International Journal of Computational Intelligence Techniques
ISSN: 0976-0466 & E-ISSN: 0976-0474, Volume 3, Issue 1, 2012, pp.-65-71.
Available online at http://www.bioinfo.in/contents.php?id=36

MOBILE SECURITY USING SUPERVISED LEARNING

PUNDIR P.R. AND GOMASE V.S.

Department of Computer Engineering, J.J.T. University, Jhunjhunu-333001, Rajasthan, India.
*Corresponding Author: Email- pradeep.pundir@gmail.com

Received: March 20, 2012; Accepted: April 27, 2012

Abstract- Wireless networks in small or large coverage are increasingly popular as they promise the expected convergence of voice and data services while providing mobility to users. Combining with current wireless communications infrastructure, wireless computing infrastructure and mobile middleware, mobile commerce provides consumers with faster and personalized services and is becoming one of the most important wireless applications. Unless the transmission, storing and processing of information is secure, neither customers nor service providers will trust mobile commerce systems. From a technical point of view, mobile commerce over wireless networks is inherently insecure compared to electronic commerce over wired networks. A number of machine learning algorithms has to be redesigned to address growing concerns with security due to unlimited explosion of new information through internet cloud and other media. In this paper, we have presented the applications of association rules, privacy decision-tree model, Artificial Neural Network, Support Vector Machine, and HMM. Mobile security is a crucial issue for mobile commerce.
Key words- Machine Learning, Classification, Prediction, Supervised Learning, Mobile.

Citation: Pundir P.R. and Gomase V.S. (2012) Mobile Security Using Supervised Learning. International Journal of Computational Intelligence Techniques, ISSN: 0976-0466 & E-ISSN: 0976-0474, Volume 3, Issue 1, pp.-65-71.

Introduction

Mobile security is a crucial issue for mobile commerce. From a technical point of view, mobile commerce over wireless networks is inherently insecure compared to electronic commerce over wired networks. The reasons are as follows:

Reliability and Integrity: Interference and fading make the wireless channel error prone. Frequent handoffs and disconnections also degrade the security services.

Confidentiality/Privacy: The broadcast nature of the radio channel makes it easier to tap. Thus, communication can be intercepted and interpreted without difficulty if no security mechanisms such as cryptographic encryption are employed.

Identification and Authentication: The mobility of wireless devices introduces an additional difficulty in identifying and authenticating mobile terminals.

Capability: Wireless devices usually have limited computation capability, memory size, communication bandwidth, and battery power. This will make it difficult to utilize high-level security schemes such as 256-bit encryption.

Threats and Attacks

The introduction of distributed systems and the use of networks and communications facilities wire line and now increasingly wireless have increased the need for network security measures to protect data both in real time and non-real time during transmission. To effectively assess the security needs, and evaluate/ choose the most effective solution, a systematic definition of the security goals or requirements and an understanding of the threats is a necessity.
Security threats or security issues can be divided into two types: Passive and Active threats.

Passive Threats
Passive threats stem from individuals attempting to gain information that can be used for their benefit or to perform active attacks at a later time. Active threats are those where the intruder does some modification to the data, network, or traffic in the network.
A passive threat is a situation in which an intruder does not do anything to the network or traffic under attack but collects information for personal benefit or for future attack purposes. Two basic passive threats are described as follows:

Eavesdropping: This has been a common security threat to human beings for ages. In this attack, the intruder listens to things he or she is not supposed to hear. This kind of attack means that the intruder can get information that is meant to be strictly confidential.

Traffic Analysis: This is a subtle form of passive attack. At times, it is enough for the intruder to know simply the location and identity of the communicating device or user. An intruder might require only information such as a message has been sent, or who is sending the message to whom, or the frequency or size of the message. Such a threat is known as traffic analysis.

Active Threats
An active threat arises when an intruder directly attacks the traffic and the network and causes a modification of the network, data, and so forth. The following list details common active attacks:

Masquerade: This is an attack in which an intruder pretends to be a trusted user. Such an attack is possible if the intruder captures information about the user, such as the authentication data simply the username and the password. Sometimes the term spoofing is used for masquerade.

Authorization violation: This occurs when an intruder or even a trusted user uses a service or resources that is not intended for that user. In the case of an intruder, this threat is similar to masquerading; having entered the network, the intruder can access services he or she is not authorized to access. On the other hand, a trusted user can also try to access unauthorized services or resources; this could be done by the user performing active attacks on the network or simply by lack of security in the network/ system.

DoS: DoS attacks are performed to prevent or inhibit normal use of communications facilities. In the case of wireless communications, it could be as simple as causing interference, sending data to a device and overloading the central processing unit (CPU), or draining the battery.

Sabotage: A form of DoS attack, that could also mean the destruction of the system itself.

Modification or forgery of information: This occurs when an intruder creates new information in the name of a legitimate user or modifies or destroys the information being sent. It could also be that the intruder simply delays the information being sent.

Machine Learning
Machine-learning methods can be categorized into four groups of learning activities: symbol-based, connectionist-based, behavior-based, and immune system-based activities. Symbol-based machine learning has a hypothesis that all knowledge can be represented in symbols and that machine learning can create new symbols and new knowledge, based on the known symbols. In symbol -based machine learning, decisions are deducted using logical inference procedures. Connectionist-based machine learning is constructed by imitating neuron net connection systems in the brain. In connectionist machine learning, decisions are made after the systems are trained and patterns are recognized. Behavior-based learning has the assumption that there are solutions to behavior identification, and is designed to find the best solution to solve the problem. The immune-system-based approach learns from its encounters with foreign objects and develops the ability to indentify patterns in data. None of these machine learning methods has noticeable advantages over the others. Thus, it is not necessary to select machine-learning methods based on these fundamental distinctions, and within the machine-learning process, mathematical models are built to describe the data randomly sampled from an unseen probability distribution.
Machine learning has to be evaluated empirically because its performance heavily depends on the type of training experience the learning machine has undergone, the performance evaluation metrics, and the strength of the problem definition. Machine-learning methods are evaluated by comparing the learning results of methods applied on the same data set or quantifying the learning results of the same methods applied on sample data sets. Machine-learning methods use training patterns to learn or estimate the form of a classifier model. The models can be parametric or unparametric. The goal of using machine-learning algorithms is to reduce the classification error on the given training sample data. The training data are finite such that the learning theory requires probability bounds on the performance of learning algorithms. Depending on the availability of training data and the desired outcome of the learning algorithms, machine-learning algorithms are categorized into supervised learning and unsupervised learning. In supervised learning, pairs of input and target output are given to train a function, and a learning model is trained such that the output of the function can be predicted at a minimum cost. The supervised learning methods are categorized based on the structures and objective functions of learning algorithms. Popular categorizations include artificial neural network (ANN), support vector machine (SVM), and decision trees. In unsupervised learning, no target or label is given in sample data. Unsupervised learning methods are designed to summarize the key features of the data and to form the natural clusters of input patterns given a particular cost function. Unsupervised learning is difficult to evaluate, because it does not have an explicit teacher and, thus, does not have labeled data for testing.
Machine learning is the computational process of automatically inferring and generalizing a learning model from sample data. Learning models use statistical functions or rules to describe the dependences among data and causalities and correlations between input and output . Theoretically, given an observed data set X, a set of parameters θ, and a learning model $f(\theta)$, a machine learning method is used to minimize the learning errors $E(f(\theta),$

X), between the learning model *f* (θ) and the ground truth. Without loss of generalization, we obtain the learning errors using the difference between the predicted output *f* (θ̂) and the observed sample data, where θ̂ is the set of approximated parameters derived from the optimization procedures for minimization of the objective function of learning errors. Machine-learning methods differentiate from each other because of the selection of the learning model *f* (θ), the parameters θ, and the expression of learning error *E*(*f* (θ), *X*). Given a training data set *S* with *m* samples (|*S*| = *m*), *d* dimensional feature space *F*, and a *l*-dimensional class label set *C* = {*C*1, …, *Cl*}, we have paired samples and target labels *S* = {(*xi*, *yi*)}, *i* = 1, …, *m*, and *F* = {*f*1, *f*2, …, *fd*}, where *xi* ∈ *X* is an instance and *yi* ∈ *Y* is the class label of instance *xi*.

Fundamentals of Supervised Machine-Learning Methods

In supervised machine learning, an algorithm is fed sample data that are labeled in meaningful ways. The algorithm uses the labeled samples for training and obtains a model. Then, the trained machine-learning model can label the data points that have never been used by the algorithm. The objective of using a supervised machine learning algorithm is to obtain the highest classification accuracy. The most popular supervised machine-learning methods include artificial neural network (ANN), support vector machine (SVM), decision trees, Bayesian networks (BNs), *k*-nearest neighbor (KNN), and the hidden Markov model (HMM).

Association Rule Classification

An association rule can be seen as an extension of the correlation property to more than two dimensions, since it can find associated isomorphism's among multiple attributes. The basics of association rules as follows.

Let *E* = {*I*1, *I*2, . . ., *Ik*} be a set of items and *D* be a database consisting of transactions *T*1, *T*2, . . ., *TN*. Each transaction *Tj*, ∀1 ≤ *j* ≤ *N* is a set of items such that *Tj* ⊆ *E*. We present an association rule *A* ⇒ *B* with the following constraints:

1. ∃*Tj*, *A*, *B* ∈ *Tj* ,
2. *A* ⊆ *E*, *B* ⊆ *E*, and
3. *A* ∩ *B* ∈ φ.

In the above rule, *A* (left-hand side of rule) is called the antecedent of the rule, and *B* (right-hand side of rule) is called the precedent of the rule. Since many such rules may be presented in the database, two interesting measures, *support* and *confidence*, are provided for association rules. *Support* indicates the percentage of data in the database that shows the correlation, and *confidence* indicates the conditional probability of a precedent if the antecedent has already occurred. Using the notations above, we define the *support* and *confidence* below

$$Support(A \Rightarrow B) = \frac{|\#T_i|\, A, B \in T_i}{N}$$

$$Confidence(A \Rightarrow B) = \frac{|\#T_i|\, A, B \in T_i}{|\#T_i|\, A \in T_i}$$

An association rule is considered strong if the support and confidence of a rule are greater than user-specified minimum support and minimum confidence thresholds. Let the above *A* describe frequent patterns of attribute-value pairs, and let *B* describe class labels. Then, association rules can conduct effective classification of *A*. Association rules have advantages in elucidating interesting relationships, such as causality between the subsets of items (attributes) and class labels. Strong association rules can classify frequent patterns of attribute-value pairs into various class labels. However, elucidation of all interesting relationships by rules can lead to computational complexity, even for moderate-sized data sets. Confining and pruning the rule space can guide association rule mining at a fast speed.

Artificial Neural Network

An ANN is a machine-learning model that transforms inputs into outputs that match targets, through nonlinear information processing in a connected group of artificial neurons (as shown in Fig. 1), which make up the layers of "hidden" units. The activity of each hidden unit and output Ŷ is determined by the composition of its input *X* and a set of neuron weights *W* : Ŷ = *f* (*X*, *W*), where *W* refers to the matrix of weight vectors of hidden layers.

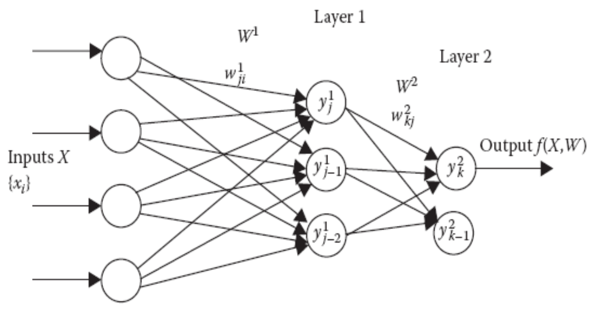

Fig. 1- Two Layer ANN Framework.

When ANN is used as a supervised machine-learning method, efforts are made to determine a set of weights to minimize the classification error. One well-known method that is common to many learning paradigms is the least mean-square convergence. The objective of ANN is to minimize the errors between the ground truth *Y* and the expected output *f*(*X*; *W*) of ANN as *E*(*X*) = sqr(*f*(*X*; *W*) − *Y*). The behavior of an ANN depends on both the weights and the transfer function *Tf* , which are specified for the connections between neurons. The net activation at the *j*th neuron of layer 1 can be presented as in Fig. 1.

$$y_j^1 = T_f\left(\sum_i x_i . w_{ji}^1\right)$$

Subsequently, the net activation at the *k*th neuron of layer 2 can be presented as

$$y_j^2 = T_f\left(\sum_i x_i . w_{ji}^2\right)$$

This transfer function typically falls into one of three categories: linear (or ramp), threshold, or sigmoid. Using the linear function, the output of *Tf* is proportional to the weighted output. Using the threshold method, the output of *Tf* depends on whether the total input is greater than or less than a specified threshold value. Using the sigmoid function, the output of *Tf* varies continuously but not linearly, as the input changes. The output of the sigmoid function bears a greater resemblance to real neurons than do linear or threshold units. In any application of these three functions, we must consider rough approximations.

ANN encompasses diverse types of learning algorithms, the most popular of which include feed-forward back-propagation (BP), radial basis function (RBF) networks, and self-organizing map (SOM).
In feed-forward BP ANN, information is transformed from an input layer through hidden layers to an output layer in a straightforward direction without any loop included in the structure of network . In feed-forward BP ANN, we train the ANN structure as follows. First, we feed input data to the network and the activations for each level of neurons are cascaded forward. We compare the desired output and real output to update BP ANN structure, e.g., weights in different layers, layer-by-layer in a direction of BP from the output layer to the input layer.
RBF ANN has only one hidden layer and uses a linear combination of nonlinear RBFs in the transfer function *Tf* . For instance, we can express the output of a RBF ANN as follows:

$$f(X,W)=\sum_{i=1}^{n} w_i T_f(\|X-c_i\|$$

Where
wi and ci are the weight and center vectors for neuron i
n is the number of neurons in the hidden layer

Typically, the center vectors can be found by using *k*-means or KNN. The norm function can be Euclidean distance, and the transfer function *Tf* can be Gaussian function. ANN methods perform well for classifying or predicting latent variables that are difficult to measure and solving nonlinear classification problems and are insensitive to outliers. ANN models implicitly define the relationships between input and output, and, thus, offer solutions for tedious pattern recognition problems, especially when users have no idea what the relationship between variables is. ANN may generate classification results that are harder to interpret than those results obtained from the classification methods that assume functional relationships between data points, such as using associate rules. However, ANN methods are data dependent, such that the ANN performance can improve with increasing sample data size.

Support Vector Machines

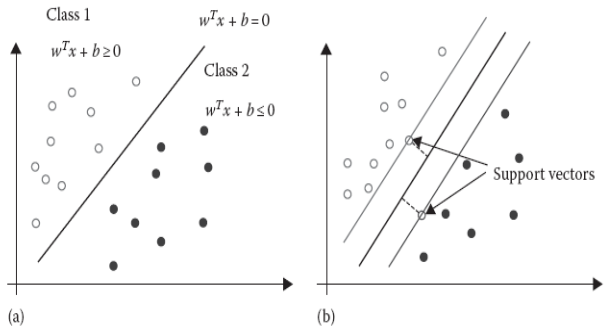

Fig. 2- SVM classification a) Hyperplane in SVM b) Support Vector in SVM

Given data points *X* in an *n* dimensional feature space, SVM separates these data points with an *n* − 1 dimensional hyperplane. In SVM, the objective is to classify the data points with the hyperplane that has the maximum distance to the nearest data point on each side. Subsequently, such a linear classifier is also called the maximum margin classifier.
As shown in Fig. 2, any hyperplane can be written as the set of points *X* satisfying *wTx* + *b* = 0, where the vector *w* is a normal vector perpendicular to the hyperplane and *b* is the offset of the hyperplane *wTx* + *b* = 0 from the original point along the direction of *w*.
Given labels of data points *X* for two classes: class 1 and class 2, we present the labels as *Y* = +1 and *Y* = −1. Meanwhile, given a pair of (*wT*, *b*), we classify data *X* into class 1 or class 2 according the sign of the function *f*(***x***) = *sign*(*wTx* + *b*), as shown in Fig. 2a. Thus, the linear separability of the data *X* in these two classes can be expressed in the combinational equation as *y* · (*wTx* + *b*) ≥ 1. In addition, the distance from data point to the separator hyperplane *wTx* + *b* = 0 can be computed as *r* = (*wTx* + *b*)/‖*w*‖, and the data points closest to the hyperplane are called support vectors. The distance between support vectors is called the margin of the separator (Fig. 2b). Linear SVM is solved by formulating the quadratic optimization problem as follows:

$$\arg\min_{w,b}\left(\frac{1}{2}\|w\|^2\right)$$

$$s.t. y(w^t x+b)\geq 1$$

Using kernel functions, nonlinear SVM is formulated into the same problem as linear SVM by mapping the original feature space to a higher-dimensional feature space where the training set is separable by using kernel functions. Nonlinear SVM is solved by using a soft margin to separate classes or by adding slack variables. SVM is better than ANN for achieving global optimization and controlling the over fitting problem by selecting suitable support vectors for classification. SVM can find linear, nonlinear, and complex classification boundaries accurately, even with a small training sample size. SVM is extensively employed for multi-type data by incorporating kernel functions to map data spaces. SVM is fast, but its running time quadruples when a sample data size doubles.
Unfortunately, SVM algorithms root in binary classification. To solve multi-class classification problems, multiple binary-class SVMs can be combined by classifying each class and all the other classes or classifying each pair of classes.

Decision Trees

A decision tree is a tree-like structural model that has leaves, which represent classifications or decisions, and branches, which represent the conjunctions of features that lead to those classifications. A binary decision tree is shown in Fig. 1., where *C* is the root node of the tree, *Ai* (*i* = 1, 2) are the leaves (terminal nodes) of the tree, and *Bj* (*j* = 1, 2, 3, 4) are branches (decision point) of the tree.
Tree classification of an input vector is performed by traversing the tree beginning at the root node, and ending at the leaf. Each node of the tree computes an inequality based on a single input variable. Each leaf is assigned to a particular class. Each inequality that is used to split the input space is only based on one input variable. Linear decision trees are similar to binary decision trees, except that the inequality computed at each node takes on an arbitrary linear form that may depend on multiple variables. With

the different selections of splitting criteria, classification and regression trees and other tree models are developed.

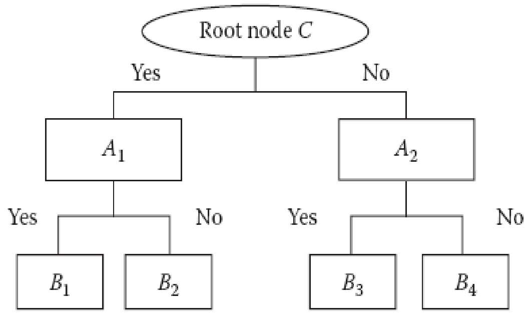

Fig. 3- Decision Tree Structure

As shown in Fig. 3, a decision tree depends on if–then rules, but requires no parameters and no metrics. This simple and interpretable structure allows decision trees to solve multi-type attribute problems. Decision trees can also manage missing values or noise data. However, they cannot guarantee the optimal accuracy that other machine - learning methods can. Although decision trees are easy to learn and implement, they do not seem to be popular methods of intrusion detection. A possible reason for the lack of popularity is that seeking the smallest decision tree, which is consistent with a set of training examples, is known to be NP-hard.

Hidden Markov Model

In the previous sections, we have discussed machine-learning methods for data sets that consist of independent and identically distributed (iid) samples from sample space. In some cases, data may be sequential, and the sequences may have correlation.

In HMM, the observed samples *yt*, *t* = 1, …, *T*, have an unobserved state *xt* at time *t* (as shown in Fig. 4). Fig. 4 shows the general architecture of an HMM. Each node represents a random variable with the hidden state *xt* and observed value *yt* at time *t*. In HMM, it is assumed that state *xt* has a probability distribution over the observed samples *yt* and that the sequence of observed samples embed information about the sequence of states. Statistically, HMM is based on the Markov property that the current true state *xt* is conditioned only on the value of the hidden variable *xt*−1 but is independent of the past and future states. Similarly, the observation *yt* only depends on the hidden state *xt*. The most famous solution to HMM is the Baum−Welch algorithm, which derives the maximum likelihood estimate of the parameters of the HMM given a data set of output sequences.

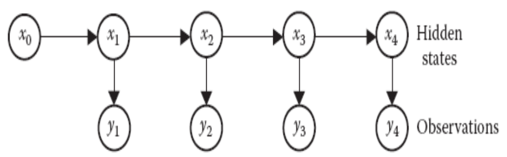

Fig. 4- Architecture of HMM

Let us formulate the HMM using the above notations as follows. Given that *Y* and *X* are the fixed observed samples and state the sequence of length *T* defined above, *Y* = (*y*1, …, *yT*) and *X* = (*x*1, …, *xT*), then, we have the state set *S* and the observable data set *O*, *S* = (*s*1, …, *sM*) and *O* = (*o*1, …, *oN*). Let us define *A* as the state transition array [*Ai,j*], *i* = 1, …, *M*, *j* = 1, …, *M*, where each element *Ai,j* represents the probability of state transformation from *si* to *sj*. The transformation can be calculated as follows:

$$A_{i,j} = prob(x_t = s_j \mid x_{t-1} = s_i)$$

Let us define *B* as the observation array [*Bj,k*], *j* = 1, …, *M*, *k* = 1, …, *N*, where each element *Bjk* represents the probability of the observation *ok* has the state *sj*. The observation array can then be calculated as follows:

$$B_{i,j} = prob(y_t = o_k \mid x_t = s_j)$$

Let us define π as the initial probability array [π*t*], *t* = 1, …, *T*, where π*t* represents the probability that the observation *yt* has the state *si*, *i* = 1, …, π can be expressed as

$$\pi_i = prob(x_i = s_i)$$

We then define an HMM using the above definitions, as follows:

$$\lambda = (A, B, \pi)$$

The above analysis is the evaluation of the probability of observations, which can be
summarized in the algorithm in four steps as follows:

1. *Initialize for t = 1, according to the initial state distribution π.*
2. *Deduct the observation value at time t corresponding to Equation 2.8.*
3. *Deduct the new state at time t + 1 according to Equation 2.9.*
4. *Iterate Steps 2 through 4 until t = T.*

Given the HMM , we can predict the probability of observations *Y* for a specific state sequence *X* and the probability of the state sequence *X* as

$$prob(Y \mid X, \lambda) = \prod_{t=1}^{T} prob(y_t \mid x_t, \lambda)$$

$$and$$

$$prob(X \mid \lambda) = \pi_1.A_{12}.A_{23}.........A_{T=1T}$$

Then, we obtain the probability of observation sequence *Y* for state sequence *X* as follows:

$$prob(Y \mid \lambda) = \sum_{x} prob(Y \mid X, \lambda).prob(X \mid \lambda)$$

Users are generally more interested in predicting the hidden state sequence for a given observation sequence. This decoding process has a famous solution known as the Viterbi algorithm, which uses the maximized probability at each step to obtain the most probable state sequence for the partial observation sequence. Given an HMM model λ, we can find the maximum probability of the state sequence (*x*1, …, *xt*) for the observation sequence (*y*1, …, *yt*) at time *t* as follows:

$$2 \le t \le T, \qquad 1 \le j \le M$$

$$p_t(i) = \max_{x_1......x_{t-1}} \left(prob\left(x_1, ..., x_t = s_i, y_1, ..., y_t \mid \lambda\right)\right)$$

The Viterbi algorithm follows the steps listed below:

1. *Initialize the state for t = 1, according to the initial state distribution π*:

$$\rho_i(i) = \pi_i B_i(y_i), \quad 1 \le i \le M, \quad \Psi_i(i) = 0.$$

2. *Deduct the observation value at time t corresponding to the following equation*:

$$\rho_t(j) = \max_i \left[\rho_t(i) A_{ij} \right] B_j(y_t),$$

and

$$\Psi_t(j) = \arg\max_i \left[\rho_{t=1}(i) A_{ij} \right], \quad 2 \le t \le T, \quad 1 \le j \le M$$

3. *Iterate Steps 2 through 4 until t = T*.

HMM can solve sequential supervised learning problems. It is an elegant and sound method to classify or predict the hidden state of the observed sequences with a high degree of accuracy when data fit the Markov property. However, when the true relationship between hidden sequential states does not fit the proposed HMM structure, HMM will result in poor classification or prediction. Meanwhile, HMM suffers from large training data sets and complex computation, especially when sequences are long and have many labels. The assumption of the independency between the historical states, or future states and the current states also hampers the development of HMM in achieving good classification or prediction accuracy.

Bootstrap, Bagging and AdaBoost

In complex machine-learning scenarios, a single machine-learning algorithm cannot guarantee satisfactory accuracy. Researchers attempt to ensemble a group of learning algorithms to improve the learning performance over single algorithms. In the next sections, we will introduce several popular ensemble learning methods, including random forest, bagging, bootstrap, and AdaBoost.

Bootstrap is most employed to yield a more informative estimate of a general statistics, such as bias and variance of an estimator. Given sample set *X* = {*xi*}, *i* = 1, …, *m*, a set of parameters θ, and a learning model *f* (θ), a machine-learning method minimizes the learning errors *E*(*f* (θ), *X*). In bootstrap, *m* data points are selected randomly with replacements from data set *X*. By repeating this sampling process independently *B* times, we obtain *B* Bootstrap sample sets. The parameters in function *f* (θ) can be estimated by each sample set, and we obtain a set of bootstrap estimate {θ̂*j*}, *j* = 1, …, *B*. Then, the estimate on bootstrap samples is

$$\hat{\theta}^{B} = \sum_{j=1}^{B} \hat{\theta}_j$$

As the bootstrap selects samples repeatedly from *X*, each data sample has 1/*m* probability of being chosen in each selection. When *m* is big enough, the probability that *xi* is selected *mboot* times is Poisson distribution with mean unity. We can obtain its unbiased estimate of parameters θ̂ statistically over sample data *X*. Then, we can obtain the bootstrap estimate of the parameter bias at bias *b*

$$bias_b(\theta) = \hat{\theta}^{B} - \hat{\theta},$$

and the bootstrap estimate of the parameter variance

$$\mathrm{var}_b(\theta) = \frac{1}{B} \sum_{j=1}^{B} (\hat{\theta}_j - \hat{\theta})^2$$

Bootstrap aggregating (bagging) aims to sample data sets for an ensemble of classifiers. In bagging, *m*' < *m* data points are selected randomly with replacement from the data set *X*. Repeating this sampling process multiple times, we obtain different training sample sets for each member of the ensemble of classifiers. The final decision is the average of the member-model decisions by voting. Bagging is commonly used to improve the stability of decision trees or other machine-learning models. However, bagging can result in redundant and lost information because of replacement.

Boosting is used to boost a strong machine-learning algorithm with an arbitrarily high accuracy by using a weighted training data set. Boosting algorithms start by finding a weak machine-learning algorithm that performs better than random guessing. Then, member classifiers are integrated into an accurate classification ensemble over the most informative subset of the training data. Boosting modifies bagging in two ways: weighting the sample and weighting the vote. Boosting can result in higher accuracy than bagging when a data set is noise free, although bagging stays more robust in noisy data.

Adaptive boosting (AdaBoost) is the most popular variant of boosting algorithms. Given training data set *S* with *m* examples (|*S*| = *m*), and an *l*-dimensional class label set *C* = {*C*1, …, *Cl* }, we have a paired data set *S* = {(*xi*, *yi*)}, *i* = 1, …, *m*, where *xi* ∈ *X* is an instance and *yi* ∈ *Y* and *yi* ∈ *C* form the class label of sample *xi*. We assign a sample weight *wt*(*i*), *t* = 1, … *T*, to each sample *xi* to determine its probability of being selected as the training set for a member classifier at iterative step *t*. This weight will be raised if the sample is not accurately classified. Likewise, it will be lowered if the sample is accurately classified. In this way, boosting will select the most informative or difficult samples over each iterative step *k*. AdaBoost algorithms can be summarized in the following steps (as shown in Fig. 5).

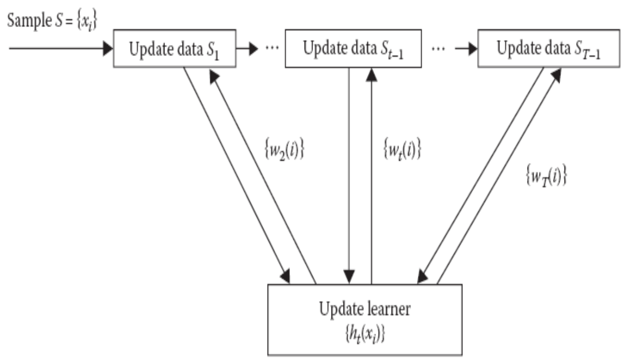

Fig. 5- Workflow of Adaboost

In the above steps, α*t* measures the confidence when assigning those samples to the classifier *ht* at step *t*. AdaBoost offers accurate machine-learning results without overfitting problems that are common in machine-learning algorithms. AdaBoost is simple for implementation and has a solid theoretical background and good generalization. Therefore, AdaBoost has been employed for various learning tasks, e.g., feature selection. However, Adaboost can only guarantee suboptimal learning solutions after greedy learning.

Conclusion

Different technologies have been developed by different organizations that come together in two main families: the family of IEEE wireless networks (Wi-Fi, WiMAX, etc.) and the family of cellular mobile networks (1G, 2G, 3G, etc.). Current trends are developing an all-IP core network to provide mobile services to users regardless of their location or terminal. On the other hand, the wireless network evolves to provide more bandwidth; this is the 4G network, which will probably be a combination of the different technologies above or simply a new technology offering very high bandwidth comparable to fixed networks. The goal remains to maximize the radio resource, support mobility, providing multimedia services (voice, data, image), transparency to the user and ensuring the security of communications.

References

[1] *http://www.geni.net/GDD/GDD-06-32.pdf*.

[2] *http://varma.ece.cmu.edu/cps/*.

[3] Abbott D., Matkovsky P. and Elder J. (1998) *Proc. of IEEE*.

[4] *http://www.fda.gov:80/cdrh/osel/guidance/1618.html*.

[5] Ghosh S. and Reilly D. (1994) *27th Hawaii International Conference on Systems Science,* 3, 621-630.

[6] Qixin Wang, et al. (2006) *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 4268-4275.

[7] *http://ostp.gov/pdf/nitrd_review.pdf*.

[8] Fawcett T. (2003) *SIGKDD Explorations,* 5(2), 40-148.

[9] Fawcett T. (1997) *AAAI Workshop*.

[10] Fanning K., Cogger K. and Srivastava R. (1995) *Journal of Intelligent Systems in Accounting, Finance and Management,* 4, 113-126.

[11] Phillip M. Wells, Koushik Chakraborty and Gurindar S. Sohi, (2007) *International Conference on Parallel Architectures and Compilation Techniques*.

[12] Jaideep Vaidya and Chris Clifton (2004) *IEEE Security & Privacy Magazine*, 2(6), 19-26.

[13] Jane W.S. (1994) *Proceedings of the IEEE*, 82(1), 83-94.

[14] Health informatics-Point-of-care medical device communication First Edition (2004) *ISO/IEEE 11073-10101*.

[15] Jungkeun Yoon, Brian D. Noble, Mingyan Liu (2006) Minkyong Kim (2006) 4*th Annual ACM/USENIX Conference on Mobile Systems Applications, and Services*.

[16] Vassilios S. Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin and Yannis Theodoridis (2004) *ACM SIGMOD Record*, 3(1), 50-57.

[17] Wenbo He, Xue Liu, Hoang Nguyen, Klara Nahrstedt and Tarek Abdelzaher (2007) *Proceedings of the 26th Annual IEEE Conference on Computer Communications* (*INFOCOM*).

[18] Lui Sha (2001) *IEEE Software*, 18(4), 20-28.

[19] Lui Sha, Ragunathan Rajkumar and John Lehoczky (1990) *IEEE Transaction on Computers*, 39(9), 1175-1185.

[20] *http://domino.watson.ibm.com/comm/research.nsf/pages/r.kdd.innovation.html*.