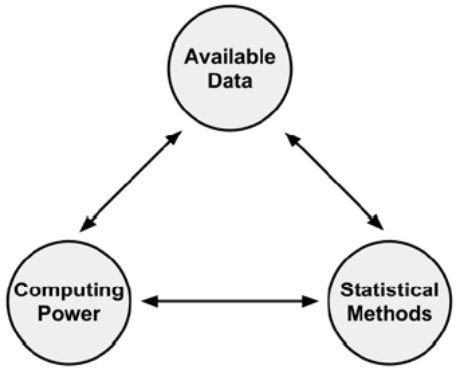
MODULE 2

Lecture Notes

APARNA S BALAN
DEPARTMENT OF COMPUTER APPLICATIONS
Vidya Academy of Science & Technology

Introduction to machine learning



Machine learning is the process of teaching a computer to learn. Vast quantities of data that can be directly processed by machines is available in the digital era. Much of this information has the potential to influence decision making. The field of study interested in the development of computer algorithms for transforming data into intelligent action is known as machine learning. Growth in data necessitated additional computing power, which in turn spurred the development of statistical methods for analyzing large datasets. This created a cycle of advancement allowing even larger and more interesting data to be collected. A machine learning algorithm takes data and identifies patterns that can be used for action.

Machine learning is primarily interested in making sense of complex data. It is used to predict the outcomes of elections, identify and filter spam messages from e-mail, foresee criminal activity, automate traffic signals according to road conditions, produce financial estimates of storms and natural disasters, examine customer churn, create auto-piloting planes and auto-driving cars, identify individuals with the capacity to donate, target advertising to specific types of consumers etc.

How machines learn

Machine learning techniques actually learn to transform data into actionable knowledge. The basic learning process is divided into three components: data input, abstraction and generalization. *Data input* utilizes observation, memory storage, and recall to provide a factual basis for further reasoning. *Abstraction* involves the translation of data into broader representations. *Generalization* uses abstracted data to form a basis for action.

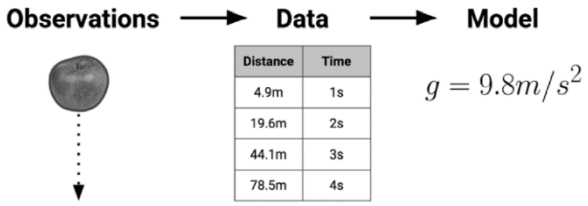
Instead of memorizing the raw data, a machine performs knowledge abstraction where learning strategies are used to create an outline or a concept map. The tools define relationships among information and in doing so, depict difficult ideas without needing to memorize them word-for-word. Generalization requires abstracted data, as well as a higher-level understanding of how to apply such knowledge to unforeseen topics.

Data storage

All learning must begin with data. Computers utilize data storage as a foundation for more advanced reasoning. Computers have capabilities of short and long-term recall using hard disk drives, flash memory, and random access memory in combination with a central processing unit. Instead of storing the entire data, a better approach is to spend time selectively, storing a small set of representative ideas while developing strategies on how the ideas relate and how to use the stored information.

Abstraction

Abstraction is the process of assigning meaning to stored data. Knowledge representation is the process of formation of logical structures that assist in turning raw sensory information into a meaningful insight. During a machine's process of knowledge representation, the computer summarizes stored raw data using a model, an explicit description of the patterns within the data. There are different types of models representation like mathematical equations, relational diagrams such as trees and graphs, logical if/else rules, groupings of data known as clusters. Model selection is influenced by learning task and data on hand. The process of fitting a model to a dataset is known as training. When the model has been trained, the data is transformed into an abstract form that summarizes the original information. Model might result in the discovery of previously unseen relationships among data.



This image is an example clearly represents the abstraction process. The 'g' term in the model explains observations of falling objects, which is nothing but an abstraction of the collected data.

Generalization

Abstraction process identifies countless underlying patterns and different ways to model these patterns. The term generalization describes the process of turning abstracted knowledge into a form that can be utilized for future actions. Thus, generalization reduces the patterns discovered during the abstraction stage into manageable chunks. So, generalization selects only the most relevant patterns.

It involves the reduction of set into a manageable number of important findings. Generalization uses heuristic approach (educated guesses) to find the most useful inferences. The heuristics employed by machine learning algorithms sometimes result in erroneous conclusions.

Evaluation

Evaluation occurs after a model has been trained on an initial training dataset. The model is evaluated on a new test dataset in order to judge how well its characterization of the training data generalizes to new, unseen data. Models fail to perfectly generalize due to the problem of noise (unexplained or unexplainable variations in data).

Trying to model noise is the reason for overfitting. Because most noisy data is unexplainable by definition, attempting to explain the noise will result in erroneous conclusions that do not generalize well to new cases. Efforts to explain the noise will also result in more complex models. A model that seems to perform well during training, but does poorly during evaluation, is said to be overfitted to the training dataset, as it does not generalize well to the test dataset.

Machine learning in practice

Following five-steps needs to be completed to apply the learning process to real-world tasks.

Data collection: The data collection step involves gathering the learning material an algorithm will use to generate actionable knowledge. The data is combined to a single source like a text file, spreadsheet, or database.

Data exploration and preparation: The quality of a machine learning project is based on the quality of its input data. This stage learns more about the data and its nuances during a practice called data exploration. Then the data is prepared for the learning process. This involves fixing or cleaning "messy" data, eliminating unnecessary data, and recoding the data to conform to the learner's expected inputs.

Model training: An appropriate algorithm is selected, which will represent the data in the form of a model.

Model evaluation: Models need to be evaluated because machine learning model results in a biased solution to the learning problem. Depending on the type of model, appropriate test dataset can be used to get the accuracy of the model or specific measures can be developed for evaluation.

Model improvement: If better performance is needed, it is necessary to utilize more advanced strategies to increase the performance of the model. Sometimes, it may be necessary to switch to a different type of model or the data needs to be supplemented with additional data or more work needs to be done with data exploration and data preparation.

After these steps are completed, if the model performs well, it can be *deployed* for its intended task. The successes and failures of the deployed model might even provide additional data to train the next generation learner.

Types of machine learning algorithms

Machine learning algorithms are divided into categories according to their purpose. A **predictive model** is used for tasks that involve the prediction of one value using other values in the dataset. In this approach the learning algorithm attempts to discover and model the relationship between the target feature (the feature being predicted) and the other features. A predictive model can also be used to predict past events.

The process of training a predictive model is known as **supervised learning** because predictive models are given clear instruction on what they need to learn and how they are intended to learn it. The supervision does not refer to human involvement, but to the fact that the target values provide a way for the learner to know how well it has learned the desired task. Given a set of data, a supervised learning algorithm attempts to optimize a function (the model) to find the combination of feature values that result in the target output.

The commonly used supervised machine learning task of predicting which category an example belongs to is known as **classification**. In classification, the target feature to be predicted is a categorical feature known as the class, and is divided into categories called levels.

Supervised learners can also be used to predict numeric data. To predict such numeric values, a common form of numeric prediction fits linear regression models to the input data. **Regression models** are widely used for forecasting.

A **descriptive model** is used for tasks that would benefit from the insight gained from summarizing. Descriptive model does not give importance to a single feature. The process of training a descriptive model is called **unsupervised learning** because there are no target features.

The descriptive modeling task called pattern discovery is used to identify useful associations within data. **Pattern discovery** is often used for **market basket analysis** on transactional purchase data. The goal of the model is to identify items that are frequently purchased together.

The descriptive modeling task of dividing a dataset into homogeneous groups is called **clustering**. This is sometimes used for **segmentation analysis** that identifies groups of individuals with similar behaviour or demographic information.

A class of machine learning algorithms known as **meta-learners** is not tied to a specific learning task, but is rather focused on learning how to learn more effectively. Meta-learning algorithm uses the result of some learnings to inform additional learning.

Lazy learning: Classification using K-NN Algorithm

Nearest neighbour classifiers are defined by their characteristic of classifying unlabeled examples by assigning them the class of the most similar labeled examples. Nearest neighbour classifiers are well-suited for classification tasks where relationships among the features and the target classes are numerous, complicated, or otherwise extremely difficult to understand, yet the items of similar class type tend to be fairly homogeneous. It can also be used in situation where the relationship is clearly visible, but difficult to define. But if there is not a clear distinction among the groups, the algorithm is not well-suited for identifying the boundary.

kNN algorithm

The nearest neighbours approach to classification is utilized by the kNN algorithm. The kNN algorithm begins with a training dataset made up of examples that are classified into several categories, as labeled by a nominal variable. Test dataset containing unlabeled examples with the same features as the training data. For each record in the test dataset, kNN identifies k records in the training data that are the "nearest" in similarity, where k is an integer specified in advance. The unlabeled test instance is assigned the class of the majority of the k nearest neighbours.

The kNN algorithm treats the features as coordinates in a multidimensional feature space. The number of features define the dimension (feature space). A distance function or a formula that measures the similarity between two instances, is used to locate nearest neighbors.

Measuring similarity with distance

There are many different ways to calculate distance. Traditionally, the kNN algorithm uses Euclidean distance which is shortest direct route between two points. The distance formula involves comparing the values of each feature. Euclidean distance can be measure by the following formulae

$$dist(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

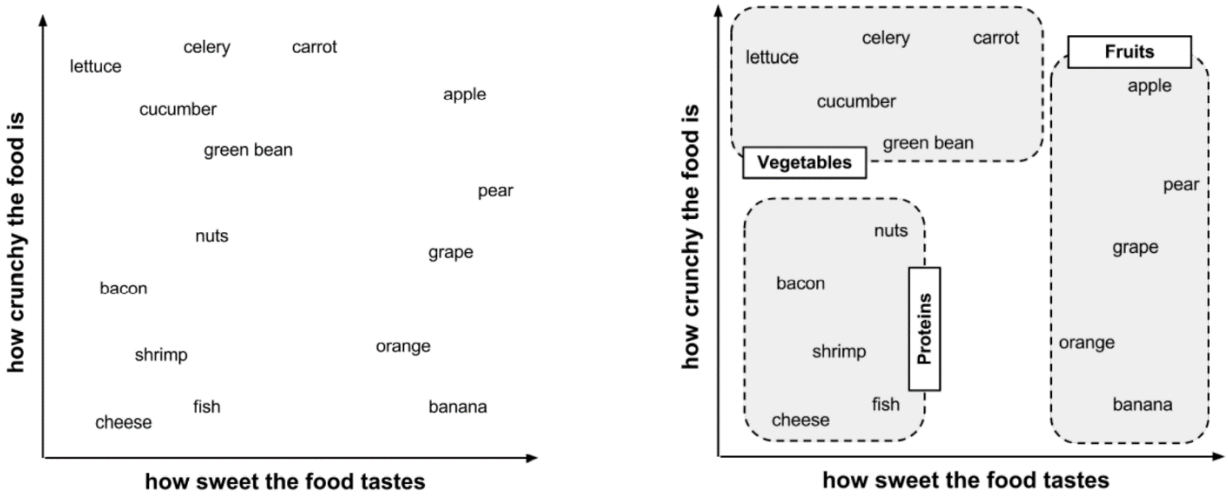
Where p and q are the examples to be compared, each having n features. The term p_1 refers to the value of the first feature of example p, while q_1 refers to the value of the first feature of example q.

The algorithm calculates the distance between the test data point and several of its closest neighbors. 1NN (k=1) classifier will classify a data point by assigning the point to the single nearest neighbor. The kNN algorithm with k = n performs a vote among the 'n' nearest neighbors. The data point is represented by the label of the majority class among 'n' neighbors.

Example

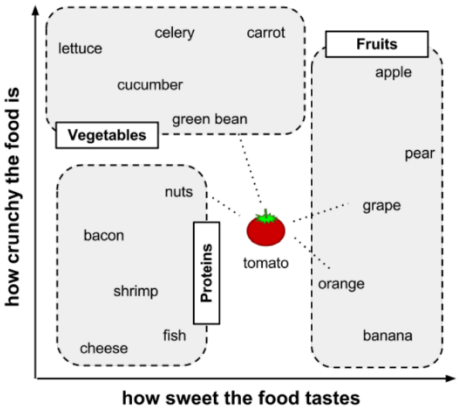
ingredient	sweetness	crunchiness	food type
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit
carrot	7	10	vegetable
celery	3	10	vegetable
cheese	1	1	protein
grape	8	5	fruit
green bean	3	7	vegetable
nuts	3	6	protein
orange	7	3	fruit

In the following table contains the details of ingredients which are used for cooking. Only two features (sweetness and crunchiness) of each ingredient are recorded here. The first is a measure from 1 to 10 of how sweet the ingredient tastes, and the second is a 1 to 10 score of how crunchy the ingredient is. Each ingredient is labeled as one of three types of food: fruits, vegetables, or proteins. Only 10 data points out of 15 are given in this table.



As the dataset includes only two features, the feature space is two-dimensional. It can be depicted using a cartesian coordinate system. This plot shows 15 ingredients, with the x dimension indicating the ingredient's sweetness and the y dimension indicating the crunchiness. The above table shows only a part of the actual data points. Similar types of food tend to be grouped closely together. As illustrated in the figure, vegetables tend to be crunchy but not

sweet, fruits tend to be sweet and either crunchy or not crunchy, while proteins tend to be neither crunchy nor sweet.



Now consider a test sample 'tomato'. Nearest neighbor approach can be used to determine which class is a better fit. Calculate the distance between the tomato (sweetness=6, crunchiness=4) and several of its closest neighbors as follows dist (tomato, green bean) = $\sqrt{((6-3)^2+(4-7)^2)}$ = 4.2

In a similar method, calculate the distance between the tomato and several of its closest neighbors as follows:

ingredient	sweetness	crunchiness	food type	distance to the tomato
grape	8	5	fruit	*sqrt((6 - 8)^2 + (4 - 5)^2) = 2.2*
green bean	3	7	vegetable	*sqrt((6 - 3)^2 + (4 - 7)^2) = 4.2*
nuts	3	6	protein	*sqrt((6 - 3)^2 + (4 - 6)^2) = 3.6*
orange	7	3	fruit	*sqrt((6 - 7)^2 + (4 - 3)^2) = 1.4*

In k=1 (1NN), tomato is assigned the class of its single nearest neighbor. By analyzing the distance value from the above table, it is clear that orange has the lowest value. Thus, the class of orange, i.e. fruit, is assigned as the class label of tomato.

In k=3 (3NN) , three nearest neighbors (orange, grape, and nuts) are considered. Because the majority class among these neighbors is fruit (2 of the 3 votes), the tomato again is classified as a fruit.

Choice of k

The balance between overfitting and underfitting the training data is a problem known as the bias-variance trade-off. Choosing a large k reduces the impact or variance caused by noisy data, but can bias the learner such that it runs the risk of ignoring small, but important patterns.

If k is set to the total number of observations in the training data, the model would always predict the majority class. Using a single nearest neighbor allows noisy data or outliers, to influence the classification of examples. The best k value is somewhere between these two extremes. Choosing k depends on the difficulty of the concept to be learned and the number of records in the training data. One common practice is to set k equal to the square root of the number of training examples. An alternative approach is to test several k values on a variety of test datasets and choose the one that delivers the best classification performance.

Preparing data for use with k-NN

The distance formula used in kNN is dependent on how features are measured. So, features are transformed (or rescaled) to a standard range prior to applying the kNN algorithm. For example, if certain features have much larger values than others, the distance measurements will be strongly dominated by the larger values. So, features are rescaled in various ways such that each one contributes relatively equally to the distance formula. There are several ways to scaling: min-max normalization, z-score standardization.

The traditional method of rescaling features for kNN is **min-max normalization**. This process transforms a feature such that all of its values fall in a range between 0 and 1. The formula for normalizing a feature is $X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$

Another common transformation is called **z-score standardization**. The z-scores fall in an unbounded range of negative and positive numbers. They have no predefined minimum and maximum. The formula for z-score standardization is $X_{new} = \frac{X - \mu}{\sigma} = \frac{X - Mean(X)}{StdDev(X)}$

The Euclidean distance formula is not defined for nominal data. Therefore, to calculate the distance between nominal features, first convert them into a numeric format. This can be done using **dummy coding**, where a value of 1 indicates one category, and 0 indicates other categories. An n-category nominal feature can be dummy coded by creating binary indicator variables for (n-1) levels of the feature.

Strengths

- Simple and effective
- Makes no assumptions about the underlying data distribution
- Fast training phase

Weaknesses

- Does not produce a model, which limits the ability to find novel insights in relationships among features
- Slow classification phase
- Requires a large amount of memory
- Nominal features and missing data require additional processing

Why is the kNN algorithm lazy?

Classification algorithms based on nearest neighbor methods are considered lazy learning algorithms because no abstraction occurs. The abstraction and generalization processes are skipped altogether. If definition of learning is strictly followed, a lazy learner is not really learning

anything. Instead, it merely stores the training data verbatim. This allows the training phase to occur very rapidly, with a potential downside being that the process of making predictions tends to be relatively slow. Due to the heavy reliance on the training instances, lazy learning is also known as instance-based learning or rote learning. Although kNN classifiers may be considered lazy, they are still quite powerful.

Probabilistic learning

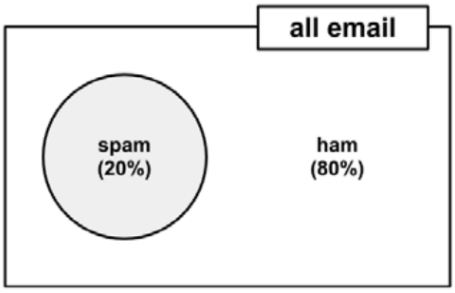
Probabilistic learning estimates are based on probabilistic methods, or methods concerned with describing uncertainty. It uses data on past events to derive future events. The Naive Bayes uses principles of probability for classification. Naive Bayes uses data about prior events to estimate the probability of future events.

Understanding naive Bayes

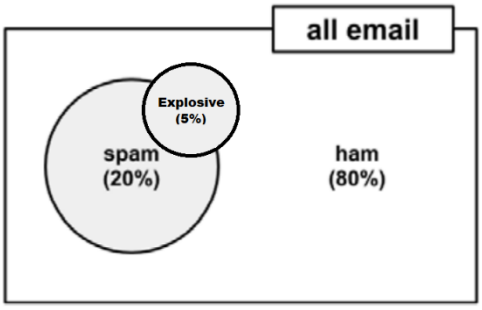
Statistical ideas used in the naive Bayes algorithm is from the work of mathematician Thomas Bayes, who developed foundational mathematical principles known as Bayesian methods for describing the probability of events, and how probabilities should be revised in light of additional information. The probability of an event is a number between 0 percent and 100 percent that captures the chance that the event will occur given the available evidence. A probability of 0 percent indicates that the event definitely will not occur, while a probability of 100 percent indicates that the event certainly will occur.

Classifiers based on Bayesian methods utilize training data to calculate an observed probability of each class based on feature values. When the classifier is used later on unlabeled data, it uses the observed probabilities to predict the most likely class for the new features. Bayesian classifiers are best applied to problems in which the information from numerous attributes should be considered simultaneously in order to estimate the probability of an outcome. While many algorithms ignore features that have weak effects, Bayesian methods utilize all available features for predictions. Bayesian classifiers have been used for text classification, intrusion detection or anomaly detection in computer networks, diagnosing medical conditions etc.

In probability *events* are possible outcomes and *trial* is a single opportunity for the event to occur. The probability of an event can be estimated from observed data by dividing the number of trials in which an event occurred by the total number of trials. For example, if 10 out of 50 email messages are spam, then the probability of spam can be estimated as 20 percent. The notation P(A) is used to denote the probability of event A, as in *P(spam)* = 0.20.



The total probability of all possible outcomes of a trial must always be 100 percent. Thus, if the trial only has two outcomes that cannot occur simultaneously, such as spam and ham (non-spam), then knowing the probability of either outcome reveals the probability of the other. For example, given the value P(spam) = 0.20, probability of ham can be calculated P(ham) = 1 – 0.20 = 0.80. This works because the events spam and ham are mutually exclusive and exhaustive. This means that the events cannot occur at the same time and are the only two possible outcomes. The notation P(¬A) can be used to denote the probability of event A not occurring, as in P(¬spam) = 0.80



But some events can be non-mutually exclusive. It implies that the occurrence of an even is connected to the occurrence of another event. For most people, this word 'Explosive' is only likely to appear in a spam message; its presence in a message is therefore a very strong piece of evidence that the email is spam. But if the mails are sent to an explosive manufacturing unit, the emails cannot be categorized as spam. This implies that not all spam messages contain the word Explosive, and not every email with the word Explosive is spam.

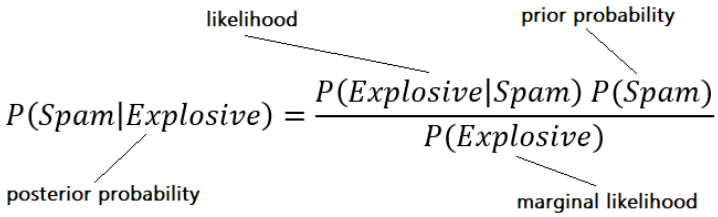
If the two events are totally unrelated, they are called **independent** events. If all events were independent, it would be impossible to predict any event using the data obtained by another. On the other hand, **dependent** events are the basis of predictive modeling. That means it is possible to predict an event based on the data of another event. **Joint probability** of two events A and B indicate how the probability of one event is related to the probability of the other ($P(A \cap B)$). Joint probability (the probability of both happening) of two independent events can be computed as $P(A \cap B) = P(A) * P(B)$. But if the events are dependent, bayes theorem can be used to define the relationship.

Conditional probability and Bayes' theorem

The relationships between dependent events can be described using Bayes' theorem. The notation P(A|B) can be read as the probability of event A given that event B occurred. This is known as conditional probability, since the probability of A is dependent (that is, conditional) on what happened with event B.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} = \frac{P(A \cap B)}{P(B)}$$

Probability that an incoming email was spam can be computed using Bayes' theorem. From the available data the probability that any prior message was spam is, 20 percent. This estimate is known as the **prior probability**. The probability that the word Explosive was used in previous spam messages is called the **likelihood** and the probability that Explosive appeared in any message at all is known as the **marginal likelihood**. By applying Bayes' theorem to this evidence, a **posterior probability** can be computed that measures how likely the message is to be spam. If the posterior probability is greater than 50 percent, the message is more likely to be spam than ham, and it should be filtered. The following formula is the Bayes' theorem for the given evidence:



	Explosive		
Frequency	**Yes**	**No**	**Total**
spam	4	16	20
ham	1	79	80
Total	5	95	100

A **frequency table** is constructed to calculate the components of Bayes' theorem. It records the number of times Explosive appeared in spam and ham messages. One dimension of the table indicates levels of the class variable (spam or ham), while the other dimension indicates levels for features (Explosive: yes or no). The frequency table can then be used to construct a likelihood table.

	Explosive		
Likelihood	**Yes**	**No**	**Total**
spam	4/20	16/20	20
ham	1/80	79/80	80
Total	5/100	95/100	100

The likelihood table reveals that $P(\text{Explosive}|\text{spam})$ = 4/20 = 0.20, indicating that the probability is 20 percent that a spam message contains the term Explosive. $P(\text{spam} \cap \text{Explosive})$ can be calculated as $P(\text{Explosive}|\text{spam})$ * $P(spam)$ = (4/20) * (20/100) = 0.04. Thus, the posterior probability is $P(\text{spam}|\text{Explosive})$ which can be computed as $P(\text{Explosive}|\text{spam})$ * $P(spam)$ / $P(\text{Explosive})$, or (4/20) * (20/100) / (5/100) = 0.80. The probability is 80 percent that a message is spam, given that it contains the word Explosive.

Naive Bayes algorithm for classification

The Naive Bayes algorithm describes a simple method to apply Bayes' theorem to classification problems. The Naive Bayes algorithm is named as such because it makes some

"naive" assumptions about the data. That is, Naive Bayes assumes that all of the features in the dataset are equally important and independent.

	Explosive(W1)		Money(W2)		Groceries(W3)		Unsubscribe(W4)		
Likelihood	**yes**	**No**	**yes**	**No**	**yes**	**No**	**yes**	**No**	**Total**
Spam	4/20	16/20	10/20	10/20	0/20	20/20	12/20	8/20	20
Ham	1/80	79/80	14/80	66/80	8/80	71/80	23/80	57/80	80
Total	5/100	95/100	24/100	76/100	8/100	91/100	35/100	65/100	100

The given example can be extended by adding a few additional terms to be monitored in addition to the term Explosive: Money, Groceries, and Unsubscribe. The Naive Bayes learner is trained by constructing a likelihood table for the appearance of these four words (labeled W1, W2, W3, and W4), as shown in the diagram for 100 e-mails.

Assume that an incoming message contains the terms Explosive and Unsubscribe, but does not contain either Money or Groceries. Now, the posterior probability to determine whether they are more likely to be spam or ham. Using Bayes' theorem, the problem can be defined as shown in the following formula. It captures the probability that a message is spam, given that Explosive = Yes, Money = No, Groceries = No and Unsubscribe = Yes.

$$P(Spam|W1 \cap \neg W2 \cap \neg W3 \cap W4) = \frac{P(W1 \cap \neg W2 \cap \neg W3 \cap W4|Spam)P(Spam)}{P(W1 \cap \neg W2 \cap \neg W3 \cap W4)}$$

This formula is computationally difficult to solve. Tremendous amount of memory is needed to store probabilities for all of the possible intersecting events. Naive Bayes assumes independence among events as so long as they are conditioned on the same class value. this assumption is called class-conditional independence. The above formulae can be simplified using the probability rule for independent events, which states that $P(A \cap B) = P(A) * P(B)$. Because the denominator does not depend on the class (spam or ham), it is treated as a constant value and can be ignored for the time being. Thus, the equals symbol is replaced by the proportional-to symbol in the formulae. This means that the conditional probability of spam can be expressed as

$$P(Spam|W1 \cap \neg W2 \cap \neg W3 \cap W4)$$

$$\propto P(W1|Spam)\, P(\neg W2|Spam)\, P(\neg W3|Spam)\, P(W4|Spam)\, P(Spam)$$

$$\propto (4/20) * (10/20) * (20/20) * (12/20) * (20/100)$$

$$= 0.012$$

And the probability that the message is ham can be expressed as:

$$P(Ham|W1 \cap \neg W2 \cap \neg W3 \cap W4)$$

$$\propto P(W1|Ham)\ P(\neg W2|Ham)\ P(\neg W3|Ham)\ P(W4|Ham)\ P(Ham)$$

$$\propto (1/80) * (66/80) * (71/80) * (23/80) * (80/10)$$

$$= 0.002$$

Because 0.012/0.002 = 6, the message is six times more likely to be spam than ham. Reintroduce the denominator to convert these numbers into probabilities. That is, rescale the likelihood of each outcome by dividing it by the total likelihood across all possible outcomes. Thus, probability of spam is equal to the likelihood that the message is spam divided by the likelihood that the message is either spam or ham.

$$0.012/(0.012 + 0.002) = 0.857$$

Similarly, the probability of ham is equal to the likelihood that the message is ham divided by the likelihood that the message is either spam or ham:

$$0.002/(0.012 + 0.002) = 0.143$$

So, it can be concluded as the message is spam with 85.7 percent probability and ham with 14.3 percent probability.

The Naive Bayes classification algorithm can be summarized by the following formula. The probability of level L for class C, given the evidence provided by features F_1 through F_n, is equal to the product of the probabilities of each piece of evidence conditioned on the class level, the prior probability of the class level, and a scaling factor 1/Z, which converts the likelihood values into probabilities:

$$P(C_L|F_1, \ldots, F_n) = \frac{1}{Z} p(C_L) \prod_{i=1}^{n} p(F_i|C_L)$$

The Laplace estimator

In the above problem the likelihood of Groceries for the spam messages is 0/20. Assume that we have a new problem statement where the incoming email has all four terms: Explosive, Groceries, Money, and Unsubscribe. Using the Naive Bayes algorithm, the likelihood spam can be computed as (4/20) * (10/20) * (0/20) * (12/20) * (20/100) = 0. The likelihood of ham is: (1/80) * (14/80) * (8/80) * (23/80) * (80/100) = 0.00005. Therefore, the probability of spam is: 0/(0 + 0.00005) = 0. And the probability of ham is: 0.00005/(0 + 0. 0.00005) = 1. These results suggest that the message is spam with 0 percent probability and ham with 100 percent probability. But

this prediction doesn't make sense. This problem happened because the term Groceries had never previously appeared in a spam message. Consequently, $P(Spam|Groceries) = 0\%$.

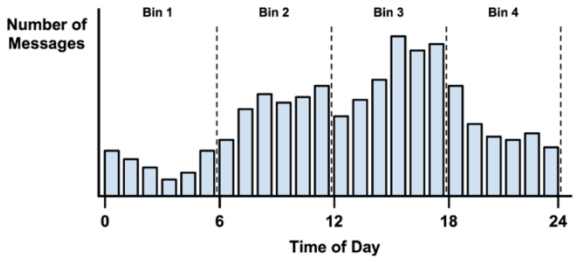
Because probabilities in the Naive Bayes formula are multiplied in a chain, this 0 percent value causes the posterior probability of spam to be zero, giving the word Groceries the ability to effectively nullify and overrule all of the other evidence. A solution to this problem involves using something called the **Laplace estimator**, which is named after the French mathematician Pierre-Simon Laplace. The Laplace estimator essentially adds a small number to each of the counts in the frequency table, which ensures that each feature has a nonzero probability of occurring with each class. Typically, the Laplace estimator is set to 1, which ensures that each class-feature combination is found in the data at least once.

Using a Laplace value of 1, we add one to each numerator in the likelihood function. The total number of 1 values must also be added to each conditional probability denominator. The likelihood of spam is therefore: (5/24) * (11/24) * (1/24) * (13/24) * (20/100) = 0.0004. The likelihood of ham is: (2/84) * (15/84) * (9/84) * (24/84) * (80/100) = 0.0001. Thus the probability of Spam is 0.0004/(0.0004+0.0001) = 0.8 and the probability of ham is 0.0001/(0.0004+0.0001) = 0.2. This means that the probability of spam is 80 percent, and the probability of ham is 20 percent.

Using numeric features with Naive Bayes

Because Naive Bayes uses frequency tables to learn the data, each feature must be categorical in order to create the combinations of class and feature values comprising of the matrix. Since numeric features do not have categories of values, the preceding algorithm does not work directly with numeric data.

One easy and effective solution is to discretize numeric features, which simply means that the numbers are put into categories known as bins. For this reason, discretization is also sometimes called binning. The most common method for discretization is to explore the data for natural categories or cut points in the distribution of data.



For example, assume that the spam dataset contains a feature that recorded the time of night or day the e-mail was sent, from 0 to 24 hours past midnight. The corresponding bin is given in the diagram. The diagram shows four natural bins of activity, as partitioned by the dashed lines indicating places where the numeric data are divided into levels of a new nominal feature, which could then be used with Naive Bayes.

Discretizing a numeric feature always results in a reduction of information as the feature's original granularity is reduced to a smaller number of categories. Too few bins can result in important trends being obscured. Too many bins can result in small counts in the Naive Bayes frequency table, which can increase the algorithm's sensitivity to noisy data.

Strengths

- Simple, fast, and very effective
- Does well with noisy and missing data
- Requires relatively few examples for training, but also works well with very large numbers of examples
- Easy to obtain the estimated probability for a prediction

Weaknesses

- Relies on an often-faulty assumption of equally important and independent features
- Not ideal for datasets with many numeric features
- Estimated probabilities are less reliable than the predicted classes