CHAPTER 1

Machine Learning

Nature is a self-made machine, more perfectly automated than any automated machine. To create something in the image of nature is to create a machine, and it was by learning the inner working of nature that man became a builder of machines.

—Eric Hoffer, *Reflections on the Human Condition*

Machine learning (ML) is a branch of artificial intelligence that systematically applies algorithms to synthesize the underlying relationships among data and information. For example, ML systems can be trained on automatic speech recognition systems (such as iPhone's Siri) to convert acoustic information in a sequence of speech data into semantic structure expressed in the form of a string of words.

ML is already finding widespread uses in web search, ad placement, credit scoring, stock market prediction, gene sequence analysis, behavior analysis, smart coupons, drug development, weather forecasting, big data analytics, and many more applications. ML will play a decisive role in the development of a host of user-centric innovations.

ML owes its burgeoning adoption to its ability to characterize underlying relationships within large arrays of data in ways that solve problems in big data analytics, behavioral pattern recognition, and information evolution. ML systems can moreover be trained to categorize the changing conditions of a process so as to model variations in operating behavior. As bodies of knowledge evolve under the influence of new ideas and technologies, ML systems can identify disruptions to the existing models and redesign and retrain themselves to adapt to and coevolve with the new knowledge.

The computational characteristic of ML is to generalize the *training experience* (or examples) and output a hypothesis that estimates the target function. The generalization attribute of ML allows the system to perform well on unseen data instances by accurately predicting the future data. Unlike other optimization problems, ML does not have a well-defined function that can be optimized. Instead, training errors serve as a catalyst to test learning errors. The process of generalization requires classifiers that input discrete or continuous feature vectors and output a class.

The goal of ML is to predict future events or scenarios that are unknown to the computer. In 1959, Arthur Samuel described ML as the "field of study that gives computers the ability to learn without being explicitly programmed" (Samuel 1959). He concluded that programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort. According to Tom M. Mitchell's definition of ML: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." Alan Turing's seminal paper (Turing 1950) introduced a benchmark standard for demonstrating machine intelligence, such that a machine has to be intelligent and responsive in a manner that cannot be differentiated from that of a human being.

The learning process plays a crucial role in generalizing the problem by acting on its historical experience. Experience exists in the form of training datasets, which aid in achieving accurate results on new and unseen tasks. The training datasets encompass an existing problem domain that the learner uses to build a general model about that domain. This enables the model to generate largely accurate predictions in new cases.

Key Terminology

To facilitate the reader's understanding of the concept of ML, this section defines and discusses some key multidisciplinary conceptual terms in relation to ML.

- ***classifier***. A method that receives a new input as an unlabeled instance of an observation or feature and identifies a category or class to which it belongs. Many commonly used classifiers employ statistical inference (probability measure) to categorize the best label for a given instance.
- ***confusion matrix*** (aka *error matrix*). A matrix that visualizes the performance of the classification algorithm using the data in the matrix. It compares the predicted classification against the actual classification in the form of false positive, true positive, false negative and true negative information. A confusion matrix for a two-class classifier system (Kohavi and Provost, 1998) follows:

Confusion Matrix		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

- TP = Outcome is correctly identified as positive.
- TN = Outcome is correctly identified as negative.
- FP = Outcome is incorrectly identified as positive.
- FN = Outcome is incorrectly identified as negative

- ***accuracy*** (aka *error rate*). The rate of correct (or incorrect) predictions made by the model over a dataset. Accuracy is usually estimated by using an independent test set that was not used at any time during the learning process. More complex accuracy estimation techniques, such as *cross-validation* and *bootstrapping*, are commonly used, especially with datasets containing a small number of instances.

$$\text{Accuracy (AC)} = \frac{TP + TN}{TP + TN + FN + FP} \tag{1-1}$$

$$\text{Precision (P)} = \frac{TP}{TP + FP} \tag{1-2}$$

$$\text{Recall (R, true positive rate)} = \frac{TP}{TP + FN} \tag{1-3}$$

$$\text{F} - \text{Measure} = \frac{\left(\beta^2 + 1\right) \cdot P \cdot R}{\beta^2 \cdot P + R}, \tag{1-4}$$

where β has a value from 0 to infinity (∞) and is used to control the weight assigned to P and R.

- ***cost***. The measurement of performance (or accuracy) of a model that predicts (or evaluates) the outcome for an established result; in other words, that quantifies the deviation between predicted and actual values (or class labels). An optimization function attempts to minimize the cost function.
- ***cross-validation***. A verification technique that evaluates the generalization ability of a model for an independent dataset. It defines a dataset that is used for testing the trained model during the training phase for overfitting. Cross-validation can also be used to evaluate the performance of various prediction functions. In k-*fold cross-validation,* the training dataset is arbitrarily partitioned into *k* mutually exclusive subsamples (or *folds*) of equal sizes. The model is trained *k* times (or *folds*), where each iteration uses one of the *k* subsamples for testing (cross-validating), and the remaining *k*-1 subsamples are applied toward training the model. The *k* results of cross-validation are averaged to estimate the accuracy as a single estimation.
- ***data mining***. The process of knowledge discovery (q.v.) or pattern detection in a large dataset. The methods involved in data mining aid in extracting the accurate data and transforming it to a known structure for further evaluation.
- ***dataset***. A collection of data that conform to a schema with no ordering requirements. In a typical dataset, each column represents a feature and each row represents a member of the dataset.
- ***dimension***. A set of attributes that defines a property. The primary functions of dimension are filtering, classification, and grouping.
- ***induction algorithm***. An algorithm that uses the training dataset to generate a model that generalizes beyond the training dataset.
- ***instance***. An object characterized by feature vectors from which the model is either trained for generalization or used for prediction.
- ***knowledge discovery***. The process of abstracting knowledge from structured or unstructured sources to serve as the basis for further exploration. Such knowledge is collectively represented as a schema and can be condensed in the form of a model or models to which queries can be made for statistical prediction, evaluation, and further knowledge discovery.

- ***model***. A structure that summarizes a dataset for description or prediction. Each model can be tuned to the specific requirements of an application. Applications in big data have large datasets with many predictors and features that are too complex for a simple parametric model to extract useful information. The learning process synthesizes the parameters and the structures of a model from a given dataset. Models may be generally categorized as either *parametric* (described by a finite set of parameters, such that future predictions are independent of the new dataset) or *nonparametric* (described by an infinite set of parameters, such that the data distribution cannot be expressed in terms of a finite set of parameters). Nonparametric models are simple and flexible, and make fewer assumptions, but they require larger datasets to derive accurate conclusions.
- ***online analytical processing*** (**OLAP**). An approach for resolving multidimensional analytical queries. Such queries index into the data with two or more attributes (or dimensions). OLAP encompasses a broad class of business intelligence data and is usually synonymous with *multidimensional OLAP* (MOLAP). OLAP engines facilitate the exploration of multidimensional data interactively from several perspectives, thereby allowing for complex analytical and ad hoc queries with a rapid execution time. OLAP commonly uses intermediate data structures to store precalculated results on multidimensional data, allowing fast computation. *Relational OLAP* (ROLAP) uses relational databases of the base data and the dimension tables.
- ***schema***. A high-level specification of a dataset's attributes and properties.
- ***supervised learning***. Learning techniques that extract associations between independent attributes and a designated dependent attribute (the label). Supervised learning uses a training dataset to develop a prediction model by consuming input data and output values. The model can then make predictions of the output values for a new dataset. The performance of models developed using supervised learning depends upon the size and variance of the training dataset to achieve better generalization and greater predictive power for new datasets. Most induction algorithms fall into the supervised learning category.
- ***unsupervised learning***. Learning techniques that group instances without a prespecified dependent attribute. This technique generally involves learning structured patterns in the data by rejecting pure unstructured noise. Clustering and dimensionality reduction algorithms are usually unsupervised.
- ***feature vector***. An *n*-dimensional numerical vector of explanatory variables representing an instance of some object that facilitates processing and statistical analysis. Feature vectors are often weighted to construct a predictor function that is used to evaluate the quality or fitness of the prediction. The dimensionality of a feature vector can be reduced by various dimensionality reduction techniques, such as *principal component analysis* (PCA), *multilinear subspace reduction*, *isomaps*, and *latent semantic analysis* (LSA). The vector space associated with these vectors is often called the *feature space*.

Developing a Learning Machine

Machine learning aids in the development of programs that improve their performance for a given task through experience and training. Many big data applications leverage ML to operate at highest efficiency. The sheer volume, diversity, and speed of data flow have made it impracticable to exploit the natural capability of human beings to analyze data in real time. The surge in social networking and the wide use of Internet-based applications have resulted not only in greater volume of data, but also increased complexity of data. To preserve data resolution and avoid data loss, these streams of data need to be analyzed in real time.

The heterogeneity of the big data stream and the massive computing power we possess today present us with abundant opportunities to foster learning methodologies that can identify best practices for a given business problem. The sophistication of modern computing machines can handle large data volumes, greater complexity, and terabytes of storage. Additionally, intelligent program-flows that run on these machines can process and combine many such complex data streams to develop predictive models and extract intrinsic patterns in otherwise noisy data. When you need to predict or forecast a target value, supervised learning is the appropriate choice. The next step is to decide, depending on the target value, between *clustering* (in the case of discrete target value) and *regression* (in the case of numerical target value).

You start the development of ML by identifying all the metrics that are critical to a decision process. The processes of ML synthesize models for optimizing the metrics. Because the metrics are essential to developing the solution for a given decision process, they must be selected carefully during conceptual stages.

It is also important to judge whether ML is the suitable approach for solving a given problem. By its nature, ML cannot deliver perfect accuracy. For solutions requiring highly accurate results in a bounded time period, ML may not be the preferred approach. In general, the following conditions are favorable to the application of ML: (a) very high accuracy is not desired; (b) large volumes of data contain undiscovered patterns or information to be synthesized; (c) the problem itself is not very well understood owing to lack of knowledge or historical information as a basis for developing suitable algorithms; and (d) the problem needs to adapt to changing environmental conditions.

The process of developing ML algorithms may be decomposed into the following steps:

1. *Collect the data*. Select the subset of all available data attributes that might be useful in solving the problem. Selecting all the available data may be unnecessary or counterproductive. Depending upon the problem, data can either be retrieved through a data-stream API (such as a CPU performance counters) or synthesized by combining multiple data streams. In some cases, the input data streams, whether raw or synthetic, may be statistically preprocessed to improve usage or reduce bandwidth.

2. *Preprocess the Data*. Present the data in a manner that is understood by the consumer of the data. Preprocessing consists of the following three steps:

    i. *Formatting*. The data needs to be presented in a useable format. Using an industry-standard format enable plugging the solution with multiple vendors that in turn can mix and match algorithms and data sources such as XML, HTML, and SOAP.

    ii. *Cleaning*. The data needs to be cleaned by removing, substituting, or fixing corrupt or missing data. In some cases, data needs to be normalized, discretized, averaged, smoothened, or differentiated for efficient usage. In other cases, data may need to be transmitted as integers, double precisions, or strings.

    iii. *Sampling*. Data need to be sampled at regular or adaptive intervals in a manner such that redundancy is minimized without the loss of information for transmission via communication channels.

3. *Transform the data.* Transform the data specific to the algorithm and the knowledge of the problem. Transformation can be in the form of feature scaling, decomposition, or aggregation. Features can be decomposed to extract the useful components embedded in the data or aggregated to combine multiple instances into a single feature.

4. *Train the algorithm.* Select the training and testing datasets from the transformed data. An algorithm is trained on the training dataset and evaluated against the test set. The transformed training dataset is fed to the algorithm for extraction of knowledge or information. This trained knowledge or information is stored as a model to be used for cross-validation and actual usage. Unsupervised learning, having no target value, does not require the training step.

5. *Test the algorithm.* Evaluate the algorithm to test its effectiveness and performance. This step enables quick determination whether any learnable structures can be identified in the data. A trained model exposed to test dataset is measured against predictions made on that test dataset which are indicative of the performance of the model. If the performance of the model needs improvement, repeat the previous steps by changing the data streams, sampling rates, transformations, linearizing models, outliers' removal methodology, and biasing schemes.

6. *Apply reinforcement learning.* Most control theoretic applications require a good feedback mechanism for stable operations. In many cases, the feedback data are sparse, delayed, or unspecific. In such cases, supervised learning may not be practical and may be substituted with *reinforcement learning* (RL). In contrast to supervised learning, RL employs dynamic performance rebalancing to learn from the consequences of interactions with the environment, without explicit training.

7. *Execute.* Apply the validated model to perform an actual task of prediction. If new data are encountered, the model is retrained by applying the previous steps. The process of training may coexist with the real task of predicting future behavior.

Machine Learning Algorithms

Based on underlying mappings between input data and anticipated output presented during the learning phase of ML, ML algorithms may be classified into the following six categories:

- *Supervised learning* is a learning mechanism that infers the underlying relationship between the observed data (also called *input data*) and a target variable (a dependent variable or label) that is subject to prediction (Figure 1-1). The learning task uses the labeled training data (training examples) to synthesize the model function that attempts to generalize the underlying relationship between the feature vectors (input) and the supervisory signals (output). The feature vectors influence the direction and magnitude of change in order to improve the overall performance of the function model. The training data comprise observed input (feature) vectors and a desired output value (also called the *supervisory signal* or *class label*). A well-trained function model based on a supervised learning algorithm can accurately predict the class labels for hidden phenomena embedded in unfamiliar or unobserved data instances. The goal of learning algorithms is to minimize the error for a given set of inputs (the training set). However, for a poor-quality training set that is influenced by the accuracy and versatility of the labeled examples, the model may encounter the problem of overfitting, which typically represents poor generalization and erroneous classification.

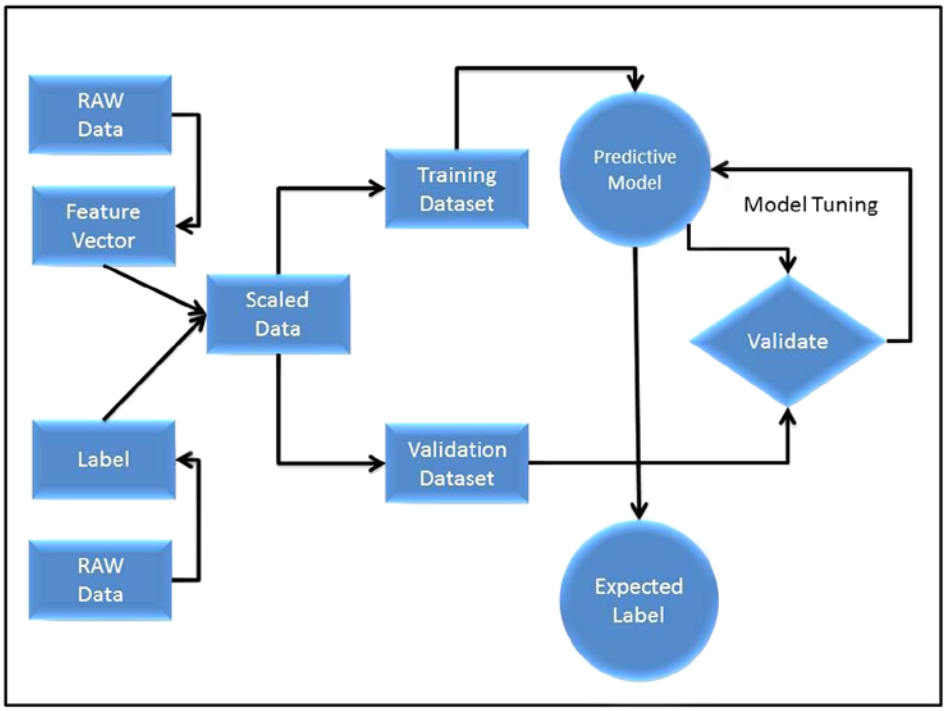

Figure 1-1. *High-level flow of supervised learning*

- ***Unsupervised learning algorithms*** are designed to discover hidden structures in unlabeled datasets, in which the desired output is unknown. This mechanism has found many uses in the areas of data compression, outlier detection, classification, human learning, and so on. The general approach to learning involves training through probabilistic data models. Two popular examples of unsupervised learning are *clustering* and *dimensionality reduction*. In general, an unsupervised learning dataset is composed of inputs $x_1, x_2, x_3 \cdots x_n$, but it contains neither target outputs (as in supervised learning) nor rewards from its environment. The goal of ML in this case is to hypothesize representations of the input data for efficient decision making, forecasting, and information filtering and clustering. For example, unsupervised training can aid in the development of phase-based models in which each phase, synthesized through an unsupervised learning process, represents a unique condition for opportunistic tuning of the process. Furthermore, each phase can act as a state and can be subjected to forecasting for proactive resource allocation or distribution. Unsupervised learning algorithms centered on a probabilistic distribution model generally use *maximum likelihood estimation* (MLE), *maximum a posteriori* (MAP), or Bayes methods. Other algorithms that are not based on probability distribution models may employ statistical measurements, quantization error, variance preserving, entropy gaps, and so on.

- ***Semi-supervised learning*** uses a combination of a small number of labeled and a large number of unlabeled datasets to generate a model function or classifier. Because the labeling process of acquired data requires intensive skilled human labor inputs, it is expensive and impracticable. In contrast, unlabeled data are relatively inexpensive and readily available. Semi-supervised ML methodology operates somewhere between the guidelines of unsupervised learning (unlabeled training data) and supervised learning (labeled training data) and can produce considerable improvement in learning accuracy. Semi-supervised learning has recently gained greater prominence, owing to the availability of large quantities of unlabeled data for diverse applications to web data, messaging data, stock data, retail data, biological data, images, and so on. This learning methodology can deliver value of practical and theoretical significance, especially in areas related to human learning, such as speech, vision, and handwriting, which involve a small amount of direct instruction and a large amount of unlabeled experience.
- ***Reinforcement learning (RL) methodology*** involves exploration of an adaptive sequence of actions or behaviors by an intelligent agent (*RL-agent*) in a given environment with a motivation to maximize the cumulative reward (Figure 1-2). The intelligent agent's action triggers an observable change in the state of the environment. The learning technique synthesizes an adaptation model by training itself for a given set of experimental actions and observed responses to the state of the environment. In general, this methodology can be viewed as a control-theoretic trial-and-error learning paradigm with rewards and punishments associated with a sequence of actions. The RL-agent changes its policy based on the collective experience and consequent rewards. RL seeks past actions it explored that resulted in rewards. To build an exhaustive database or model of all the possible action-reward projections, many unproven actions need to be tried. These untested actions may have to be attempted multiple times before ascertaining their strength. Therefore, you have to strike a balance between exploration of new possible actions and likelihood of failure resulting from those actions. Critical elements of RL include the following:
  - The *policy* is a key component of an RL-agent that maps the control-actions to the perceived state of the environment.
  - The *critic* represents an estimated value function that criticizes the actions that are made according to existing policy. Alternatively, the critic evaluates the performance of the current state in response to an action taken according to current policy. The critic-agent shapes the policy by making continuous and ongoing corrections.
  - The *reward function* estimates the instantaneous desirability of the perceived state of the environment for an attempted control-action.
  - *Models* are planning tools that aid in predicting the future course of action by contemplating possible future situations.

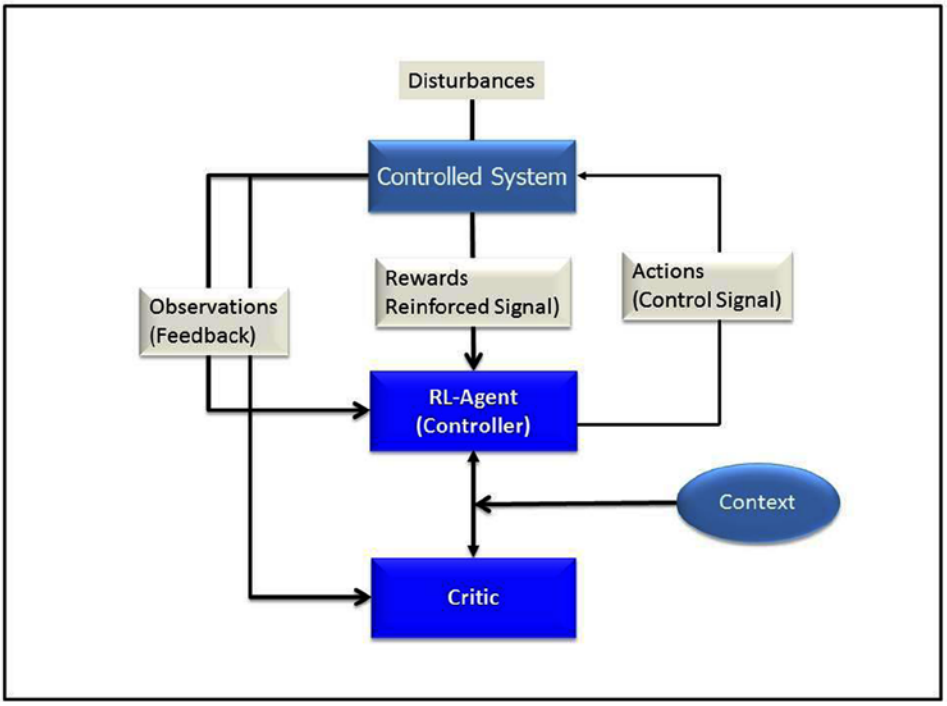

Figure 1-2. *High-level flow of reinforcement learning*

- ***Transductive learning*** (aka *transductive inference*) attempts to predict exclusive model functions on specific test cases by using additional observations on the training dataset in relation to the new cases (Vapnik 1998). A local model is established by fitting new individual observations (the training data) into a single point in space—this, in contrast to the global model, in which new data have to fit into the existing model without postulating any specific information related to the location of that data point in space. Although the new data may fit into the global model to a certain extent (with some error), thereby creating a global model that would represent the entire problem, space is a challenge and may not be necessary in all cases. In general, if you experience discontinuities during the model development for a given problem space, you can synthesize multiple models at the discontinuous boundaries. In this case, newly observed data are the processed through the model that fulfill the boundary conditions in which the model is valid.

- ***Inductive inference*** estimates the model function based on the relation of data to the entire hypothesis space, and uses this model to forecast output values for examples beyond the training set. These functions can be defined using one of the many representation schemes, including linear weighted polynomials, logical rules, and probabilistic descriptions, such as Bayesian networks. Many statistical learning methods start with initial solutions for the hypothesis space and then evolve them iteratively to reduce error. Many popular algorithms fall into this category, including SVMs (Vapnik 1998), *neural network* (NN) models (Carpenter and Grossberg 1991), and neuro-fuzzy algorithms (Jang 1993). In certain cases, one may apply a *lazy learning* model, in which the generalization process can be an ongoing task that effectively develops a richer hypothesis space, based on new data applied to the existing model.

Popular Machine Learning Algorithms

This section describes in turn the top 10 most influential data mining algorithms identified by the IEEE International Conference on Data Mining (ICDM) in December 2006: C4.5, *k*-means, SVMs, Apriori, estimation maximization (EM), PageRank, AdaBoost, *k*–nearest neighbors (*k*-NN), naive Bayes, and classification and regression trees (CARTs) (Wu et al. 2008).

C4.5

C4.5 classifiers are one of the most frequently used categories of algorithms in data mining. A C4.5 classifier inputs a collection of cases wherein each case is a sample preclassified to one of the existing classes. Each case is described by its *n*-dimensional vector, representing attributes or features of the sample. The output of a C4.5 classifier can accurately predict the class of a previously unseen case. C4.5 classification algorithms generate classifiers that are expressed as decision trees by synthesizing a model based on a tree structure. Each node in the tree structure characterizes a feature, with corresponding branches representing possible values connecting features and leaves representing the class that terminates a series of nodes and branches. The class of an instance can be determined by tracing the path of nodes and branches to the terminating leaf.

Given a set *S* of instances, C4.5 uses a divide-and-conquer method to grow an initial tree, as follows:

- If all the samples in the list *S* belong to the same class, or if the list *S* is small, then create a leaf node for the decision tree and label it with the most frequent class.
- Otherwise, the algorithm selects an attribute-based test that branches *S* into multiple subbranches (*partitions*) (*S1*, *S2*, ...), each representing the outcome of the test. The tests are placed at the root of the tree, and each path from the root to the leaf becomes a *rule script* that labels a class at the leaf. This procedure applies to each subbranch recursively.
- Each partition of the current branch represents a child node, and the test separating *S* represents the branch of the tree.

This process continues until every leaf contains instances from only one class or further partition is not possible. C4.5 uses tests that select attributes with the highest normalized information gain, enabling disambiguation of the classification of cases that may belong to two or more classes.

k-Means

The k-*means* algorithm is a simple iterative clustering algorithm (Lloyd 1957) that partitions *N* data points into *K* disjoint subsets S_j so as to minimize the sum-of-squares criterion. Because the sum of squares is the squared Euclidean distance, this is intuitively the "nearest" mean,

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} | x_n - \mu_j |^2, \tag{1-5}$$

where
x_n = vector representing the n^{th} data point
μ_j = geometric centroid of the data points in S_j

The algorithm consists of a simple two-step re-estimation process:

1. *Assignment*: Data points are assigned to the cluster whose centroid is closest to that point.
2. *Update*: Each cluster centroid is recalculated to the center (mean) of all data points assigned to it.

These two steps are alternated until a stopping criterion is met, such that there is no further change in the assignment of data points. Every iteration requires $N \times K$ comparisons, representing the time complexity of one iteration.

Support Vector Machines

Support vector machines (SVMs) are supervised learning methods that analyze data and recognize patterns. SVMs are primarily used for classification, regression analysis, and novelty detection. Given a set of training data in a two-class learning task, an SVM training algorithm constructs a model or classification function that assigns new observations to one of the two classes on either side of a hyperplane, making it a nonprobabilistic binary linear classifier (Figure 1-3). An SVM model maps the observations as points in space, such that they are classified into a separate partition that is divided by the largest distance to the nearest observation data point of any class (the *functional margin*). New observations are then predicted to belong to a class based on which side of the partition they fall. Support vectors are the data points nearest to the hyperplane that divides the classes. Further details of support vector machines are given in Chapter 4.

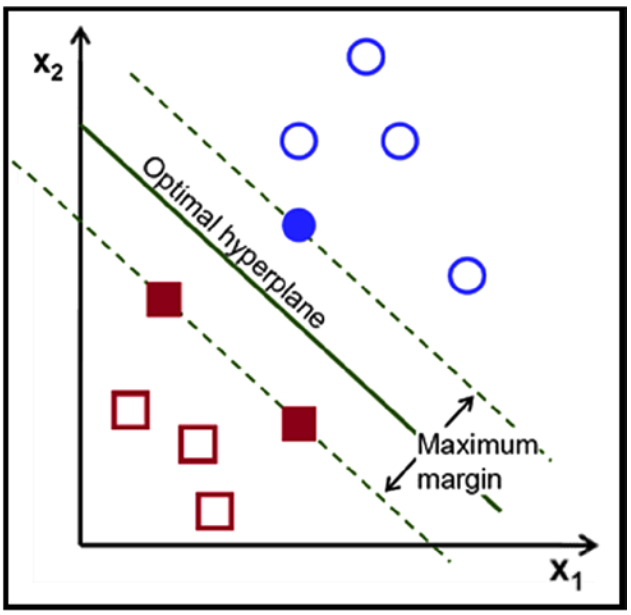

Figure 1-3. *The SVM algorithm finds the hyperplane that maximizes the largest minimum distance between the support vectors*

Apriori

Apriori is a data mining approach that discovers frequent itemsets by using candidate generation (Agrawal and Srikant 1994) from a transactional database and highlighting association rules (general trends) in the database. It assumes that any subset of a frequently occurring pattern must be frequent. Apriori performs breadth-first search to scan frequent 1-itemsets (that is, itemsets of size 1) by accumulating the count for each item that satisfies the minimum support requirement. The set of frequent 1-itemsets is used to find the set of frequent 2-itemsets, and so on. This process iterates until no more frequent k-itemsets can be found. The Apriori method that identifies all the frequent itemsets can be summarized in the following three steps:

1. Generate candidates for frequent $k + 1$-itemsets (of size $k + 1$) from the frequent k-itemsets (of size k).
2. Scan the database to identify candidates for frequent $k + 1$-itemsets, and calculate the support of each of those candidates.
3. Add those itemsets that satisfy the minimum support requirement to frequent itemsets of size $k + 1$.

Thanks in part to the simplicity of the algorithm, it is widely used in data mining applications. Various improvements have been proposed, notably, the *frequent pattern growth* (FP-growth) extension, which eliminates candidate generation. Han et al. (Han, Pei, and Yin 2000) propose a *frequent pattern tree* (FP-tree) structure, which stores and compresses essential information to interpret frequent patterns and uses FP-growth for mining the comprehensive set of frequent patterns by pattern fragment growth. This Apriori technique enhancement constructs a large database that contains all the essential information and compresses it into a highly condensed data structure. In the subsequent step, it assembles a *conditional-pattern base* which represents a set of counted patterns that co-occur relative to each item. Starting at the frequent header table, it traverses the FP-tree by following each frequent item and stores the prefix paths of those items to produce a conditional pattern base. Finally, it constructs a *conditional FP-tree* for each of the frequent items of the conditional pattern base. Each node in the tree represents an item and its count. Nodes sharing the same label but residing on different subtrees are conjoined by a node–link pointer. The position of a node in the tree structure represents the order of the frequency of an item, such that a node closer to the root may be shared by more transactions in a transactional database.

Estimation Maximization

The *estimation–maximization* (EM) algorithm facilitates parameter estimation in probabilistic models with incomplete data. EM is an iterative scheme that estimates the MLE or MAP of parameters in statistical models, in the presence of hidden or latent variables. The EM algorithm iteratively alternates between the steps of performing an expectation (E), which creates a function that estimates the probability distribution over possible completions of the missing (unobserved) data, using the current estimate for the parameters, and performing a maximization (M), which re-estimates the parameters, using the current completions performed during the E step. These parameter estimates are iteratively employed to estimate the distribution of the hidden variables in the subsequent E step. In general, EM involves running an iterative algorithm with the following attributes: (a) observed data, X; (b) latent (or missing) data, Z; (c) unknown parameter, θ; and (d) a likelihood function, $L(\theta; X, Z) = P(X, Z|\theta)$. The EM algorithm iteratively calculates the MLE of the marginal likelihood using a two-step method:

1. *Estimation (E)*: Calculate the expected value of the log likelihood function, with respect to the conditional distribution of Z, given X under the current estimate of the parameters $\theta(t)$, such that

$$Q(\theta \mid \theta(t)) = E_{Z|X,\theta(t)}\left[\log L(\theta; X, Z)\right]. \tag{1-6}$$

2. *Maximization (M)*: Find the parameter that maximizes this quantity:

$$\theta(t+1) = \arg_{\theta} \max Q(\theta \mid \theta(t)). \tag{1-7}$$

PageRank

PageRank is a link analysis search algorithm that ranks the elements of hyperlinked documents on the World Wide Web for the purpose of measuring their importance, relative to other links. Developed by Larry Page and Sergey Bin, PageRank produces static rankings that are independent of the search queries. PageRank simulates the concept of prestige in a social network. A hyperlink to a page counts as a vote of support. Additionally, PageRank interprets a hyperlink from source page to target page in such a manner that the page with the higher rank improves the rank of the linked page (the source or target). Therefore, backlinks from highly ranked pages are more significant than those from average pages. Mathematically simple, PageRank can be calculated as

$$r(P) = \sum_{Q \in B_p} \frac{r(Q)}{|Q|}, \tag{1-8}$$

where
$r(P)$ = rank of the page P
B_p = the set of all pages linking to page P
$|Q|$ = number of links from page Q
$r(Q)$ = rank of the page Q

AdaBoost (Adaptive Boosting)

AdaBoost is an ensemble method used for constructing strong classifiers as linear combinations of simple, weak classifiers (or rules of thumb) (Freund and Schapire 1997). As in any ensemble method, AdaBoost employs multiple learners to solve a problem with better generalization ability and more accurate prediction. The strong classifier can be evaluated as a linear combination of weak classifiers, such that

$$H(x) = \sum_{t=1}^{T} \beta_t \cdot h_t(x),$$

where
$H(x)$ = strong classifier
$h_t(x)$ = weak classifier (*feature*)

The Adaboost algorithm may be summarized as follows:

Input:
Data-Set $I = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \cdots, (x_m, y_m)\}$,
Base learning algorithm L
Number of learning rounds T

Process:

$D_1^i = \frac{1}{m}$ *// Initialize weight distribution*
FOR (t = 1 to T) DO ***// Run the loop for t = T iterations***
$h_t = L(I, D_t)$ *// Train a weak learner h_t from I using D_t*
$\in_t = \sum_i D_t^i \mid h_t(x_i) - y_i \mid$ *// calculate the error of h_t*

$$\beta_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \qquad \textit{// calculate the weight of } h_t$$

$$D_{t+1}^{i} = \frac{D_t^i}{Z_t}\cdot e^{(-\beta_t \cdot y_i \cdot h_t(x_i))} \qquad \textit{// Update the distribution,}$$

$$\textit{// } Z_t \textit{ is the normalization factor}$$

END

Output:

$$H(x) = sign\left(\sum_{t=1}^{T}\beta_t h_t(x)\right) \quad \text{// Strong classifier}$$

The AdaBoost algorithm is adaptive, inasmuch as it uses multiple iterations to produce a strong learner that is well correlated with the true classifier. As shown above, it iterates by adding weak learners that are slightly correlated with the true classifier. As part of the adaptation process, the weighting vector adjusts itself to improve upon misclassification in previous rounds. The resulting classifier has a greater accuracy than the weak learners' classifiers. AdaBoost is fast, simple to implement, and flexible insofar as it can be combined with any classifier.

k-Nearest Neighbors

The k-*nearest neighbors* (k-*NN*) classification methodology identifies a group of *k* objects in the training set that are closest to the test object and assigns a label based on the most dominant class in this neighborhood. The three fundamental elements of this approach are

- an existing set of labeled objects
- a distance metric to estimate distance between objects
- the number of nearest neighbors (*k*)

To classify an unlabeled object, the distances between it and labeled objects are calculated and its *k*-nearest neighbors are identified. The class labels of these nearest neighbors serve as a reference for classifying the unlabeled object. The *k-NN* algorithm computes the similarity distance between a training set, $(x, y) \in I$, and the test object, $z = (\hat{x}, \hat{y})$, to determine its nearest-neighbor list, I_z. *x* represents the training object, and *y* represents the corresponding training class. $\hat{x}$ and $\hat{y}$ represent the test object and its class, respectively. The algorithm may be summarized as follows:

Input:
Training object $(x, y) \in I$ and test object $z = (\hat{x}, \hat{y})$

Process:
Compute distance $d = (\hat{x}, x)$ between *z* and every object $(x, y) \in I$.
Select $I_z \subseteq I$, the set of *k* closest training objects to *z*.

Output (Majority Class):

$$\hat{y} = \arg_v max \sum_{(x_i, y_i)\in I_Z} F(v = y_i)$$

F(.) = 1 if argument (.) is TRUE and 0 otherwise, *v* is the class label.

The value of *k* should be chosen carefully. A smaller value can result in noisy behavior, whereas a larger value may include too many points from other classes.

Naive Bayes

Naive Bayes is a simple probabilistic classifier that applies Bayes' theorem with strong (naive) assumption of independence, such that the presence of an individual feature of a class is unrelated to the presence of another feature.

Assume that input features $x_1, x_2 \cdots x_n$ are conditionally independent of each other, given the class label Y, such that

$$P(x_1, x_2 \cdots x_n \mid Y) = \prod_{i=1}^{n} P(x_i \mid Y) \tag{1-9}$$

For a two-class classification ($i = 0,1$), we define $P(i|x)$ as the probability that measurement vector $x = \{x_1, x_2 \cdots x_n\}$ belongs to class *i*. Moreover, we define a classification score

$$\frac{P(1 \mid x)}{P(0 \mid x)} = \frac{\prod_{j=1}^{n} f(x_j \mid 1) P(1)}{\prod_{j=1}^{n} f(x_j \mid 0) P(0)} = \frac{P(1)}{P(0)} \prod_{j=1}^{n} \frac{f(x_j \mid 1)}{f(x_j \mid 0)} \tag{1-10}$$

$$\ln \frac{P(1 \mid x)}{P(0 \mid x)} = \ln \frac{P(1)}{P(0)} + \sum_{j=1}^{n} \ln \frac{f(x_j \mid 1)}{f(x_j \mid 0)}, \tag{1-11}$$

where $P(i|x)$ is proportional to $f(x|i)P(i)$ and $f(x|i)$ is the conditional distribution of x for class i objects.

The naive Bayes model is surprisingly effective and immensely appealing, owing to its simplicity and robustness. Because this algorithm does not require application of complex iterative parameter estimation schemes to large datasets, it is very useful and relatively easy to construct and use. It is a popular algorithm in areas related to text classification and spam filtering.

Classification and Regression Trees

A *classification and regression tree* (CART) is a nonparametric decision tree that uses a binary recursive partitioning scheme by splitting two child nodes repeatedly, starting with the root node, which contains the complete learning sample (Breiman et al. 1984). The tree-growing process involves splitting among all the possible splits at each node, such that the resulting child nodes are the "purest." Once a CART has generated a "maximal tree," it examines the smaller trees obtained by pruning away the branches of the maximal tree to determine which contribute least to the overall performance of the tree on training data. The CART mechanism is intended to yield a sequence of nested pruned trees. The right-sized, or "honest," tree is identified by evaluating the predictive performance of every tree in the pruning sequence.

Challenging Problems in Data Mining Research

Data mining and knowledge discovery have become fields of interdisciplinary research in the areas related to database systems, ML, intelligent information systems, expert systems, control theory, and many others. Data mining is an important and active area of research but not one without theoretical and practical challenges from working with very large databases that may be noisy, incomplete, redundant, and dynamic in nature. A study by Yang and Wu (2006) reviews the most challenging problems in data mining research, as summarized in the following sections.

Scaling Up for High-Dimensional Data and High-Speed Data Streams

Designing classifiers that can handle very high-dimensional features extracted through high-speed data streams is challenging. To ensure a decisive advantage, data mining in such cases should be a continuous and online process. But, technical challenges prevent us from computing models over large amounts streaming data in the presence of *environment drift* and *concept drift*. Today, we try to solve this problem with incremental mining and offline model updating to maintain accurate modeling of the current data stream. Information technology challenges are being addressed by developing in-memory databases, high-density memories, and large storage capacities, all supported by high-performance computing infrastructure.

Mining Sequence Data and Time Series Data

Efficient classification, clustering, and forecasting of sequenced and time series data remain an open challenge today. Time series data are often contaminated by noise, which can have a detrimental effect on short-term and long-term prediction. Although noise may be filtered, using signal-processing techniques or smoothening methods, lags in the filtered data may result. In a closed-loop environment, this can reduce the accuracy of prediction, because we may end up overcompensating or underprovisioning the process itself. In certain cases, lags can be corrected by differential predictors, but these may require a great deal of tuning the model itself. Noise-canceling filters placed close to the data I/O block can be tuned to identify and clean the noisy data before they are mined.

Mining Complex Knowledge from Complex Data

Complex data can exist in many forms and may require special techniques to extract the information useful for making real-world decisions. For example, information may exist in a graphical form, requiring methods for discovering graphs and structured patterns in large data. Another complexity may exist in the form of *non—independent-and-identically-distributed* (non-iid) data objects that cannot be mined as an independent single object. They may share relational structures with other data objects that should be identified.

State-of-the-art data mining methods for unstructured data lack the ability to incorporate domain information and knowledge interface for the purpose of relating the results of data mining to real-world scenarios.

Distributed Data Mining and Mining Multi-Agent Data

In a distributed data sensing environment, it can be challenging to discover distributed patterns and correlate the data streamed through different probes. The goal is to minimize the amount of data exchange and reduce the required communication bandwidth. Game-theoretic methodologies may be deployed to tackle this challenge.

Data Mining Process-Related Problems

Autonomous data mining and cleaning operations can improve the efficiency of data mining dramatically. Although we can process models and discover patterns at a fast rate, major costs are incurred by preprocessing operations such as data integration and data cleaning. Reducing these costs through automation can deliver a much greater payoff than attempting to further reduce the cost of model-building and pattern-finding.

Security, Privacy, and Data Integrity

Ensuring users' privacy while their data are being mined is critical. Assurance of the knowledge integrity of collected input data and synthesized individual patterns is no less essential.

Dealing with Nonstatic, Unbalanced, and Cost-Sensitive Data

Data is dynamic and changing continually in different domains. Historical trials in data sampling and model construction may be suboptimal. As you retrain a current model based on new training data, you may experience a learning drift, owing to different selection biases. Such biases need to be corrected dynamically for accurate prediction.

Summary

This chapter discussed the essentials of ML through key terminology, types of ML, and the top 10 data mining and ML algorithms. Owing to the explosion of data on the World Wide Web, ML has found widespread use in web search, advertising placement, credit scoring, stock market prediction, gene sequence analysis, behavior analysis, smart coupons, drug development, weather forecasting, big data analytics, and many more such applications. New uses for ML are being explored every day. Big data analytics and graph analytics have become essential components of cloud-based business development. The new field of data analytics and the applications of ML have also accelerated the development of specialized hardware and accelerators to improve algorithmic performance, big data storage, and data retrieval performance.

References

Agrawal, Rakesh, and Ramakrishnan Srikant. "Fast Algorithms for Mining Association Rules in Large Databases." In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94), September 12–15, 1994, Santiago de Chile, Chile*, edited by Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo. San Francisco: Morgan Kaufmann (1994): 487–499.

Breiman, Leo, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.

Carpenter, Gail A., and Stephen Grossberg. *Pattern Recognition by Self-Organizing Neural Networks*. Massachusetts: Cambridge, MA: Massachusetts Institute of Technology Press, 1991.

Freund, Yoav, and Robert E. Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting." *Journal of Computer and System Sciences* 55, no. 1 (1997): 119–139.

Han, Jiawel, Jian Pei, and Yiwen Yin. "Mining Frequent Patterns without Candidate Generation." In *SIGMOD/PODS '00: ACM international Conference on Management of Data and Symposium on Principles of Database Systems, Dallas, TX, USA, May 15–18, 2000*, edited by Weidong Chen, Jeffrey Naughton, Philip A. Bernstein. New York: ACM (2000): 1–12.

Jang, J.-S. R. "ANFIS: Adaptive-Network-Based Fuzzy Inference System." *IEEE Transactions on Systems, Man and Cybernetics* 23, no. 3 (1993): 665–685.

Kohavi, Ron, and Foster Provost. "Glossary of Terms." *Machine Learning* 30, no. 2–3 (1998): 271–274.

Lloyd, Stuart P. “Least Squares Quantization in PCM,” in special issue on quantization, *IEEE Transactions on Information Theory*, IT-28, no. 2(1982): 129–137.

Samuel, Arthur L. “Some Studies in Machine Learning Using the Game of Checkers,” *IBM Journal of Research and Development* 44:1.2 (1959): 210–229.

Turing, Alan M. “Computing machinery and intelligence.” *Mind* (1950): 433–460.

Vapnik, Vladimir N. *Statistical Learning Theory*. New York: Wiley, 1998.

Wu, Xindong, Vipin Kumar, Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. “Top 10 Algorithms in Data Mining.” *Knowledge and Information Systems* 14 (2008): 1–37.

Yang, Qiang, and Xindong Wu. “10 Challenging Problems in Data Mining Research.” *International Journal of Information Technology and Decision Making* 5, no. 4 (2006): 597–604.