarchical classification method. This method works well even if the data is not linearly separable. The understanding of the concept of decision tress requires the knowledge of its terminology.

2.4.1 Terminology

There are some common terms used with decision trees, as follows.
Node: A node in a decision tree indicates an attribute based on which a split is done to two or more branches. A node may have an incoming branch and multiple outgoing branches.

Branch: A branch is a directional connection between two nodes. A branch is represented by an arrow. The node at the start of a branch is called a parent node, while that at the end is called a child node. Hence, a parent and a child node are connected using a single branch.
Root Node: A root node is a node which has a splitting, but no incoming nodes. As such, this node represents the entire population. A decision tree has only one root node.
Leaf Node: Nodes that do not split further are called Leaf or Terminal Nodes. Each leaf node holds a class label.
Splitting: When a node branches out into two or more nodes, this process is called splitting.
Decision Node: Any node which splits into further nodes is called a decision node. Hence, all non-leaf nodes are decision nodes. The decision nodes denote a test on an attribute. Each branch represents the outcome of a test.
Pruning: The removal of decision nodes is called pruning. This is the opposite process of splitting.
Sub-Tree: A sub section of entire decision tree is called sub-tree.
The approach is to form a tree and minimize the training error in each leaf of the tree. The target variable is usually categorical or made categorical using some thresholding techniques. If multiple trees can fit the examples, then the smallest tree is selected.

2.4.2 Method

The decision trees use entropy, information gain or Gini impurity index, for splitting the attributes. Entropy is a measure of disorder or impurity in a node. It is computed using the formula

$$H = -\sum_{i=1}^{n} p_i \log p_i \tag{2.2}$$

Here, H is the entropy, when there are i classes having a probability of being selected in the training examples, p_i. The difference in entropies of parent and child nodes gives information gain. The greater the information gain, the greater the decrease in entropy or uncertainty. Hence, information gain is used to select an attribute at a specific node.
Another important metric useful for splitting attributes is the Gini impurity. This index measures the frequency at which any element of the dataset will be mislabelled when it is randomly labelled. The Gini index GI is computed as

$$GI = 1 - \sum_{i=1}^{n} p_i^2 \tag{2.3}$$

The minimum value of the Gini Index is 0, which indicates highest level of purity. For example, if a fair coin is tossed 5 times and the sequence is HHHHH, then $P(H) = p_1 = 1$ and $P(T) = p_2 = 0$. This makes the GI equal to 0, from (2.3). This happens when the node is pure, this means that all the contained elements in the node are of one unique class.

The range of Entropy lies in between 0 to 1 and the range of Gini Impurity lies in between 0 to 0.5. Since entropy contains logarithms to calculate, Gini index is faster to compute.

2.4.3 Training Decision Trees

The training of decision trees is the process of converting the dataset into a tress. There are several algorithms to train a decision tree. The most popular one is the Iterative Dichotomiser 3 (ID3).

ID3 Algorithm The ID3 algorithm is given as follows.

- The ID3 begins with the original set S at the root node.
- On each iteration of the algorithm, it iterates through every unused attribute of the set S and calculates the entropy H and the information gain IG.
- It then selects the attribute which has the smallest entropy or largest information gain value.
- The set S is then split or partitioned by the selected attribute to produce subsets of the data.
- The algorithm continues to recur on each subset, considering only attributes never selected before.

2.4.4 Advantages and Limitations

The advantages of using decision trees are as follows.

- Decision trees does not require feature scaling.
- Missing values in the data also do not affect the process of building a decision tree to any considerable extent.
- The explainability of decision trees is high.

Apart from these advantages, decision trees come with certain limitations.

- Decision trees can be unstable and highly sensitive to small variations in the data. A small noise can result in a completely different tree.
- Decision tree algorithms can be biased with unbalanced dataset, if some classes dominate. It is therefore recommended to balance the data set prior to fitting with the decision tree.
- Decision trees tend to overfit.

2.4.5 Pruning

A major limitation of decision trees is overfitting. A method to reduce overfitting in a decision tree is pruning. Pruning involves removing the branches that make use of features having low importance. This way the complexity of the tree gets reduced. This process also increases its predictive power. Pruning can start either from the root or from the leaves.

2.5 Support Vector Machines (SVM)

A support vector machine (SVM) is a classifier ideally suited for binary classification problems. If a dataset has N features, SVM learns an $N-1$-dimensional hyperplane which divides the N-dimensional plane into two halves, so each half contains either of the classes. For an N-dimensional feature space, the $N-1$-dimensional hyperplane is trained using the data. A test data is checked for which half of the plane it lies, thereby performing the classification. Hence, a 2D feature space is separated by a line, while a 3D space is separated by a plane.

2.5.1 Illustration

Suppose the training data has two features and two classes. One such case can be classifying gender as male or female using height and weight as the features. The data points can be easily represented as a 2D scatter plot as shown in Fig. 2.4. Since the data is linearly separable, it is easy to find a line separating the two classes. However, there are infinite number of lines which can separate the two classes. The objective of SVM is to find the optimal line, that maximizes the distance from the line to the nearest training data point of any class. The nearest training data points of each class are called the support vectors. The distance between the support vectors in the perpendicular direction of the classification line is called the margin.

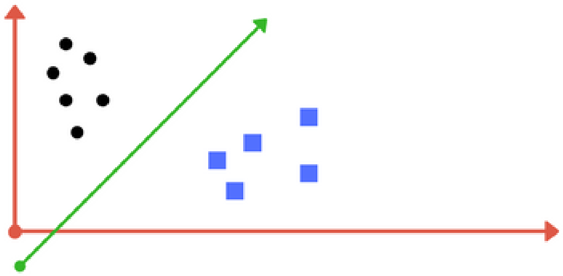


Figure 2.4: A 2D scatter plot containing data of two classes

2.5.2 Linear SVM

A training dataset of M points of the form $(\vec{x}_1, y_1), \ldots, (\vec{x}_M, y_M)$ is given, where the y_i's are either 1 or -1, each indicating the class to which the point $\vec{x}_i$ belongs. Any hyperplane can be written as the set of points $\vec{x}$ satisfying as

$$\vec{w} \cdot \vec{x} - b = 0 \tag{2.4}$$

Here $\vec{w}$ is the normal vector to the hyperplane. The parameter $\frac{b}{\|\vec{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\vec{w}$.
Once the training is complete, the hyperplane parameters $\vec{w}$ and b are fixed. Hence for a test instance $\vec{x}_k$, the classification is as follows:

$$\begin{aligned} \vec{x}_k \cdot \vec{w} + b \geq 0, \quad y_k = +1 \\ \vec{x}_k \cdot \vec{w} + b < 0, \quad y_k = -1 \end{aligned} \tag{2.5}$$

2.5.3 Kernel

In most of the real-world cases, the data is not linearly separable. In such cases, the feature vectors are operated with kernels to make them separable. The most popular SVM kernel functions are:

- *Linear Kernel:* It is the most basic type of kernel, usually one dimensional in nature. This is used when the data is linearly separable. Linear kernel functions are faster than other kernel functions.
- *Polynomial Kernel:* This kernel K is obtained by placing a power d to the the linear kernel, as follows.

  $$K = (\vec{x} \cdot \vec{w} + b)^d \tag{2.6}$$

  It is not as preferred as other kernel functions as it is less efficient and accurate.
- *Radial Basis Function (RBF):* The RBF kernel is defined as

  $$K = \mathrm{e}^{-\frac{\|\vec{\mathrm{x}}-\tilde{\mathrm{x}'}\|^2}{2\sigma^2}} \tag{2.7}$$

  Here, $\|\vec{x} - \vec{x'}\|^2$ is the squared Euclidean distance between the two feature vectors, while σ is a free parameter. The RBF kernel decreases with distance and ranges between 0 and 1. It is one of the most preferred and used kernel functions in SVM for non-linear data.

2.5.4 Training

The training involves finding the optimal values of the parameters defining the hyperplane. These parameters are the $\vec{w}$ and b.

2.5.5 Advantages and Limitations

The advantages of using SVMs are:

- SVM works relatively well when there is a clear margin of separation between classes.

- SVM is more effective in high dimensional spaces.
- SVM is effective in cases where the number of dimensions is greater than the number of samples.
- SVM is relatively memory efficient.

The limitations include:

- SVM algorithm is not suitable for large data sets.
- SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.

2.6 Bayesian Learning

Bayesian learning is the use of data to construct statistical models based on Bayes' Theorem , which is stated as follows.

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \tag{2.8}$$

Here, θ is the distribution, while x is the data. The term $p(\theta|x)$ defines the posterior distribution, which says the probability of a distribution given the data. The term $p(x|\theta)$ gives the likelihood that the data x is produced by distribution θ, while $p(\theta)$ and $p(x)$ are the respective prior probabilities to observe the distribution and data. In other words, Bayes theorem can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}} \tag{2.9}$$

2.6.1 Naive Bayes Classifier

A Naive Bayes classifier is a probabilistic ML model, based on the Bayes' theorem. They are named naive because of their strong independence assumptions between the features, given the class label.

2.6.2 Naive Learning

The training step is to find out the posterior probabilities of each class for a given feature vector. The assignment of a specific class in the testing phase is achieved using the Maximum A Posteriori (MAP) estimate. The MAP estimate is the class label with the maximum posterior probability for a given feature vector set. Hence, it is a generative classification model.

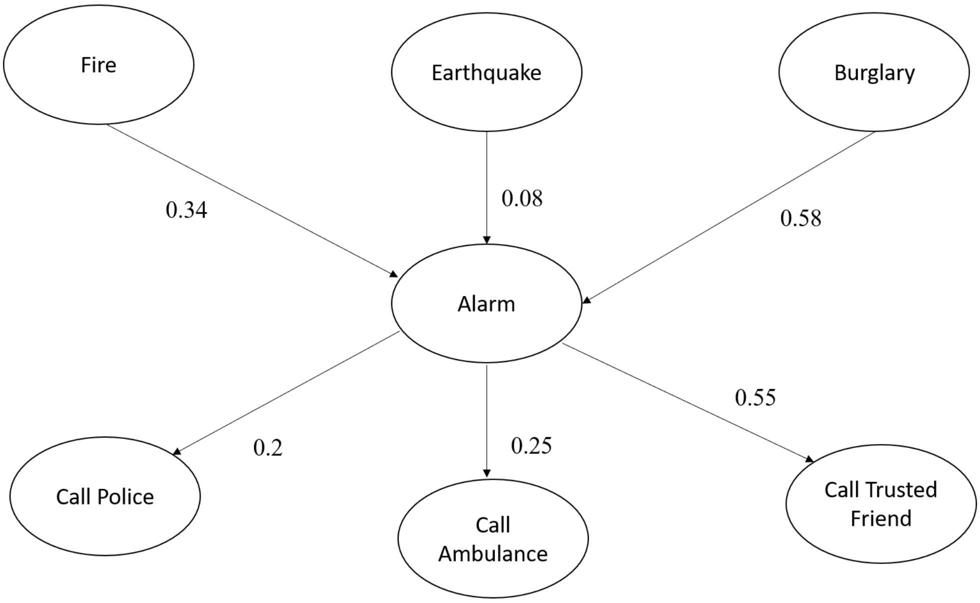


Figure 2.5: Belief network for report of leaving

2.6.3 Advantages and Limitations

The advantages of Naive Bayes classifier are:

- It is very easy and fast to work with the Naive Bayes classifier.
- It is robust to noisy data to some extent.
- It works well even with complex and high dimensional data.

The limitations include:

- Naive Bayes classifiers suffer from the zero frequency problem. This happens when a category is not present in the training set. It will give it a 0 probability.
- The results are biased with less amount of training data, and requires a significant amount of examples for each class.
- The assumption of independence of features do not hold with real-life data, as most of the features are usually co-related with each other.

2.6.4 Bayesian Belief Network

It consists of nodes representing states, and a pair of nodes are connected by directed edges with weights. The edge weight between two nodes, A to B, denotes the probability of switching states from A to B. As such, a Bayesian Belief Network (BBN) is a weighted directed graph, as shown in Fig. 2.5. BBN's are causal networks, as they model a cause-effect process. The stochastic binary units in belief networks have a state of 0 or 1, and the probability of becoming one is determined by a bias and weighted input from other units. The top layer nodes are called the parent nodes. In Fig. 2.5, the Earthquake, Burglary and Fire form the parent nodes.

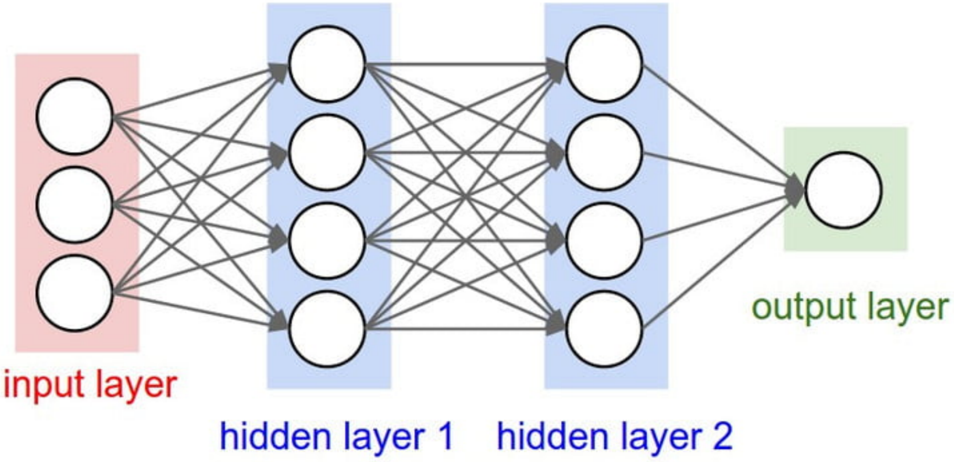


Figure 2.6: ANN architecture with two hidden layers

2.7 Artificial Neural Networks

An artificial neural network (ANN) is a network or interconnections of nodes called artificial neurons. Each connection transmits a signal to other neurons. An artificial neuron receives a signal and then processes it for further communication to connected neurons. The connections between any two neurons are called edges. The edges typically have a weight associated with them. The weight of an edge signifies the strength of the connection. Generally, neurons are aggregated into layers. The signals travel from the input layer to the output layer, often passing via intermediate layers called hidden layers. The ANN is vaguely inspired by biological neural networks.

2.7.1 Types of Artificial Neural Networks

Feedforward Neural Networks

In feedforward neural networks (FNN), information travels in only one direction from input to output.

Recurrent Neural Networks

Data flows in multiple directions in a recurrent neural network (RNN), unlike FNN. These neural networks possess better learning abilities and are widely employed for more complex tasks such as learning handwriting or language recognition. A Hopfield network is an example of an RNN.

Convolutional Neural Networks

In convolutional neural networks (CNN), convolution layers get slid over the whole data instance in the input layer.

2.8 Conclusion

In this chapter, several classification methods have been discussed. Each of them has certain advantages over the other, and hence it is important to know them to select which classifier best solves the problem at hand, given the data. It is important to note that the classification performance usually declines as the number of classes increases.

In many classification problems, it is found that regression is performed first, followed by some thresholding operations to classify. The next chapter deals with understanding regression and the type of regression problems.

Chapter 3

Regression

Regression attempts to determine the strength of the relationship between one dependent variable and a series of independent variables. In this case, the output label is continuous, instead of being categorical for classification tasks. The most common form of regression is the Linear regression.

3.1 Linear Regression

In Linear regression, the relationship between the dependent and independent variables is linear. Linear regression may be simple or multiple. In simple Linear regression, there is only a single independent variable x. The general form is

$$y = mx + c \tag{3.1}$$

For multiple linear regression with K dependent variables, the general form is

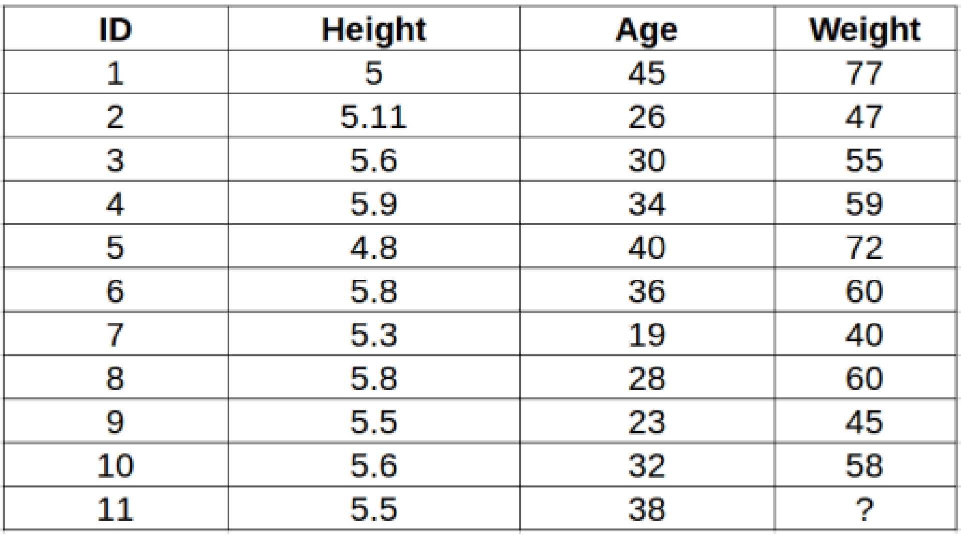
$$y = \sum_{i=1}^{K} m_i x_i + c \tag{3.2}$$

The basic assumptions of linear regression are

- linear relationship
- homoscedasticity
- independence
- multivariate normality

3.2 K Nearest Neighbors - Regression

Let us start with a simple example. Consider the following table – it consists of the height, age and weight (target) value for 10 people. As you can see, the weight value of ID11 is missing. We need to predict the weight of this person based on their height and age. The KNN Algorithm is presented as follows.



ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?

Figure 3.1: Sample table showing height, age, and weight of 11 peoeple with a missing data

- Load the data.
- Initialize K to your chosen number of neighbors.
- For each example in the data
  - Calculate the distance between the query example and the current example from the data.
  - Add the distance and the index of the example to an ordered collection.
- Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances.
- Pick the first K entries from the sorted collection.
- Get the labels of the selected K entries.
- Return the mode of the K labels

In the above graph, the y-axis represents the height of a person (in feet) and the x-axis represents the age (in years). The points are numbered according to the ID values. The yellow point (ID 11) is our test point. If I ask you to identify the weight of ID11 based on the plot, what would be your answer? You would likely say that since ID11 is closer to points 5 and 1, so it must have a weight similar to these IDs, probably between 72-77 kgs (weights of ID1 and ID5 from the table).

The algorithm uses 'feature similarity' to predict values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set. From our example, we know that ID11 has height and age similar to ID1 and ID5, so the weight would also approximately be the same.

- First, the distance between the new point and each training point is calculated.

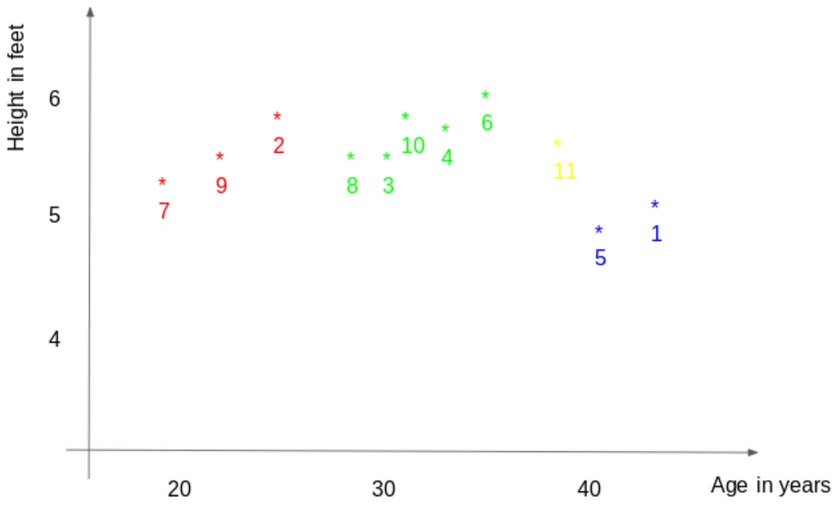


Figure 3.2: Scatter plot of age and height

- The closest k data points are selected (based on the distance). In this example, points 1, 5, 6 will be selected if value of k is 3.
- The average of these data points is the final prediction for the new point. Here, we have weight of ID11 = (77+72+60)/3 = 69.66 kg.

The first step is to calculate the distance between the new point and each training point. There are various methods for calculating this distance, of which the most commonly known methods are – Euclidean, Manhattan (for continuous) and Hamming distance (for categorical).

3.2.1 How to choose the k factor?

The second step is to select the k value. This determines the number of neighbors we look at when we assign a value to any new observation.

3.3 Regression Trees

3.3.1 Overfitting

Overfitting is a major problem in regression. The most popular way to solve the overfitting problem is Regularization.

3.3.2 Regularization

It is a process of introducing additional information in order to solve an ill-posed problem or to prevent over-fitting.

Chapter 4

Clustering

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. These data points are clustered by using the basic concept that the data point lies within the given constraint from the cluster center. Clustering is very much important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for a good clustering. It depends on the user, what is the criteria they may use which satisfy their need.
Clustering can be hard or soft. In hard-clustering algorithms, the membership vector is binary in nature because either an item belongs to a cluster or it doesn't. In hard clustering, each data point either belongs to a cluster completely or not. For soft clustering algorithms, we need to compute a fuzziness coefficient that controls the degree of fuzziness. The most important clustering methods include

- Centroid-based Clustering
- Density-based Clustering
- Distribution-based Clustering
- Hierarchical Clustering

4.1 Centroid-based Clustering

Centroid-based clustering organizes the data into non-hierarchical clusters. k-means is the most widely-used centroid-based clustering algorithm. Centroid-based algorithms are efficient but sensitive to initial conditions and outliers.

4.1.1 K-means clustering algorithm

K-means algorithm partition n observations into k clusters where each observation belongs to the cluster with the nearest mean serving as a prototype of the cluster.

4.2 Density-based Clustering

These methods consider the clusters as the dense region having some similarity and different from the lower dense region of the space. These methods have good accuracy and ability to merge two clusters. Examples include DBSCAN (Density-Based Spatial Clustering of Applications with Noise) , OPTICS (Ordering Points to Identify Clustering Structure) etc.

4.3 Distribution-based Clustering

This clustering approach assumes data is composed of distributions, such as Gaussian distributions. As distance from the distribution's center increases, the probability that a point belongs to the distribution decreases.

4.4 Hierarchical Clustering

Hierarchical clustering is preferred if the number of clusters are not known apriori, unlike the k-means clustering. The clusters formed in this method forms a tree type structure based on the hierarchy. New clusters are formed using the previously formed one. Examples are CURE (Clustering Using Representatives), BIRCH (Balanced Iterative Reducing Clustering and using Hierarchies) etc. This hierarchy of clusters is represented as a tree called the dendrogram. Such algorithms fall into two categories:

- agglomerative clustering (bottom-up)
- divisive clustering (top-down)

4.4.1 Agglomerative clustering

Agglomerative clustering treats each data point as a single cluster at the outset. It then successively merges pairs of clusters until the final clusters are obtained.

Chapter 5

Density Estimation

Probability density is the relationship between observations and their probability. The overall shape of the probability density is referred to as a probability distribution . The calculation of probabilities for specific outcomes of a random variable is performed by a probability density function (PDF). It is useful to know the pdf for a sample of data in order to know whether a given observation is unlikely, or so unlikely as to be considered an outlier or anomaly and whether it should be removed. It is also helpful in order to choose appropriate learning methods that require input data to have a specific probability distribution.

It is unlikely that the pdf for a random sample of data is known. As such, the probability density must be approximated using a process known as probability density estimation.

5.1 Probability Density

A random variable x has a probability distribution $p(x)$. The relationship between the outcomes of a random variable and its probability is referred to as the density. For a discrete random variable, the number of outcomes is finite. Hence, the probability is directly obtained from the density. For a continuous random variable, the number of outcomes is infinite. Hence, the absolute probability for a specific outcome is zero. The density, hence, provides a relative likelihood of the outcome. For obtaining the probability for a range of outcomes, the pdf is integrated in the specified range.

5.2 Density Estimation

The problem of density estimation is that given a random variable, we estimate the density of its probabilities. For example, given a random sample of a variable, we might want to know things like the shape of the probability distribution, the most likely value, the spread of values, and other properties. Knowing the probability distribution for a random variable can help to calculate moments of the distribution, like the mean and variance, but can also be useful for other more general considerations, like determining whether an observation is unlikely or very unlikely and might be an outlier or anomaly.

This problem is referred to as probability density estimation, or simply "density estimation," as we are using the observations in a random sample to estimate the general density of probabilities beyond just the sample of data we have available. There are a few steps in the process of density estimation for a random variable.

- The first step is to review the density of observations in the random sample with a simple histogram.
- From the histogram, we might be able to identify a common and well-understood probability distribution that can be used, such as a normal distribution.
- If not, we may have to fit a model to estimate the distribution.

5.2.1 Summarize Density With a Histogram

The first step in density estimation is to create a histogram of the observations in the random sample. A histogram is a plot that involves first grouping the observations into bins and counting the number of events that fall into each bin. The counts, or frequencies of observations, in each bin are then plotted as a bar graph with the bins on the x-axis and the frequency on the y-axis. The choice of the number of bins is important as it controls the coarseness of the distribution. This governs in turn, how well the density of the observations is plotted.

Reviewing a histogram of a data sample with a range of different numbers of bins will help to identify whether the density looks like a common probability distribution or not. In most cases, you will see a unimodal distribution, such as the familiar bell shape of the normal, the flat shape of the uniform, or the descending or ascending shape of an exponential or Pareto distribution.

5.2.2 Parametric Density Estimation

The shape of a histogram of most random samples will match a well-known probability distribution . The common distributions are common because they occur again and again in different and sometimes unexpected domains. Once identified, you can attempt to estimate the density of the random variable with a chosen probability distribution. This can be achieved by estimating the parameters of the distribution from a random sample of data.

For example, the normal distribution has two parameters: the mean and the standard deviation. Given these two parameters, we now know the probability distribution function. These parameters can be estimated from data by calculating the sample mean and sample standard deviation. We refer to this process as parametric density estimation. Once we have estimated the density, we can check if it is a good fit. This can be done in many ways, such as:

- Plotting the density function and comparing the shape to the histogram.
- Sampling the density function and comparing the generated sample to the real sample.
- Using a statistical test to confirm the data fits the distribution.

5.2.3 Non-parametric Density Estimation

In some cases, a data sample may not resemble a common probability distribution or cannot be easily made to fit the distribution. This is often the case when the data has two peaks (bimodal distribution) or many peaks (multimodal distribution). In this case, parametric density estimation is not feasible and alternative methods can be used that do not use a common distribution. Instead, an algorithm is used to approximate the probability distribution of the data without a pre-defined distribution, referred to as a nonparametric method. The distributions will still have parameters but are not directly controllable in the same way as simple probability distributions. For example, a nonparametric method might estimate the density using all observations in a random sample, in effect making all observations in the sample "parameters."
Perhaps the most common non-parametric approach for estimating the probability density function of a continuous random variable is called kernel smoothing, or kernel density estimation (KDE) . In this case, a kernel is a mathematical function that returns a probability for a given value of a random variable. The kernel effectively smooths or interpolates the probabilities across the range of outcomes for a random variable such that the sum of probabilities equals one. The kernel function weights the contribution of observations from a data sample based on their relationship or distance to a given query sample for which the probability is requested. A parameter, called the smoothing parameter or the bandwidth, controls the scope, or window of observations, from the data sample that contributes to estimating the probability for a given sample. As such, kernel density estimation is sometimes referred to as a Parzen-Rosenblatt window . The smoothing parameter or bandwidth controls the number of samples or window of samples used to estimate the probability for a new point. A large window may result in a coarse density with little details, whereas a small window may have too much detail and not be smooth or general enough to correctly cover new or unseen examples. The contribution of samples within the window can be shaped using different functions, sometimes referred to as basis functions. The Basis Function or kernel controls the contribution of samples in the dataset toward estimating the probability of a new point. As such, it may be useful to experiment with different window sizes and different contribution functions and evaluate the results against histograms of the data.

Chapter 6

Dimensionality Reduction

We are generating a tremendous amount of data daily. As data generation and collection keeps increasing, visualizing it and drawing inferences becomes more and more challenging. Suppose we have 2 variables, Age and Height. We can use a scatter or line plot between Age and Height and visualize their relationship easily. Now consider a case in which we have, say 100 variables (p=100). In this case, we can have 100(100-1)/2 = 5000 different plots. It does not make much sense to visualize each of them separately, right? In such cases where we have a large number of variables, it is better to select a subset of these variables ($p << 100$) which captures as much information as the original set of variables. Here are some of the benefits of applying dimensionality reduction to a dataset:

- Space required to store the data is reduced as the number of dimensions comes down.
- Less dimensions lead to less computation and training time.
- Some algorithms do not perform well when we have a large dimensions. So reducing these dimensions needs to happen for the algorithm to be useful.
- It helps in visualizing data.

6.1 Missing Values Ratio

Data columns with too many missing values are unlikely to carry much useful information. Thus data columns with number of missing values greater than a given threshold can be removed. The higher the threshold, the more aggressive the reduction.

6.2 Low Variance Filter

Similarly to the previous technique, data columns with little changes in the data carry little information. Thus all data columns with variance lower than a given threshold are removed. A word of caution: variance is range dependent; therefore normalization is required before applying this technique.

6.3 High Correlation Filter

Data columns with very similar trends are also likely to carry very similar information. In this case, only one of them will suffice to feed the machine learning model. Here we calculate the correlation coefficient between numerical columns and between nominal columns as the Pearson's Product Moment Coefficient and the Pearson's chi square value respectively. Pairs of columns with correlation coefficient higher than a threshold are reduced to only one. A word of caution: correlation is scale sensitive; therefore column normalization is required for a meaningful correlation comparison.

6.4 Backward Feature Elimination

In this technique, at a given iteration, the selected classification algorithm is trained on n input features. Then we remove one input feature at a time and train the same model on n-1 input features n times. The input feature whose removal has produced the smallest increase in the error rate is removed, leaving us with n-1 input features. The classification is then repeated using n-2 features, and so on. Each iteration k produces a model trained on n-k features and an error rate e(k). Selecting the maximum tolerable error rate, we define the smallest number of features necessary to reach that classification performance with the selected machine learning algorithm.

6.5 Forward Feature Selection

This is the inverse process to the Backward Feature Elimination. We start with 1 feature only, progressively adding 1 feature at a time, i.e. the feature that produces the highest increase in performance. Both algorithms, Backward Feature Elimination and Forward Feature Construction, are quite time and computationally expensive. They are practically only applicable to a data set with an already relatively low number of input columns.

6.6 Factor Analysis

Factor analysis extracts maximum common variance from all variables and puts them into a common score. This algorithm creates factors from the observed variables to represent the common variance i.e. variance due to correlation among the observed variables. Factor loading is the correlation coefficient for the variable and factor. Factor loading shows the variance explained by the variable on that particular factor. Eigenvalues is also called characteristic roots. Eigenvalues shows variance explained by that particular factor out of the total variance. According to the Kaiser Criterion, Eigenvalues form a good criterion for determining a factor. If Eigenvalues are greater than one, we should consider that a factor and if Eigenvalues is less than one, then we should not consider that a factor.

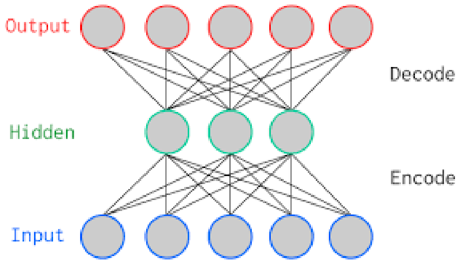


Figure 6.1: Single-layer vanilla autoencoder

6.7 t-Distributed Stochastic Neighbor Embedding (t-SNE)

The t-Distributed Stochastic Neighbor Embedding is a non-linear dimensionality reduction algorithm used for exploring high-dimensional data. It maps multi-dimensional data to two or more dimensions suitable for human observation.

6.8 Principal component analysis (PCA)

Principal component analysis (PCA) is a linear classification technique. PCA uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. The new components of the co-ordinate system are known as the principal components. The principal components are in the decreasing order of their variances of the components.

6.9 Independent component analysis (ICA)

6.10 Autoencoders

An autoencoder is an unsupervised machine learning algorithm that takes an image as input and tries to reconstruct it using fewer number of bits from the bottleneck also known as latent space. The image is majorly compressed at the bottleneck. The compression in autoencoders is achieved by training the network for a period of time and as it learns it tries to best represent the input image at the bottleneck. The general image compression algorithms like JPEG and JPEG lossless compression techniques compress the images without the need for any kind of training and do fairly well in compressing the images. The major difference between autoencoders and PCA lies in the transformation part, PCA uses linear transformation whereas autoencoders use non-linear transformations. As shown in the Fig. 6.1, the input and output layers have the same number of neurons. You feed an image with just five pixel values into the autoencoder which is compressed by the encoder into three pixel values at the bottleneck (middle layer) or latent space. Using these three values, the decoder tries to reconstruct the five pixel values or rather the input image which you fed as an input to the network. Autoencoder can be broken into three parts.

- Encoder: this part of the network compresses or downsamples the input into a fewer number of bits. The space represented by these fewer number of bits is often called the latent-space or bottleneck. The bottleneck is also called the ”maximum point of compression” since at this point the input is compressed the maximum. These compressed bits that represent the original input are together called an “encoding” of the input.

- Code: this is the part of the network that represents the compressed input fed to the decoder.

- Decoder: this part of the network tries to reconstruct the input using only the encoding of the input. When the decoder is able to reconstruct the input exactly as it was fed to the encoder, you can say that the encoder is able to produce the best encodings for the input with which the decoder is able to reconstruct well. There are variety of autoencoders, such as the convolutional autoencoder, denoising autoencoder, variational autoencoder and sparse autoencoder.

6.10.1 Properties

Some of the important properties of autoencoders are:

- They do not need any labels to be trained on.

- They are data-specific. They can compress only data similar to which they are trained with.

- The compression is lossy, as the ouput would not be an exact replica of the input.

6.10.2 Training

There are certain hyper-parameters to be set during training. They are:

- Code size: This is the number of nodes in the middle layer, known as the bottleneck. Smaller size results in more compression.

- Number of layers: This is the same for both the encoder and decoder sections. This has to be set considering a trade-off of a good compression, and overfitting.

- Loss function: Generally, the mean-squared error and binary cross entropy are used.

- Number of nodes per layer: Stacked autoencoder looks like a sandwich.

Chapter 7

Ensemble Learning

Ensemble learning is the process by which multiple models are strategically generated and combined. Ensemble learning is primarily used to improve the performance of a machine learning model. Such improvement may be concerning lower error on regression or high accuracy for classification. Ensemble methods usually produce more accurate solutions than a single model would.

7.1 Bagging

Suppose we have M models. In bagging, we create M sets from the training data, by sampling with replacement. Then we train the M unique models, each with one of the M dataset. For a test set, we combine the results of individual models to arrive at a prediction. One such combination method can be voting, in case of classification, and averaging, in case of regression. As such, bagging is a way to decrease the variance in the prediction.
The sampling with replacement is often known as bootstrapping, while the combining of decisions is called aggregation. Hence, bagging is also known as bootstrap aggregating.

7.2 Boosting

Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers. Weak classifiers are models that achieve accuracy just above random chance on a classification problem. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added. Hence, boosting is a sequential model and does not involve bootstrap sampling.

7.2.1 AdaBoost

AdaBoost was the first really successful boosting algorithm developed for binary classification. The most suited and therefore most common algorithm used with

AdaBoost are decision trees with one level. Because these trees are so short and only contain one decision for classification, they are often called decision stumps. Each instance in the training dataset is weighted. The AdaBoost algorithm is as follow:

1. The initial observation weights are set to:

$$w_i(0) = \frac{1}{N} \tag{7.1}$$

   Here N is the number of training instances. Suppose we have M classifiers. Then for $m = 1, 2, \ldots, M$

   (a) Fit a classifier $G_m(x)$ to the training data.

   (b) Compute the misclassification error e_m as

$$e_m = \frac{\sum_{i=1}^{N} w_i I(y_i - G_m(x_i))}{\sum_{i=1}^{N} w_i} \tag{7.2}$$

   (c) Compute α_m as

$$\alpha_m = \log \frac{1 - e_m}{e_m} \tag{7.3}$$

   (d) Update weights as

$$w_i(j) = w_i(j)[1 - e^{\alpha_m I(y_i - G_m(x_i))}] \tag{7.4}$$

2. The rate is calculated for the trained model. Traditionally, this is calculated, with N_c as the number of correct classifications, as:

A weak classifier (decision stump) is prepared on the training data using the weighted samples. Only binary (two-class) classification problems are supported, so each decision stump makes one decision on one input variable and outputs a +1.0 or -1.0 value for the first or second class value.

7.2.2 XGBoost

XGBoost (eXtreme Gradient Boosting) is a popular supervised-learning algorithm used for regression and classification on large datasets. It uses sequentially-built shallow decision trees to provide accurate results and a highly-scalable training method that avoids overfitting. Compared to random forests and XGBoost, AdaBoost performs worse when irrelevant features are included in the model.

7.3 Comparison - Bagging vs Boosting

If the problem is that the single model gets a very low performance, Bagging will rarely get a better bias. However, Boosting could generate a combined model with lower errors as it optimises the advantages and reduces pitfalls of the single model.

Chapter 8

Reinforcement Learning

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize some notion of cumulative reward. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of training dataset, it is bound to learn from its experience. As an example, consider Fig. 8.1, where we have an agent and a reward, with many hurdles in between. The agent is supposed to find the best possible path to reach the reward. The figure shows robot, diamond and fire. The goal of the robot is to get the reward that is the diamond and avoid the hurdles that is fire. The robot learns by trying all the possible paths and then choosing the path which gives him the reward with the least hurdles. Each right step will give the robot a reward and each wrong step will subtract the reward of the robot. The total reward will be calculated when it reaches the final reward that is the diamond.

8.0.1 Main points in Reinforcement learning

The input should be an initial state from which the model will start. There are many possible outputs as there are variety of solution to a particular problem. The training is based upon the input. The model will return a state and the user will decide to reward or punish the model based on its output. The model continues to learn. The best solution is decided based on the maximum reward.

8.0.2 Types of Reinforcement

There are two types of Reinforcement: positive and negative.

Positive Reinforcement

Positive Reinforcement is defined as when an event, occurs due to a particular behavior, increases the strength and the frequency of the behavior. In other words

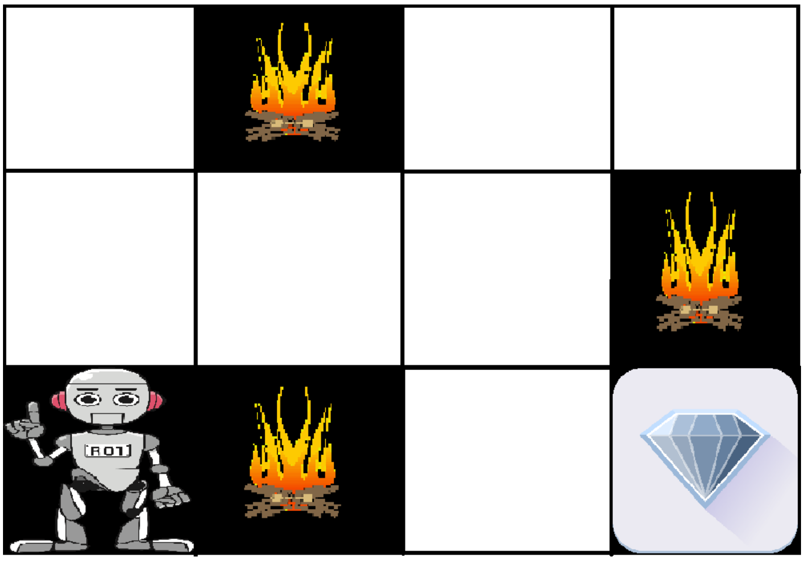


Figure 8.1: Example of reinforcement learning with a robot, diamond and fire

it has a positive effect on the behavior. Advantages of positive reinforcement learning are:

- Maximizes Performance
- Sustain Change for a long period of time

Disadvantages of positive reinforcement learning is that too much reinforcement can lead to overload of states which can diminish the results.

Negative Reinforcement

Negative Reinforcement is defined as strengthening of a behavior because a negative condition is stopped or avoided. Advantages of negative reinforcement learning:

- Increases Behavior
- Provide defiance to minimum standard of performance

Disadvantages of negative reinforcement learning is that it Only provides enough to meet up the minimum behavior.

8.0.3 Terminology

Agent- The agent is an entity that we are going to train. For example, a robot that is going to be trained to assist in cooking is an agent. Environment- The surroundings around the agent are called Environment. The kitchen is an environment in the case of the above-mentioned robot. State (S)- The current situation of the agent is called the state. So, in the case of the robot, the position where the robot is, the temperature of the robot, its posture, etc. collectively define the state of the robot. Action (A)- The robot can move left or right, or it can pass an onion to the chef, these are some of the actions that the agent (robot) can take. Policy (π)- The policy is the reasoning behind taking an action. Reward (R) -A reward is received by the agent for taking a desirable action. Value (V)- The value is the potential future reward that the agent can receive.

Agent Types

Simple Reflex Agents: Simple reflex agents ignore the history of the environment and its interaction with the environment and act entirely on the current situation. Model-Based Reflex Agents: These models perceive the world according to the predefined models. This model also keeps track of internal conditions, which can be adjusted according to the changes made in the environment. Goal-Based Agents: These kinds of agents react according to the goals given to them. Their ultimate aim is to reach that goal. If the agent is provided with multiple-choice, it will select the choice that will make it closer to the goal. Utility-based Agents: Sometimes, reaching the desired goal is not enough. You have to make the safest, easiest, and cheapest trip to the goal. Utility-based agents chose actions based on utilities (preferences set) of the choices. Learning Agents: These kinds of agents can learn from their experiences.

8.0.4 Practical Applications

RL can be used in robotics for industrial automation, create training systems that provide custom instruction and materials according to the requirement of students.

Markov's decision process

Markov's decision process (MDP) is a mathematical approach for reinforcement learning. Markov's decision process (MDP) is a mathematical framework used to solve problems where outcomes are partially random and partly controlled. To solve a complex problem using Markov's decision process, the following basic things are needed.

8.0.5 Q Learning

Q-learning is an off policy reinforcement learning algorithm that seeks to find the best action to take given the current state. It is considered off-policy because the q-learning function learns from actions that are outside the current policy, like taking random actions, and therefore a policy is not needed. More specifically, q-learning seeks to learn a policy that maximizes the total reward. Q Learning is a model-free learning policy that chooses the best course of action in an environment, depending on where in the environment the agent is (an agent is an entity that makes a decision and enables AI to be put into action). Model-free learning policy means that the nature and predictions of the environment to learn and move forward. It does not reward a system to learn, it uses the trial and error method instead.

Chapter 9

Deep Learning

Deep Learning uses multiple layers to progressively extract higher level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.
Traditionally, a shallow neural network (SNN) is one with one or two hidden layers. Thus, a deep neural network (DNN) is one with more than two hidden layers. This is the most accepted definition. The model is updated layer-by-layer backward from the output to the input using an estimate of error that assumes the weights in the layers prior to the current layer are fixed.

9.1 Challenges

Training deep neural networks with many layers is challenging as they can be sensitive to the initial random weights and configuration of the learning algorithm.
Less number of training samples may result in either underfitting or overfitting of the model.

Table 9.1: Choice of neural network

Task	**Network**
Text Recognition	RNTN, RNN
Image Recognition	CNN, DBN
Object Recognition	CNN, RNTN
Time Series Analysis	RNN
Speech Recognition	RNN

9.2 Optimizers

An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate. Thus, it helps in reducing the overall loss and improve the accuracy.

9.2.1 Gradient Descent

Gradient Descent is an iterative algorithm used in loss function to find the global minima. The loss can be any differentiable function.
Gradient Descent uses the whole training data to update weight and bias. Suppose if we have millions of records then training becomes slow and computationally very expensive.

9.2.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) solves the Gradient Descent problem by using only single records to update parameters. But, still, SGD is slow to converge because it needs forward and backward propagation for every record, and the path to reach global minima becomes very noisy.

9.2.3 Mini-batch Gradient Descent (SGD)

Mini-batch GD overcomes the SDG drawbacks by using a batch of records to update the parameter. Since it doesn't use entire records to update parameter, the path to reach global minima is not as smooth as Gradient Descent.
Exponentially Weighted Averages is used in sequential noisy data to reduce the noise and smoothen the data.

9.2.4 SGD with momentum

In SGD with momentum, we have added momentum in a gradient function.

9.2.5 Adagrad (Adaptive Gradient Algorithm)

The idea behind Adagrad is to use different learning rates for each parameter based on iteration. The reason behind the need for different learning rates is that the learning rate for sparse features parameters needs to be higher compare to the dense features parameter because the frequency of occurrence of sparse features is lower. Lastly, despite not having to manually tune the learning rate there is one huge disadvantage i.e due to monotonically decreasing learning rates, at some point in time step, the model will stop learning as the learning rate is almost close to 0.

9.2.6 Adadelta

Adadelta is an extension of Adagrad that attempts to solve its radically diminishing learning rates.

9.2.7 Adam

Adam optimizer is by far one of the most preferred optimizers. The idea behind Adam optimizer is to utilize the momentum concept from “SGD with momentum” and adaptive learning rate from “Ada delta”.

9.2.8 Batch normalization

Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks. It does this scaling the output of the layer, specifically by standardizing the activations of each input variable per mini-batch, such as the activations of a node from the previous layer. Batch normalization can be implemented during training by calculating the mean and standard deviation of each input variable to a layer per mini-batch and using these statistics to perform the standardization.

9.2.9 Data Augmentation

Data Augmentation is a technique for taking an image and using it to generating new ones.

9.3 Deep Neural Networks (DNN)

9.3.1 Convolutional Neural Networks (CNN)

Activation Functions

ReLU ReLU activation function has a gradient of 0 for all the values of inputs that are less than zero, which would deactivate the neurons in that region and may cause dying ReLU problem.

Leaky ReLU Leaky Rectified Linear Unit, or Leaky ReLU, is a type of activation function based on a ReLU, but it has a small slope for negative values instead of a flat slope. The slope coefficient is determined before training, i.e. it is not learnt during training. This type of activation function is popular in tasks where we may suffer from sparse gradients, for example training generative adversarial networks.

Parametric ReLU (PReLU) Parametric ReLU (PReLU) is a type of leaky ReLU that, instead of having a predetermined slope like 0.01, makes it a parameter for the neural network to figure out itself.

9.3.2 Recurrent Neural Networks (RNN)

Basic feedforward networks remember things too, but they remember the things they learnt during training. While RNNs learn similarly while training, in addition, they remember things learnt from prior input(s) while generating output(s).

Image illustrating long-term dependencies. source: colah's (CC0). It is used in different types of models-

1.) Vector-Sequence Models- They take fixed-sized vectors as input and output vectors of any length, for example, in image captioning, the image is given as an input and the output describes the image.

2.) Sequence-Vector Model- Take a vector of any size and output a vector of fixed size. Eg. Sentiment analysis of a movie rates the review of any movie as positive or negative as a fixed size vector.

3.) Sequence-to-Sequence Model- The most popular and most used variant, take input as a sequence and give output as another sequence with variant sizes. Eg. Language translation, for time series data for stock market prediction.

Its disadvantages-

Slow to train. Long sequence leads to vanishing gradient or, say, the problem of long term dependencies. In simple terms, its memory is not that strong when it comes to remembering the old connection. For Eg. The clouds are in the .

It is obvious that the next word will be sky, as it is linked with the clouds. Here we see the distance between clouds and the predicted word is less so RNN can predict it easily.

But, for another example,

I grew up in Germany with my parents, I spent many years and have proper knowledge about their culture, thats why I speak fluent.

Here the predicted word is German, which is directly connected with Germany, but the distance between Germany and the predicted word is more in this case, so it is difficult for RNN to predict.

So, Unfortunately, as that gap grows, RNNs become unable to connect, as their memory fades with distance.

9.3.3 Recursive Neural Tensor Network (RNTN)

Recursive neural tensor networks (RNTNs) are neural nets useful for natural-language processing. They have a tree structure with a neural net at each node. Word vectors are used as features and serve as the basis of sequential classification. They are then grouped into subphrases, and the subphrases are combined into a sentence that can be classified by sentiment and other metrics.

Long Short Term Memory (LSTM)

LSTM networks take time to train. Words are passed sequentially.

9.3.4 Deep Belief Network (DBN)

A deep belief network (DBN) is a generative graphical model. It is composed of multiple layers of latent variables, with connections between the layers but not between units within each layer.

9.4 Supervised Learning

9.4.1 Object detection

Object detection is a challenging computer vision task that involves predicting the following aspects of the object.

- presence
- location
- type

The Mask Region-based Convolutional Neural Network (R-CNN) model is one of the state-of-the-art approaches for object recognition tasks. The Mask R-CNN is the most recent variation of the family of models and supports both object detection and object segmentation. Object segmentation not only involves localizing objects in the image but also specifies a mask for the image, indicating exactly which pixels in the image belong to the object.

9.5 Unsupervised

The popular types of Unsupervised learning problems use Restricted Boltzman Machines or Autoencoders.

9.5.1 Restricted Boltzman Machines

A restricted Boltzmann machine (RBM) is a generative stochastic artificial neural network that can learn a probability distribution over its set of inputs. It is useful for dimensionality reduction, classification, regression, collaborative filtering, feature learning and topic modelling.

9.5.2 Deep Belief Network

Deep Belief Network (DBN) is a class of deep neural network which comprises of multiple layers of graphical model having both directed and undirected edges. It is composed of multiple layers of hidden units, where each layers are connected with each others but units are not.

9.5.3 Deep Autoencoders

Convolutional Autoencoders

Since your input data consists of images, it is a good idea to use a convolutional autoencoder. It is not an autoencoder variant, but rather a traditional autoencoder stacked with convolution layers. Convolution layers along with max-pooling layers, convert the input from wide (a 28 x 28 image) and thin (a single channel or gray scale) to small (7 x 7 image at the latent space) and thick (128 channels). Denoising overcomplete AEs: recreate images without the random noises originally injected. Log loss is usually the loss function. Undercomplete AEs for anomaly detection: use AEs for credit card fraud detection via anomaly detection. Among these deep generative models, two major families stand out and deserve a special attention: Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). Variational AEs for creating synthetic faces: with a convolutional VAEs, we can make fake faces. A VAE is an autoencoder whose encodings distribution is regularized during the training in order to ensure that its latent space has good properties allowing us to generate some new data. Latent loss (K-L divergence) is usually the loss function.

9.6 Transfer Learning

Transfer learning is an approach in DL, where knowledge is transferred from one model to another. The early and central layers are employed in transfer learning, and the latter layers are only retrained. The training and testing data need not come from the same source or be with the same distribution. Factors that can affect transfer include:

- Context and degree of original learning: how well the learner acquired the knowledge.
- Similarity: commonalities between original learning and new, such as environment and other memory cues.
- Critical attributes: characteristics that make something unique.

9.6.1 Types

There are three types of transfer of learning:

- Positive transfer: When learning in one situation facilitates learning in another situation, it is known as a positive transfer.
- Negative transfer: When learning of one task makes the learning of another task harder- it is known as a negative transfer.
- Neutral transfer: When learning of one activity neither facilitates or hinders the learning of another task, it is a case of neutral transfer. It is also called as zero transfer.

Index