Cloud-Based Machine and Deep Learning

Machine Learning and AI Algorithm
Basic Concept of Artificial Intelligence (AI)

- AI has been a research field for at least 60 years ever since Alan Turing who suggested the very first “Thinking Machine” known as Turing Machines

- Intelligent machines must have two fundamental elements:
  - One is cognitive power such as 5 senses of cognition: Vision, hearing, taste, touch, and smell
  - Another is language skill that a human has to understand, express and comprehend things (Animals have none)

- Turing did not realize intelligent machines at his time, but he pioneered the key concept which we are now producing some smart machines with hardware, software and sensing devices
AI Pioneers: Marvin Minsky (MIT) and Terry Minograd (Stanford)

Three Things that must interact to achieve AI: Syntax (structure), Semantics (meaning), and Inference (reasoning). In fact, human intelligence also requires these three ingredients.
Evolution of Deep Learning from ML and The ANN Approach

- Artificial Neural Networks
- Deep Learning
- Supervised Learning
- Machine Learning (Unsupervised)
Machine Learning is a scientific discipline that explores the construction and study of algorithms that can learn from data.

Machine learning algorithms operate by building a model from executing example inputs and using that model to make predictions or decisions.

Machine Learning is a subfield of computer science stemming from research into artificial intelligence. It has strong ties to statistics and mathematical optimization.
Traditional Programming Model versus Machine Learning Model

Traditional Programming

Human Programmer → Program → Output

Machine Learning

Data → Learning Algorithm → Program → Output
Computer Vision, Audio Recognition, and Text Understanding
What is Computer Perception?

Input

E.g., SIFT, HoG, etc.

Object detection
- Image
- Vision features
- Detection

Audio classification
- Audio
- Audio features
- Speaker ID

NLP
- Text
- Text features
- Text classification, Machine translation, Information retrieval, etc.
Computer Vision Features
Audio Recognition Features

- Parser features
- NER/SRL
- Stemming

Anaphora

POS tagging

WordNet features
Nature Language Processing (NLP) Features
Machine Learning Algorithms

Supervised learning with Training Data

Unsupervised without Training Samples

Semi-Supervised with Partial Samples
12 Machine Learning Methods

(a) Regression Algorithm  
(b) Instance-based Algorithm  
(c) Regularization Algorithm  
(d) Decision Tree Algorithm

(e) Bayesian Algorithms  
(f) Clustering Algorithms  
(g) Association Rule Learning Algorithms  
(h) Artificial Neural Network Algorithms

(i) Deep Learning Algorithms  
(j) Dimensional Reduction Algorithms  
(k) Support Vector Machine Algorithms  
(l) Ensemble Algorithms
Machine Learning Approaches (1)

1. Decision Tree Learning: Using multi-way tree to make categorical decisions

2. Association Rule Learning: Discovering interesting relations between variables in large databases

3. Artificial Neural Networks (ANN): Learning algorithm that is inspired by the structure and functional aspects of biological neural networks

4. Inductive Logic Programming (ILP): Using logical programming to represent input data, background knowledge, and hypotheses
5. **Support Vector Machines (SVM):** Using a set of related supervised learning methods for classification and regression

6. **Clustering Analysis:** Grouping sample data into clusters with similar properties or some predefined criteria

7. **Bayesian Networks:** A belief network or a probabilistic DAG model that represent a set of random variables and their conditional independencies

8. **Reinforcement Learning:** Based on how an agent ought to take actions in an environment to maximize some notion of long-term reward
9. Representative Learning: Based on preserving the input information and transforming it as a pre-processing process for other classification algorithms

10. Similarity and Metric Learning: To learn from a similarity function or distance metric function

11. Sparse Dictionary Learning: Each datum is represented as a linear combination of basic functions, and the coefficients are assumed sparse

12. Genetic Algorithms (GA): A research heuristic that mimics the process of natural selection and uses methods such as mutation and crossover to generate genotype towards making better decision
Machine Learning and Cloud Analytics

- Machine Learning is a scientific discipline that explores the construction and study of algorithms that can learn from data.

- Machine learning algorithms operate by building a model from executing example inputs and using that model to make predictions or decisions.

- Machine Learning is a subfield of computer science stemming from research into artificial intelligence. It has strong ties to statistics and mathematical optimization.
Data Mining and Machine learning

Machine Learning

Supervised Learning
- Regression Model
- Decision Tree
- Bayesian Networks
- KNN, SVM, ANN

Unsupervised Learning
- K-means Clustering
- Hierarchical Clustering
- DBSCAN

Others Learning
- Reinforcement Learning
- Transfer Learning
- Deep Learning

Data Mining

Association Analysis
- Apriori Algorithm
- FP-growth Algorithm

Classification
- Decision Tree
- Bayesian Networks
- KNN, SVM, ANN

Cluster Analysis
- K-means Clustering
- Hierarchical Clustering
- DBSCAN

Big Data Source

Big Data Application
## Data Mining Software Tool

<table>
<thead>
<tr>
<th>Data Mining Software Tool</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rapid-I RapidMiner/RapidAnalytics free edition (737), 30.9% alone</td>
<td>39.2%</td>
</tr>
<tr>
<td>R (704), 6.5% alone</td>
<td>37.4%</td>
</tr>
<tr>
<td>Excel (527), 0.9% alone</td>
<td>28.0%</td>
</tr>
<tr>
<td>Weka/Pentaho (269), 5.6% alone</td>
<td>14.3%</td>
</tr>
<tr>
<td>Python with any of numpy/scipy/pandas/iPython packages (250), 0% alone</td>
<td>13.3%</td>
</tr>
<tr>
<td>Rapid-I RapidAnalytics/RapidMiner Commercial Edition (225), 52.4% alone</td>
<td>12.0%</td>
</tr>
<tr>
<td>SAS (202), 2.0% alone</td>
<td>10.7%</td>
</tr>
<tr>
<td>MATLAB (186), 1.6% alone</td>
<td>9.9%</td>
</tr>
<tr>
<td>StatSoft Statistica (170), 45.9% alone</td>
<td>9.0%</td>
</tr>
<tr>
<td>IBM SPSS Statistics (164), 1.8% alone</td>
<td>8.7%</td>
</tr>
<tr>
<td>Microsoft SQL Server (131), 1.5% alone</td>
<td>7.0%</td>
</tr>
</tbody>
</table>
Histograms for Outlier Detection
Model Fitting in Machine Learning

Source data including some training and testing data

Data storage

Data filtering, mining and feature selection

Compute engine

Data cleaning, training and preprocessing

Model selection and fitting (clustering shown)

Training feedback to optimize the model

Cloud platform

Decision making or predictive analytics

Training or testing results

Output results

Time

Machine Learning Pipeline Engine
We have covered two Machine Learning Paradigms so far: Supervised Decision Tree and Unsupervised Clustering Method.
## Consumer Lifetime Value (CLV)

<table>
<thead>
<tr>
<th>Month $t$</th>
<th>Revenue in Month $t$ ($R_t$)</th>
<th>Cost in Month $t$ ($C_t$)</th>
<th>Survival Probability in Month $t$ ($s_t$)</th>
<th>$(R_t - C_t) \times \frac{s_t}{(1 + d)^t}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>150</td>
<td>5</td>
<td>0.94</td>
<td>135.22</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>10</td>
<td>0.92</td>
<td>82.80</td>
</tr>
<tr>
<td>3</td>
<td>120</td>
<td>5</td>
<td>0.88</td>
<td>101.20</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>0</td>
<td>0.84</td>
<td>84.00</td>
</tr>
<tr>
<td>5</td>
<td>130</td>
<td>10</td>
<td>0.82</td>
<td>98.40</td>
</tr>
<tr>
<td>6</td>
<td>140</td>
<td>5</td>
<td>0.74</td>
<td>99.90</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>15</td>
<td>0.7</td>
<td>45.50</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>10</td>
<td>0.68</td>
<td>61.20</td>
</tr>
<tr>
<td>9</td>
<td>120</td>
<td>10</td>
<td>0.66</td>
<td>72.60</td>
</tr>
<tr>
<td>10</td>
<td>90</td>
<td>20</td>
<td>0.6</td>
<td>42.00</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>0</td>
<td>0.55</td>
<td>55.00</td>
</tr>
<tr>
<td>12</td>
<td>130</td>
<td>10</td>
<td>0.5</td>
<td>60.00</td>
</tr>
<tr>
<td><strong>Yearly WACC</strong></td>
<td>10%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Monthly WACC</strong></td>
<td>1%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
CLV = \sum_{i=1}^{n} \frac{(R_t - C_t)s_t}{(1 + d)^t}
\]

CLV = 937.82
Visualize How Customers Move from One Delinquency To Another During A Specific Time Frame

**Figure 3.1** Roll Rate Analysis

*Source: N. Siddiqi, Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring (Hoboken, NJ: John Wiley & Sons, 2005).*
LINEAR REGRESSION

Linear regression is a baseline modeling technique to model a continuous target variable. For example, in a CLV modeling context, a linear regression model can be defined to model CLV in terms of the RFM (recency, frequency, monetary value) predictors as follows:

$$CLV = \beta_0 + \beta_1 R + \beta_2 F + \beta_3 M$$

The parameters $\beta$ are then typically estimated using ordinary least squares (OLS) to minimize the sum of squared errors. As part of the estimation, one then also obtains standard errors, p-values indicating variable importance (remember important variables get low p-values), and confidence intervals. A key advantage of linear regression is that it is simple and usually works very well.

Note that more sophisticated variants have been suggested in the literature (e.g., ridge regression, lasso regression, time series models [ARIMA, VAR, GARCH], multivariate adaptive regression splines [MARS]).
LOGISTIC REGRESSION

Consider a classification data set for response modeling as depicted in Table 3.2.

Table 3.2 Example Classification Data Set

<table>
<thead>
<tr>
<th>Customer</th>
<th>Age</th>
<th>Income</th>
<th>Gender</th>
<th>Response</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>30</td>
<td>1,200</td>
<td>M</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>Sarah</td>
<td>25</td>
<td>800</td>
<td>F</td>
<td>Yes</td>
<td>1</td>
</tr>
<tr>
<td>Sophie</td>
<td>52</td>
<td>2,200</td>
<td>F</td>
<td>Yes</td>
<td>1</td>
</tr>
<tr>
<td>David</td>
<td>48</td>
<td>2,000</td>
<td>M</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>Peter</td>
<td>34</td>
<td>1,800</td>
<td>M</td>
<td>Yes</td>
<td>1</td>
</tr>
</tbody>
</table>

When modeling the response using linear regression, one gets:

\[ Y = \beta_0 + \beta_1 \text{Age} + \beta_2 \text{Income} + \beta_3 \text{Gender} \]

When estimating this using OLS, two key problems arise:

1. The errors/target are not normally distributed but follow a Bernoulli distribution.
2. There is no guarantee that the target is between 0 and 1, which would be handy because it can then be interpreted as a probability.

Consider now the following bounding function:

\[ f(z) = \frac{1}{1 + e^{-z}} \]

which can be seen in Figure 3.2.
Logistic Regression Method

Figure 3.2 Bounding Function for Logistic Regression

For every possible value of \( z \), the outcome is always between 0 and 1. Hence, by combining the linear regression with the bounding function, we get the following logistic regression model:

\[
P(response = yes | age, income, gender) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 age + \beta_2 income + \beta_3 gender)}}
\]

The outcome of the above model is always bounded between 0 and 1, no matter what values of age, income, and gender are being used, and can as such be interpreted as a probability.

The general formulation of the logistic regression model then becomes:

\[
P(Y = 1 | X_1, \ldots, X_n) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \ldots + \beta_N X_N)}}
\]

or, alternately,

\[
P(Y = 0 | X_1, \ldots, X_n) = 1 - P(Y = 1 | X_1, \ldots, X_n)
\]

\[
= 1 - \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \ldots + \beta_N X_N)}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \ldots + \beta_N X_N)}}
\]

Hence, both \( P(Y = 1 | X_1, \ldots, X_N) \) and \( P(Y = 0 | X_1, \ldots, X_N) \) are bounded between 0 and 1.
Reformulating in terms of the odds, the model becomes:

\[
\frac{P(Y = 1 \mid X_1, \ldots, X_N)}{P(Y = 0 \mid X_1, \ldots, X_N)} = e^{(\beta_0 + \beta_1 X_1 + \ldots + \beta_N X_N)}
\]

or, in terms of log odds (logit),

\[
\ln \left( \frac{P(Y = 1 \mid X_1, \ldots, X_N)}{P(Y = 0 \mid X_1, \ldots, X_N)} \right) = \beta_0 + \beta_1 X_1 + \ldots + \beta_N X_N
\]

The $\beta_i$ parameters of a logistic regression model are then estimated by optimizing a maximum likelihood function. Just as with linear regression, the optimization comes with standard errors, p-values for variable screening and confidence intervals.

Since logistic regression is linear in the log odds (logit), it basically estimates a linear decision boundary to separate both classes. This is illustrated in Figure 3.3.
**Figure 3.8** Using a Validation Set to Stop Growing a Decision Tree
### Table 4.1 Examples of Descriptive Analytics

<table>
<thead>
<tr>
<th>Type of Descriptive Analytics</th>
<th>Explanation</th>
<th>Example</th>
</tr>
</thead>
</table>
| Association rules             | Detect frequently occurring patterns between items | Detecting what products are frequently purchased together in a supermarket context  
Detecting what words frequently co-occur in a text document  
Detecting what elective courses are frequently chosen together in a university setting |
| Sequence rules                | Detect sequences of events                 | Detecting sequences of purchase behavior in a supermarket context  
Detecting sequences of web page visits in a web mining context  
Detecting sequences of words in a text document |
| Clustering                    | Detect homogeneous segments of observations | Differentiate between brands in a marketing portfolio  
Segment customer population for targeted marketing |
Basic Concepts of Regression Analysis (1)

- **Regression analysis** performs a sequence of parametric or non-parametric estimations. The method finds the causal relationship between the input and output variables. The estimation function can be determined by experience using prior knowledge or visual observation of the data.

- **Regression analysis** is aimed to understand how the typical values of the output variables change, while the input variables are held unchanged. Thus regression analysis estimates the average value of the dependent variable when the independent variables are fixed.
Most regression methods are parametric in nature and have a finite dimension in the analysis space. We will not deal with nonparametric regression analysis, which may be infinite-dimensional.

The accuracy or the performance of regression methods depends on the quality of the dataset used. In a way, regression offers an estimation of continuous response variables, as opposed to the discrete decision values used in classification.
Three Cases To be Modeled in A Regression Process

- When \( N < k \), most classical regression analysis methods can be applied. Since the defining equation is underdetermined, there are not enough data to recover the unknown parameters \( \beta \).

- When \( N = k \) and the function \( f \) is linear, the equations \( Y = f(X, \beta) \) can be solved exactly without approximation, because there are \( N \) equations to solve \( N \) components in \( \beta \). The solution is unique as long as the \( X \) components are linearly independent. If \( f \) is nonlinear, many solutions may exist or no solution at all.

- In general, we have the situation that \( N > k \) data points. This implies that there is enough information in the data that can estimate a unique value for \( \beta \) under an overdetermined situation.

Consider an example to toss a small ball in the air. We measure its heights of ascent \( h \) at various time instant \( t \). The relationship is modeled as

\[
h = \beta_1 t + \beta_2 t^2 + \varepsilon
\]

where \( \beta_1 \) determines the initial velocity of the ball, \( \beta_2 \) is proportional to the standard gravity, and \( \varepsilon \) is due to the measurement error.

Here, linear regression is used to estimate the values of \( \beta_1 \) and \( \beta_2 \) from the measured data. This model is non-linear in time variable \( t \), but it is linear with respect to the unknown parameters \( \beta_1 \) and \( \beta_2 \).
Consider a set of data points in a 2-dimensional sample space, \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\). If they can be approximated by a straight line, then we obtain the following linear regression expression.

\[ y = ax + b + \varepsilon \]  

(7.2)

where \(x\) stands for an explanatory variable, \(y\) is a continuous variable in the real number range, \(a\) and \(b\) are corresponding coefficients, and \(\varepsilon\) is a random error, which follows an independent normal distribution. One needs to figure out the expectation by using a linear regression expression \(y = ax + b\) illustrated in Fig.7.3.
Logistic Regression Method: This is a linear regression analysis model extended to a broader application for prediction and classification, it is commonly used in fields such as data mining, automatic diagnosis for diseases and economical prediction.

The logistic model may only be used to solve problem of dichotomy. As for logistic classification, the principle is to conduct classification to sample data with a logistic function, known as a sigmoid function defined by:

$$f(x) = \frac{1}{1 + e^{-x}}$$

The input domain of the sigmoid function is in the range (0, 1). In this sense, the sigmoid function is a probability density function for the sample data shown in Fig,7.4.
The basic idea of logistic regression is to consider vector $x$ with $m$ independent input variables. Each dimension of $x$ stands for one attribute (feature) of the sample data (training data). In logistic regression, multiple features of the sample data are combined into one feature by using linear function.

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m$$

We need to figure out the probability of the feature with designated data and apply the sigmoid function to act on that feature. We obtain the logistic regression as plotted in Fig. 7.5.

$$P(Y = 1 \mid x) = \pi(x) = \frac{1}{1 + e^{-z}}$$

$$\begin{align*}
x & \in 1, \text{ if } P(Y = 1 \mid x) > 0.5 \\
x & \in 0, \text{ if } P(Y = 0 \mid x) < 0.5
\end{align*}$$
The Sigmoid Function

Figure 7.5: Fundamental concept of using logistic regression for classification purpose.
An Example of Using Logistic Regression for Classification

LOGISTIC REGRESSION

Consider a classification data set for response modeling as depicted in Table 3.2.

Table 3.2 Example Classification Data Set

<table>
<thead>
<tr>
<th>Customer</th>
<th>Age</th>
<th>Income</th>
<th>Gender</th>
<th>...</th>
<th>Response</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>30</td>
<td>1,200</td>
<td>M</td>
<td></td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>Sarah</td>
<td>25</td>
<td>800</td>
<td>F</td>
<td></td>
<td>Yes</td>
<td>1</td>
</tr>
<tr>
<td>Sophie</td>
<td>52</td>
<td>2,200</td>
<td>F</td>
<td></td>
<td>Yes</td>
<td>1</td>
</tr>
<tr>
<td>David</td>
<td>48</td>
<td>2,000</td>
<td>M</td>
<td></td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>Peter</td>
<td>34</td>
<td>1,800</td>
<td>M</td>
<td></td>
<td>Yes</td>
<td>1</td>
</tr>
</tbody>
</table>

When modeling the response using linear regression, one gets:

\[ Y = \beta_0 + \beta_1 Age + \beta_2 Income + \beta_3 Gender \]
Logistic Regression Method

Figure 3.2 Bounding Function for Logistic Regression

For every possible value of $z$, the outcome is always between 0 and 1. Hence, by combining the linear regression with the bounding function, we get the following logistic regression model:

$$P(response = yes|age, income, gender) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 age + \beta_2 income + \beta_3 gender)}}$$

The outcome of the above model is always bounded between 0 and 1, no matter what values of age, income, and gender are being used, and can as such be interpreted as a probability.
Four Clustering Techniques

Agglomerative and Divisive work in opposite directions over subdivided clustering steps, K-means apply initial seeds as centroids and eventually converge to K centroids. SOM: Self-Organizing Maps are feedforward neural networks
kNN and k-Mean Clustering

k-Nearest Neighbor (kNN) Clustering:

- This is a kind of lazy learning method or a type of instance-based learning which is the simplest to implement. The objective function is only approximated locally with a deferred classification.

- The idea is to consider the input data elements as among the $k$ closest training examples in the feature space. The output depends on whether kNN is used for classification or for regression, as defined below:

  - **For kNN classification**, an object is classified by a majority vote of its neighbors, meaning the element being classified as a member by the most common among its $k$ nearest neighbors

  - **For kNN regression**, the output is the property value for the data object, which is the average of the values of its $k$ nearest neighbors. This means that the data object is weighted by the nearer neighbors
Divisive vs. Agglomerative Hierarchical Clustering

Figure 4.4 Divisive versus Agglomerative Hierarchical Clustering
Concept of k-mean Clustering

(a) Choose 3 initial means for $k = 3$
(b) 3 initial clusters with the nearest mean
(c) Centroids in 3 clusters become new means
(d) After repeating steps 2 and 3 to converge

Figure 7.8: Four steps to generate 3 clusters out of 15 data elements (Reprint with permission from Wikipedia)
Cluster Analysis

Original unclustered data vs Clustered data

Clustered data

$\xi_2$

$\xi_1$
Example 6.6: Using k-mean Clustering to Classify the Iris Flowers
(contributed by Wenhao Zhang, USC 2016)

Consider solving an iris flower classification problem with $k$-mean clustering for $k = 3$ clusters. Given a data set of 150 unlabeled data points on Iris flowers. These flowers are classified into three clusters, namely *Iris Setosa, Iris Versicolour,* and *Iris Virginica.*

The complete flower data set, denoted as “iris.csv” is from UC Irvine Machine Learning Repository. There are 4 features in this dataset: (1) *sepal length in cm,* (2) *sepal width in cm,* (3) *petal length in cm,* and (4) *petal width in cm.*

To simplify the final display of the clustering results, we consider the data points with only two most important features {3, 4}. Identify the clustering centers (centroids) in successive steps, and also draw a 2-D Euclidean space diagram (x-axis: petal.length; y-axis: petal.width) to show the final clustering results.
inputData <- read.csv("./iris.csv") # Read the input dataset
initialCenters <- matrix(c(2, 4, 6, 0.8, 1.6, 2.5), 3, 2) # Select initial means of 3 clusters.
results <- kmeans(inputData, initialCenters, iter.max = 1, trace=TRUE) # First iteration
firstCenters <- results$centers # Save the clustering centers after one iteration
results <- kmeans(inputData, initialCenters, iter.max = 2, trace=TRUE) # Second iteration
plot(inputData, pch = results$cluster) # Plot the partitioning results after it converges
points(initialCenters, pch = 16, cex = 2) # Plot the initial centers
points(firstCenters, pch = 15, cex = 2) # Plot the centers after 1st iteration
points(results$centers, pch = 17, cex = 2) # Plot the centers after 2nd iteration, convergence
• Consider each attribute and class label as random variables

• Given a record with attributes $(A_1, A_2, \ldots, A_n)$
  – Goal is to predict any class denoted as $C$
  – Specifically, we want to find the value of $C$ that maximizes $P(C| A_1, A_2, \ldots, A_n)$

• Can we estimate $P(C| A_1, A_2, \ldots, A_n)$ directly from data?
Bayes Classifier (2)

- A probabilistic framework for solving the classification problems

- Conditional Probabilities:
  \[ P(C \mid A) = \frac{P(A, C)}{P(A)} \]
  \[ P(A \mid C) = \frac{P(A, C)}{P(C)} \]

- Bayes theorem:
  \[ P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)} \]
Bayesian Classifiers (3)

• Compute the posterior probability $P(C | A_1, A_2, \ldots, A_n)$ for all values of $C$ using the Bayes theorem

$$P(C | A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n | C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

– Choose the value of $C$ that maximizes $P(C | A_1, A_2, \ldots, A_n)$
– Equivalent to choosing value of $C$ that maximizes $P(A_1, A_2, \ldots, A_n | C) P(C)$
– How to estimate $P(A_1, A_2, \ldots, A_n | C)$?
• Assume statistical independence among all n attributes $A_i$ when class $C_j$ is given:

$$P(A_1, A_2, ..., A_n | C_j) = P(A_1 | C_j) P(A_2 | C_j) ... P(A_n | C_j)$$

– Can estimate $P(A_i | C_j)$ for all $A_i$ and $C_j$ from the sample data points with labels

– New testing data point is classified to class $C_j$, if $P(C_j) P(A_1 | C_j) P(A_2 | C_j) ... P(A_n | C_j)$ is maximal
Estimate Probabilities from Data (5)

- **Class:** \( P(C) = \frac{N_c}{N} \)
  
  \( P(\text{No}) = 7/10, \quad P(\text{Yes}) = 3/10 \)

- **For discrete attributes:**
  
  \[ P(A_i \mid C_k) = \frac{|A_{ik}|}{N_c} \]

  where \(|A_{ik}|\) is the number of \(k\) instances having attribute \(A_i\) belonging to class \(C_k\).
Naïve Bayes Classifier (6)

• If one of the conditional probabilities is zero, then the entire expression becomes zero.

Original: $P(A_i | C) = \frac{N_{ic}}{N_c}$

Laplace: $P(A_i | C) = \frac{N_{ic} + 1}{N_c + c}$

$m$-estimate: $P(A_i | C) = \frac{N_{ic} + mp}{N_c + m}$

- $c$: Number of classes
- $p$: Prior probability
- $m$: Parameter
# Example of Bayesian Classifier

<table>
<thead>
<tr>
<th>Name</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Have Legs</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>python</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>non-mammals</td>
</tr>
<tr>
<td>salmon</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>non-mammals</td>
</tr>
<tr>
<td>whale</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>frog</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>komodo</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>bat</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>pigeon</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>cat</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>leopard shark</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>non-mammals</td>
</tr>
<tr>
<td>turtle</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>penguin</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>porcupine</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>eel</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>non-mammals</td>
</tr>
<tr>
<td>salamander</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>gila monster</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>platypus</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>owl</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>dolphin</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>eagle</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
</tbody>
</table>
Consider an unlabeled testing data item characterized by an attribute vector: \( A^* = < A_1, A_2, A_3, A_4> = <\text{yes, no, yes, no}> \).

<table>
<thead>
<tr>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Have Legs</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>?</td>
</tr>
</tbody>
</table>

**Table 6.6. Pre-Test Attribute Probability for Sample Data in Table 6.5**

<table>
<thead>
<tr>
<th>Attributes Statistics</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Have Legs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Counts</td>
<td>M</td>
<td>6</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>1</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>Probability</td>
<td>M</td>
<td>6/7</td>
<td>1/7</td>
<td>6/7</td>
</tr>
</tbody>
</table>
Example of Naïve Bayes Classifier (8)

A: attributes;  M: mammals;  N: non-mammals

\[
P(A \mid M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06
\]
\[
P(A \mid N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042
\]
\[
P(A \mid M)P(M) = 0.06 \times \frac{7}{20} = 0.021
\]
\[
P(A \mid N)P(N) = 0.004 \times \frac{13}{20} = 0.0027
\]

<table>
<thead>
<tr>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Have Legs</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>?</td>
</tr>
</tbody>
</table>

\[
P(A \mid M)P(M) = 0.06 \times (7/20) = 0.021 > P(A \mid N)P(N) = 0.0042 \times (13/20) = 0.0027
\]

Predicted Outcome: Mammals
Now, let us analyze the accuracy of using the Baysian classifier by testing 4 creatures using the above method. We list the following results as listed below. We obtain the posterior probabilities $P(M \mid A_1, A_2, A_3, A_4)$ and $P(N \mid A_1, A_2, A_3, A_4)$ for each of the 4 testing animals. Choose the class with highest probability as the predicted class.

<table>
<thead>
<tr>
<th>Animal Name</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Have Legs</th>
<th>Predicted Class</th>
<th>Actual Class</th>
<th>Prediction Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>M</td>
<td>M</td>
<td>TP</td>
</tr>
<tr>
<td>Monostream</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>N</td>
<td>M</td>
<td>FN</td>
</tr>
<tr>
<td>Alligator</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>N</td>
<td>N</td>
<td>TN</td>
</tr>
<tr>
<td>Horse</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>M</td>
<td>M</td>
<td>TP</td>
</tr>
</tbody>
</table>

Comparing the predicted results with the actual classes, we discover 4 possible prediction statuses at the rightmost column. The TP (true positive) refers to a true case correctly predicted, TN (True Negative) for a true case incorrectly predicted, FP (False Positive) means a false case correctly predicted and FN (False negative) for the false case that is incorrectly predicted. Based on the comparison results, we have the following performance results: $TP = 2/4 = 0.5$, $TN =1/4=0.25$, $FP=0$, and $FN=1/4=0.25$. Then, we use two performance metrics to assess the accuracy of the Baysian classifier.

Prediction accuracy $= (TP+TN) / (TP+TN+FP+FN) = 0.75$

Prediction error $= (FP+FN)/ (TP+TN+FP+FN) = 0.25$
## Prediction Accuracy Analysis

Table 7.6 Predicted Results of 4 Animals Compared with Their Actual Classes

<table>
<thead>
<tr>
<th>Animal Name</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Have Legs</th>
<th>Predicted Class</th>
<th>Actual Class</th>
<th>Prediction Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>M</td>
<td>M</td>
<td>TP</td>
</tr>
<tr>
<td>Monostream</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>N</td>
<td>M</td>
<td>FN</td>
</tr>
<tr>
<td>Alligator</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>N</td>
<td>N</td>
<td>TN</td>
</tr>
<tr>
<td>Horse</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>M</td>
<td>M</td>
<td>TP</td>
</tr>
</tbody>
</table>
Dimensionality Reduction

• **Purpose:**
  – Avoid curse of dimensionality
  – Reduce amount of time and memory required by data mining algorithms
  – Allow data to be more easily visualized
  – Help eliminate irrelevant features or reduce noise

• **Techniques**
  – Principle Component Analysis
  – Singular Value Decomposition
  – Others: supervised and non-linear techniques
Support Vector Machines (1)
Extending linear programming with multiple decision boundaries (hyperplanes) to separate the classes

Two key shortcomings of neural networks are the fact that the objective function is nonconvex (and hence may have multiple local minima) and the effort that is needed to tune the number of hidden neurons. Support vector machines (SVMs) deal with both of these issues.\textsuperscript{21}

The origins of classification SVMs date back to the early dates of linear programming.\textsuperscript{22} Consider the following linear program (LP) for classification:

$$\min e_1 + e_2 + \ldots + e_{ng} + e_{nb}$$

subject to

\begin{align*}
w_1 x_{i1} + w_2 x_{i2} + \ldots + w_n x_{in} &\geq c - e_i, 1 \leq i \leq n_g \\
w_1 x_{i1} + w_2 x_{i2} + \ldots + w_n x_{in} &\leq c + e_i, n_g + 1 \leq i \leq n_g + n_b \\
e_i &\geq 0
\end{align*}

The LP assigns the good customers a score above the cut-off value $c$, and the bad customers a score below $c$. $n_g$ and $n_b$ represent the number of goods and bads, respectively. The error variables $e_i$ are needed to be able to solve the program because perfect separation will typically not be possible. Linear programming has been very popular in the early days of credit scoring. One of its benefits is that it is easy to include domain or business knowledge by adding extra constraints to the model.

A key problem with linear programming is that it can estimate multiple optimal decision boundaries, as illustrated in Figure 3.19, for a perfectly linearly separable case.
Support Vector Machines (2)

Figure 3.19 Multiple Separating Hyperplanes

SVMs add an extra objective to the analysis. Consider, for example, the situation depicted in Figure 3.20. It has two hyperplanes sitting at the edges of both classes and a hyperplane in between, which will serve as the classification boundary. The perpendicular distance from the first hyperplane $H_1$ to the origin equals $|b-1|/||w||$, whereby $||w||$ represents the Euclidean norm of $w$ calculated as $||w|| = \sqrt{w_1^2 + w_2^2}$. Likewise, the perpendicular distance from $H_2$ to the origin equals $|b+1|/||w||$. Hence, the margin between both hyperplanes equals $2/||w||$. SVMs will now aim at maximizing this margin to pull both classes as far apart as
Support Vector Machines (3)

- Find a linear hyperplane (decision boundary) that separates the data

- Find a hyperplane to maximize the margin $\Rightarrow$ $B_1$ is better than $B_2$
Support Vector Machines (4)

\[ \vec{w} \cdot \vec{x} + b = 0 \]

\[ \vec{w} \cdot \vec{x} + b = -1 \]

\[ \vec{w} \cdot \vec{x} + b = +1 \]

Margin = \( \frac{2}{\| \vec{w} \|^2} \)

\[
f(\vec{x}) = \begin{cases} 
1 & \text{if } \vec{w} \cdot \vec{x} + b \geq 1 \\
-1 & \text{if } \vec{w} \cdot \vec{x} + b \leq -1 
\end{cases}
\]
Support Vector Machines (5)

- We want to maximize:
  \[ \text{Margin} = \frac{2}{\| \vec{w} \|^2} \]
  Which is equivalent to minimizing:
  \[ \ell(w) = \frac{\| \vec{w} \|^2}{2} \]
  - But subjected to the following constraints:

- This is a constrained optimization problem
  - Numerical approach to solving the problem.
    (e.g., Quadratic programming)

\[ f(\vec{x}_i) = \begin{cases} 
  1 & \text{if } \vec{w} \cdot \vec{x}_i + b \geq 1 \\
  -1 & \text{if } \vec{w} \cdot \vec{x}_i + b \leq -1 
\end{cases} \]
What if the problem is not linearly separable?

Introduce slack variables to minimize:

\[
L(w) = \frac{||w||^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k\right)
\]

Subject to:

\[
f(x_i) = \begin{cases} 
1 & \text{if } \langle w, \tilde{x}_i \rangle + b \geq 1 - \xi_i \\
-1 & \text{if } \langle w, \tilde{x}_i \rangle + b \leq -1 + \xi_i 
\end{cases}
\]
Feature Screening and Dimension Reduction (1)

- Revealing the correlation between features, one can cut off some features.
- Such feature screening may reduce either the overfitting or even the under-fitting effects, depending on feature distributions.
- Those features with limited representations in the data pace could be eliminated.
Feature Screening and Dimension Reduction (2)

- The method is known as feature screening or dimension reduction
- Association analysis or correlation analysis may help eliminate some weak dimensions
- Principal Component Analysis goes the extreme to concentrate on the key features based on a spectrum analysis of the harmonics.
Predictive Model Evaluation

- Larger training data set could produce better prediction model with higher accuracy
- Test data results can feedback to the model trainer to improve or extend the model for better accuracy
Analytics Model Performance Metrics

Figure 3.30 Calculating Predictions Using a Cut-Off
Confusion Matrix: Four Basic Metrics

<table>
<thead>
<tr>
<th>Predicted status</th>
<th>Positive (churn)</th>
<th>True positive (John)</th>
<th>False positive (Sophie)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative (no churn)</td>
<td>False negative (David)</td>
<td>True negative (Emma, Bob)</td>
<td></td>
</tr>
</tbody>
</table>

Based upon this matrix, one can now calculate the following performance measures:

- Classification accuracy = \(\frac{TP + TN}{TP + FP + FN + TN} = 3/5\)
- Classification error = \(\frac{FP + FN}{TP + FP + FN + TN} = 2/5\)
- Sensitivity = \(\frac{TP}{TP + FN} = 1/2\)
- Specificity = \(\frac{TN}{FP + TN} = 2/3\)
Basic Concepts of Machine Learning Performance

- Machine Learning Performance Metrics
  - Quality of training, datasets, and training time
  - Accuracy of forecast, prediction, and classification results
  - Implementation, update, and maintenance costs

- Sampling Scores vs. Cross Validation Scores

- Model Fitting Modes:
  - Perfect Fitting (Sampling score matches with testing scores)
  - Overfitting (Model over biased by sampling data)
  - Underfitting (Poor choice of sampling data)
Figure 6.19: The training score and cross validation score match nicely in a well-fitting machine learning model created.
Selecting Data Sets or Training Samples

- **Common Datasets**: Dividing the original data set into two parts with equal characteristics and distributions in the training set and testing set. This subdivision should result in a model performance to avoid either the over-fitting or the under-fitting problems.

- **Cross Validation**: Dividing the original data set into k parts and selecting a part in turn as a test set while the remaining as the training set. This demands k validation testing runs. This model performance shows the average accuracy of the model over many subdivided test sets.

- **Bootstrap Cycle**: Randomly sampling with replacement of some data elements repeatedly in different training samples. Let the sampled data be the training set, while the remaining as the test sets. Repeat these sampling cycle k times. The may end up with a weighted mean performance of all test sets.
Selecting Algorithms for General Datasets

Given a dataset from a known application domain. The following procedure shows how to select the proper machine learning algorithm, based on the dataset characteristics and performance requirement. Consider six ML categories: Decision Trees, Regression, Clustering, Bayesian, SVM, and ANN’s.

In general, the following options can be considered in solving the under-fitting problems. These methods appeal in particular, to improve the classification problem, since the model performance is so sensitive to the data sets applied. We consider three options to select the data sets.
Over-Fitting in ML Models

- This is the case that the training score is very high, but the cross-validation score is very low for testing datasets applied

- As shown in Fig. 7.28, the two scores are separated far apart from each other. This status implies that the model fits the training set very closely

- Overfitting model has ignored the noise margins in the validation data set. In other words, the training set is heavily biased on a particular training data set

- This sample data set stays far away from common data distribution or characteristics in general applications. In this case, the overfitting model simply cannot model the testing data accurately
Figure 6.20  The over-fitting case when creating a learning model using the linear-SVC algorithm with a small data set up to 160 samples
Enlarge the Data Set Size To Cover More Data Points and Feature Patterns

- Enlarging the training data set will make it more representative to catch the variety, veracity and volume features
- The noise effects will be reduced to results in high biases
- Manual labeling is added to separate some artificial sample space
- Sample data set can be also transformed to balance the distribution in different feature dimensions
Methods To Reduce Overfitting

• Lean network architecture using just 3 convolutional layers and 2 fully connected layers considering the size of the dataset and labels involved (8 age classes and 2 gender classes)

• **Dropout learning**: Randomly set the output value of network neurons to 0 with a dropout ratio of 0.5 (50% chance)

• **Weight decay**: Used to keep the magnitude of weights close to zero

• **Data Augmentation**: Took random crop of 227x227 from image of 256x256 and randomly mirrored it in each forward-backward training pass

All these measures help in keeping the number of free parameters in the network low reducing complexity and thus over-fitting
Under-Fitting Models

- This is the case when the model produced by a given training set ends up a very low score performance, which is far below the user’s expectation.
- Under-fitting phenomenon implies that a poor training set was chosen.
- The trained model cannot perform well at all on real testing data sets.
- Therefore, the model so obtained is totally unacceptable to users.
Figure 6.21  Reducing the model overfitting effects by enlarging the training set to 350 samples
Underfitting due to Linear–SVC Algorithm Used
Methods To Avoid Model Under-Fitting Effects

- Under-fitting is often resulted from poor samples collection
- Mismatch between the machine learning algorithm and problem domain environment
- Consider the case of SVM (Simple Vector Machine) model for solving a classification problem. If the problem is not linearly separable, one must switch to the nonlinear SVM model.
Loss functions express the discrepancy between the trained model prediction and the actual problem instances. The loss function reveals the effects of losing the expected performance of an ML algorithm. We consider below 5 loss functions:

- **Zero-one loss function**: This policy offers a very sharp division between success and failure. The 0-1 loss function just counts the number of miss-predictions in the classification problems. It is not practical in real-life applications.

- **Hinge loss function**: This is often used in SVM (Support Vector Machines) applications for its relative strength to reflect the unusual sensitivity to noise effects. This function is not supported by the probabilistic distributions.
Figure 6.24: Effects of using different loss functions in machine learning model selection.
Machine Learning Model Selection Options: (2)

- **Log loss function**: This reflects a probabilistic distribution
  - The log-loss function is suitable for multi-classification problems
  - The shortcoming lies in lower sensitivity to noises
- **Exponential loss function**: This has been applied in AdaBoost, very sensitive to crowd and noises, but effective to deal with boosting algorithms
Machine Learning Model Selection Options (3)

- Perceptron loss function:
  - This is a variation from the hinge loss, which imposes a heavy penalty to misjudgement of the boundary points
  - The perceptron loss is satisfied with accurate classification from sample data.
  - The advantage is its simplicity. The shortcoming lies in the fact it offers a weaker for lack of max-margin boundary
Summary of Model Fitting

- Choose a good sample dataset which is sufficiently large and representative of typical data behavior
- Training the model to fit with the sampling data
- Crossed validation of the model with testing data
- Modify the training set with more representative data points or change some features
- Modify the Machine Learning algorithm such as dimension reduction, etc.
- Use alternating methods to optimize the performance such as use ensembles or extending trees to forest