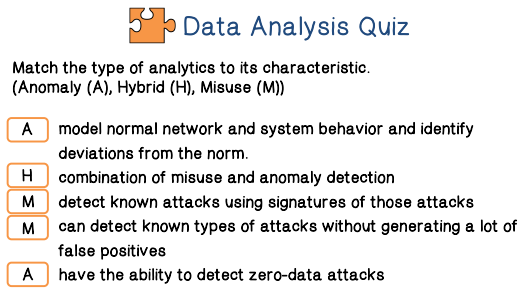
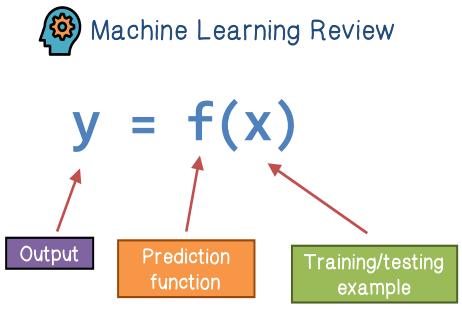
In this lesson, we will review machine learning, then discuss how to use it to determine if a network flow is benign, or anomalous. The goal of applying machine learning to intrusion detection is to be able to automatically and quickly identify new attacks. The earlier we can detect bad behavior, the faster we can stop it.

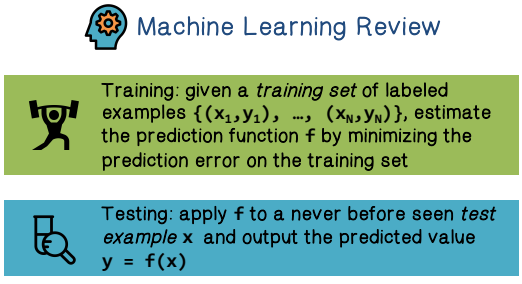


Let us start with a quiz on data analysis. Match the type of analytics to its characteristic. The types are anomaly detection, hybrid, or misuse detection.

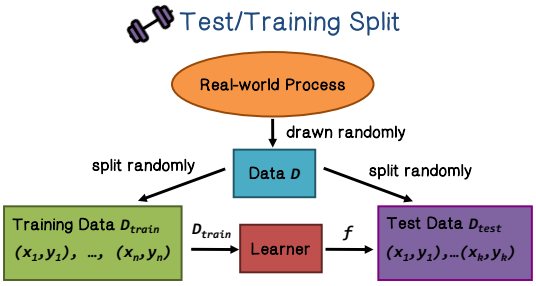
The first one, model normal network and system behavior and identify deviations from the norm. This is anomaly detection. Second, combination of misuse and anomaly detection, this is hybrid detection. The third one, detect known attacks using signatures of those attacks, this is misuse detection. The fourth one, can detect known types of attacks without generating a lot of false positives, this is misuse detection. The last one, have the ability to detect zero-day attacks, this is anomaly detection.



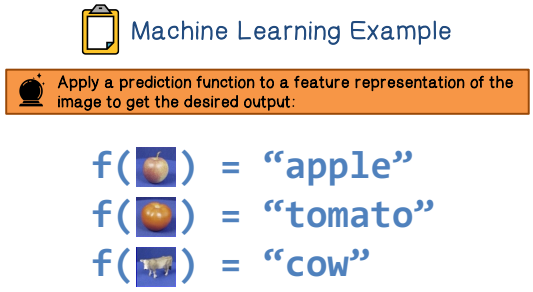
Let us have a quick overview of machine learning. The task of machine learning is that, given training examples as input, we would learn a function that can predict the output. Then, once we learn this function, we can apply it to testing examples and this function can then produce the output. For example, the input can be fruits and the output can be the names of the fruits. For example, is it an apple or orange? A partition function will be able to analyze the features of these fruits and determine whether it is apple versus orange.



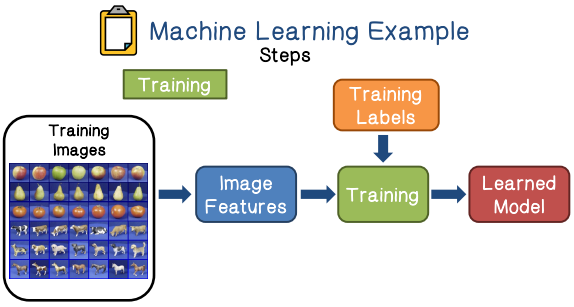
The first step in machine learning is training. Here, the x's are the examples and the y's are the labels of these examples. Then, the machine learning algorithm is going to produce the best possible function by minimizing the prediction error on this training data set. We are going to discuss a few example machine learning algorithms. In a testing phase, we apply the learned function to a set of test data, which was not used in training, and the function should be able to produce predicted value of y given x.



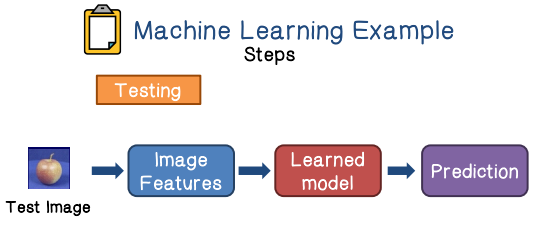
Ideally, data used in machine learning should reflect the real world. Therefore, we use a process to draw data from the real-world application. Then, that data is randomly split into a training data set and test data set. We can then apply a machine learning algorithm to training data to learn the particular function that can be applied to test dataset.



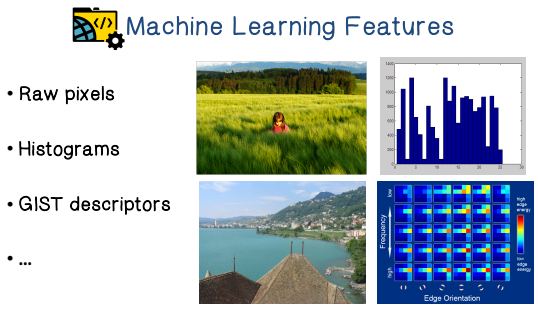
Here is an example of machine learning. Suppose you want to learn the name or predict the name of an object, for machine learning purpose, these objects are represented as feature vectors of their images. For example, apple, tomato, and cow are all represented as feature vectors, and then our function based on these feature vectors can predict the type of the object.



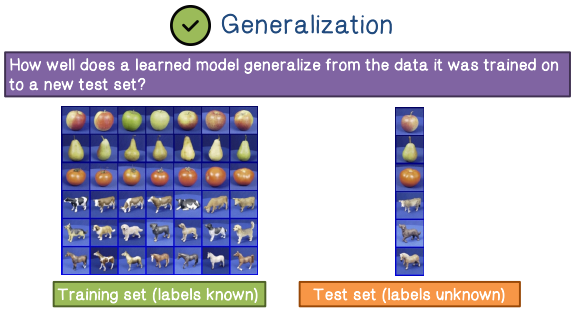
So, let us go over the steps in this machine learning example. We start with the images of these objects; we then construct features from these images so that each of these objects now is represented as a feature vector. Of course, we put the correct label to each object. For example, all these are apples, all these are pears, and so on. Then, once we have the objects represented as feature vectors with their labels, we can apply machine learning algorithm to train a predictive function. This predictive function is the learned model.



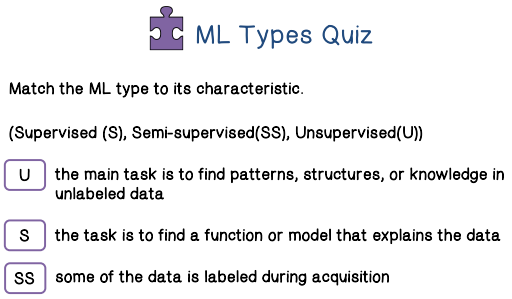
In testing, again, we start with the image of an object, we extract the same set of image features as in training. That is, now the object is represented as a feature vector. We then apply the learned model to this feature vector. The learned model, which is a predictive function, is going to tell us the prediction or the type of this object.



The features using machine learning are very much application dependent. For images, these are useful features: the raw pixels, the histogram, and GIST, which is a representation of the scene, and so on. We are going to discuss features useful for network monitoring later.

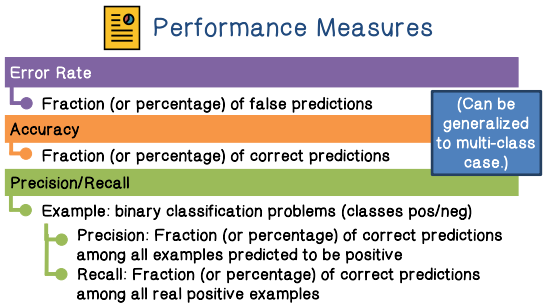


A good machine learning model should be able to generalize from the data that it was trained on to new test dataset. That is, we do not want to just learn the data from the training dataset. That is, given the objects in a training dataset, for example, the different apples here, we want the machine learning algorithm to be able to learn the general characteristics of the apples so that given an image of apple that was not previously seen in the training dataset, the learned model is able to correctly predict that it is indeed an apple. In short, generalization is the most important property of machine learning.

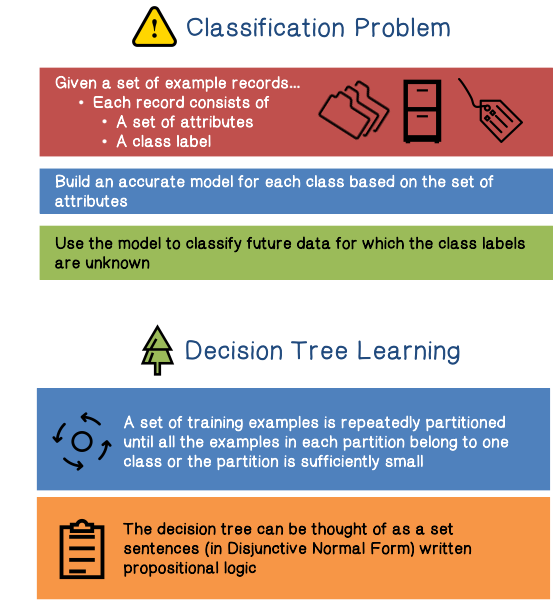


Let us do a quiz on different types of machine learning. Match the machine learning type to its characteristic, supervised, semi-supervised, or unsupervised.

The first one, the main task is to find patterns, structures or knowledge in unlabeled data that is unsupervised because the data is not labeled. That is, the machine learning algorithm is not told which label belongs to which data. Second, the task is to find the function or model that explains the data. This is supervised because the data is labeled, and machine learning is going to produce a function that models the relationship between the data and its label. Third, some of the data is labeled during acquisition. This is semi-supervised.



Now, let us discuss some of the key performance metrics of machine learning. The error rate is the fraction of false predictions. Accuracy is the fraction of correct predictions. Precision is a fraction of correct predictions among all examples predicted to be positive. Recall is the fraction of correct predictions among all real positive examples. Precision and recall are not limited to two-class or positive versus negative problems. That is, they can be generalized into multi-class applications.



Many machine learning applications are classification problems, that is, the training dataset contains records with attributes or features and class labels. Then, the machine learning algorithm is to produce a model that would output the class label based on a set of attributes or features. Once we have this model, we can then use it to classify future data.

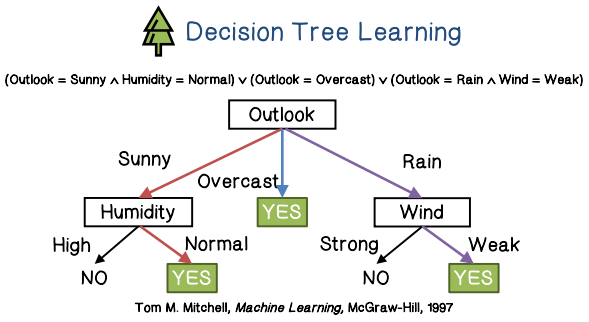
Now, let us discuss a few examples of classification models. The first one is decision tree, here the training data is repeatedly partitioned until all examples in each partition belong to one class. The decision tree can also be thought of as a set of rules that describe the decision logic.

Decision Tree Learning

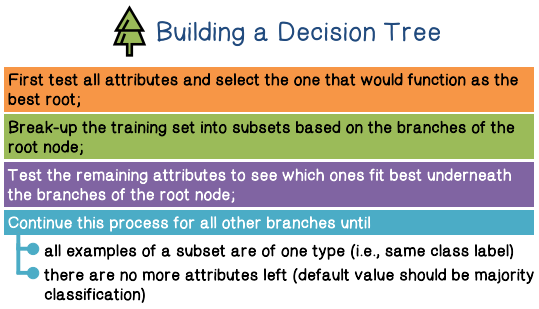
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Tom M. Mitchell, *Machine Learning*, McGraw-Hill, 1997

Let us go over a simple example, the application here is to learn the decision tree to decide whether we should play tennis or not. So, the class label here is yes or no to play tennis, and the features or attributes include the weather conditions.

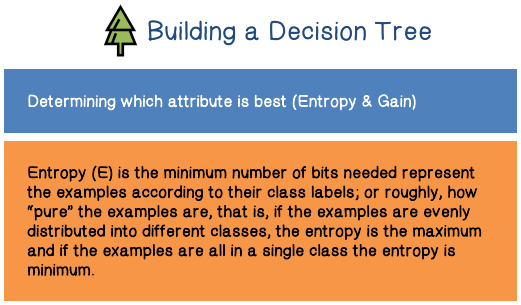


And here is the decision tree produced by a machine learning algorithm. Again, a decision tree can be represented as a set of rules. For example, if you see outlook equal to sunny and humidity equal to normal, then yes, we play tennis. If the outlook is overcast then also yes to play tennis, or if the outlook is rain and wind is weak, then we also say yes to play tennis.

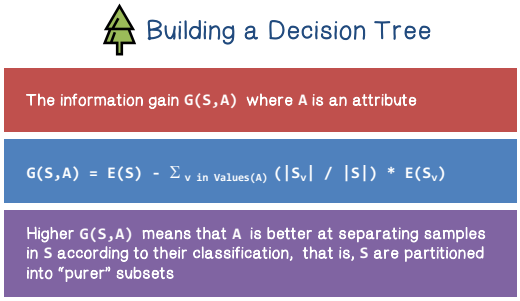


Here is a high-level overview of the process of building a decision tree. We first find the best attributes that can serve as the root. Then we break-up the training dataset into subsets based on the values of this root attribute, that is, we grow from the root into branches. Therefore, for each subset we test the remaining attributes to see which is best at partitioning this subset, that

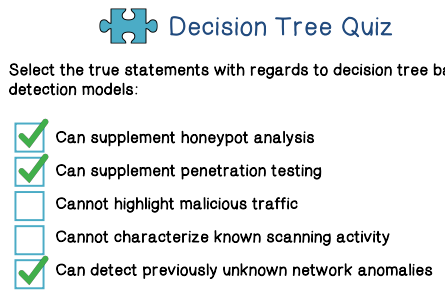
is, for each branch, we test attributes to determine how to best grow into branches. We continue this process until all examples in a subset belong to one class, or there is no more attribute left to further partition to subset. In this case, the default class label is the majority.



Now the question is, how do we determine which attribute is the best at partitioning a set into subsets? We use entropy and information gain to determine this. The entropy of a set is the minimal number of bits needed to represent examples in this set according to their class labels. Therefore, roughly speaking, the entropy of a set represents how pure the examples are in this set. That is, if the examples of this set are evenly distributed into different classes, then the entropy is the maximum and if the examples are all in a single class, then the entropy is the minimum.

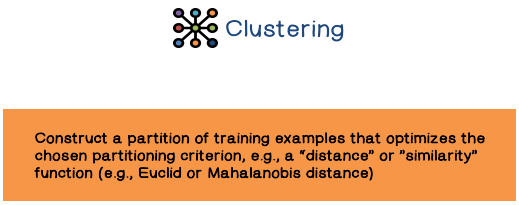


In order to determine which attribute is the best at partitioning a set, we compute its information gain. As you can see from this formula, a high information gain value for attribute A means that A is better as separating samples in a set S according to their classification, that is, using A, S a petition into purer subsets. Therefore, we should select the attribute that has the highest information gain and use it to partition a set.

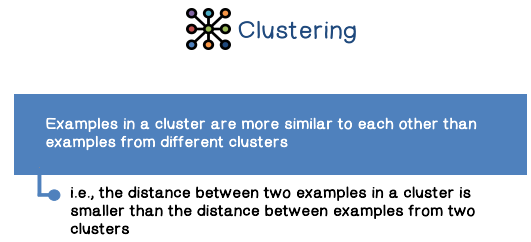


Now, let us do a quiz on decision tree. Select the true statements with regards to decision tree based detection models. That is, if we use a decision tree based models for intrusion detection, which statements are true?

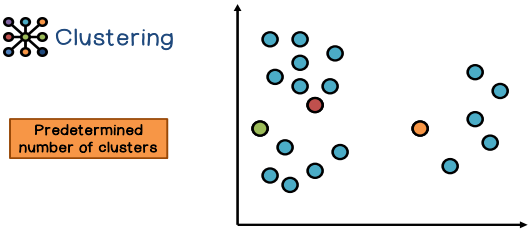
First, can it supplement honeypot analysis? Yes. Can it supplement penetration testing? Yes. Can it highlight or detect malicious traffic? Yes, and therefore the third statement is false. Fourth, can it characterize known scanning activity? Yes, and therefore the fourth statement is false. The last one, can it detect previously unknown network anomalies? The answer is yes.



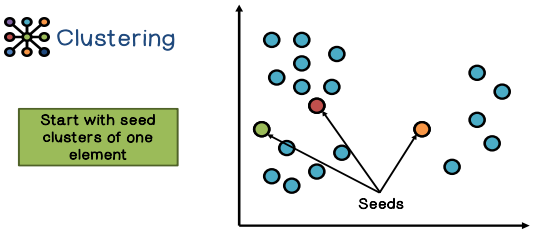
Clustering can also be used for classification. In clustering, we assign training examples into different clusters based on some distance measure.



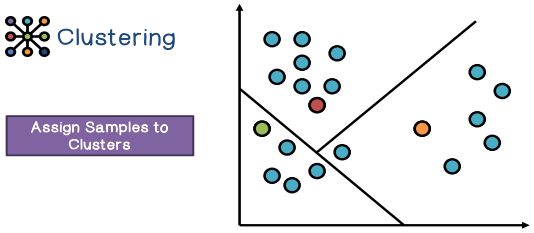
That is, examples in the same cluster are more similar to each other than examples from different clusters. Typically, we defined some distance function to measure similarity.



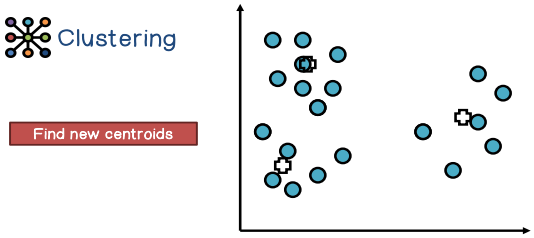
Here is an overview of the clustering process. Typically, we predetermine the number of clusters.



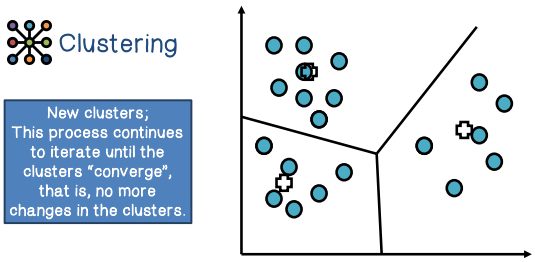
We also start with seed clusters, each with one element. That is, each seed is a representative example of a cluster.



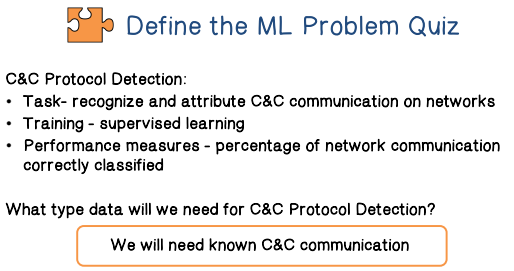
They may assign samples to these C clusters based on distance measures.



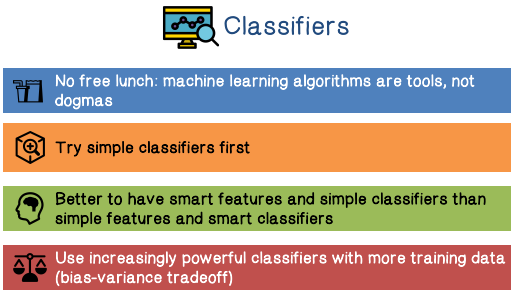
Then we adjust or find a new centroid. A centroid is the center of the cluster according to distance measure.



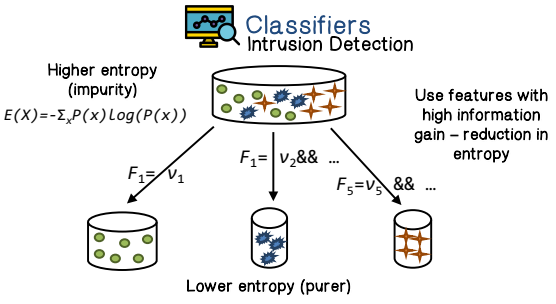
Then based on the new centroids, we can adjust cluster membership, so the samples are assigned to the best fit clusters. The example here belongs to this cluster because it is closer to its centroid than to the centroids of other clusters. In other words, cluster membership is based on distance to the centroid of the cluster. We continue this process of finding new centroids and adjusting cluster membership, until the clusters converge. That is, there is no more changes in the cluster membership. Once the clusters are finalized, then each cluster can be represented by its centroid. Given a test data, it is assigned to a cluster if its distance to the centroid is the shortest.



Now, let us do a quiz. The context is applying machine learning to the problem of detecting botnet command-and-control. Supposed we decide to use supervised learning, that means data should be labeled, and we want a high percentage of network flows to be correctly classified. The question is which type of data do we need for C&C protocol detection? We need label data. That is, we need known C&C communication examples.

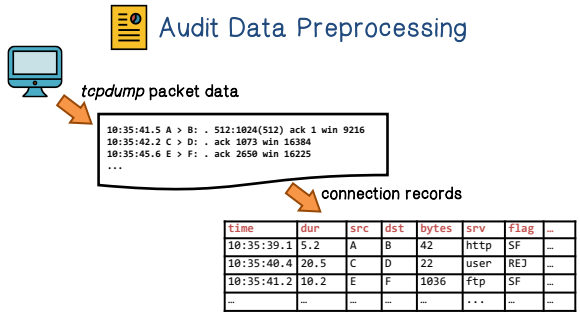


As a way to summarize, classifiers are very useful to many applications, including security applications, such as intrusion detection. However, we should be reminded that, machine learning algorithms are just tools. Therefore, we need to use them carefully. We should always try simple classifiers first. Most specifically, it is more important to have smart features than complicated or smart classifiers. The reason is that, smart features capture the domain knowledge. This is the key to producing effective classifiers. On the other hand, if we have a lot of data or wide variety of data, we can use more powerful classifiers.

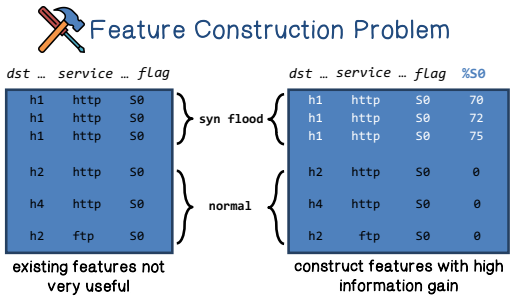


The problem of intrusion detection can be considered as a classification problem. Suppose we are monitoring network traffic. The traffic can contain normal traffic and various types of attack traffic. The process of intrusion detection is to determine, among this set of all traffic, which ones are normal and which ones belong to different types of attacks. In other words, intrusion detection is about classifying each network flow into normal or different attack types. Therefore, it

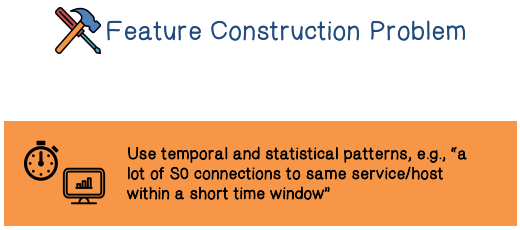
is natural to think that, we can apply machine learning algorithm, to learn a classifier, to detect intrusions. That is, given a traffic data which has higher entropy, because it is mixed with intrusions and normal traffic, our goal is to partition the data into subsets so that each has a pure class, that is, either normal or one of the attack types. As we have discussed in decision tree, we need to use features that has high information gain in this process of partitioning the original dataset into subsets of pure classes. So, that is the high-level intuition of applying machine learning to the problem of intrusion detection. Now let us discuss some details.



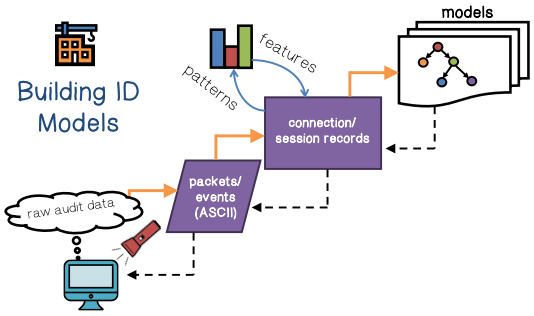
We start with raw data captured from the network tab, for example, packet data, from which we can summarize the data into connection records. Each record has a set of attributes, for example, time stamp, duration, source IP, destination IP, number of bytes, the service, and the flag, and so on. For example, SF means the connection has gone through both SYN and FIN, and REJ means that the connection request has been rejected.



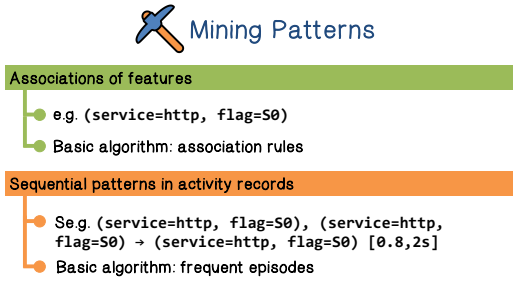
As we have discussed earlier, having smart features is the key to building effective classification models. This is known as the feature construction problem. In the context of intrusion detection, the attributes or features of the connection records are not very useful. For example, if you look at SYN flag, that means, we only see the initial SYN packet. Of course, with SYN flag, we will see a lot of connection attempts with SYN flag. Of course, normal data can also have this S0 flag, for example, because the connection request or response is lost. That is, by looking at the feature flag, it is not sufficient to detect intrusion. On the other hand, if you look at the percentage of S0 flag connections to particular host, then we can see that this feature can distinguish intrusion from normal. In other words, this feature has high information gain. The problem is that this feature is not in the original feature set, which means we need to construct features that have high information gains.



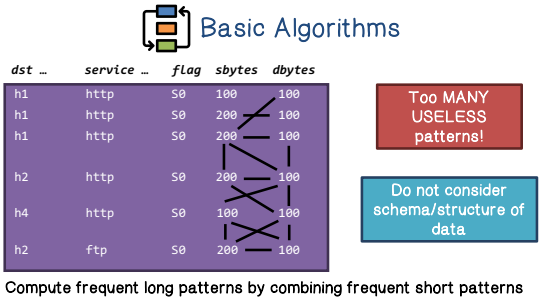
So, the question is how we construct features that have high information gains? That is, features that are useful to distinguish normal versus intrusions. Our approach is to use temporal and statistical patterns associated with intrusions, for example, a lot of S0 connections to the same service and host, within a short period of time. This pattern is associated with SYN flag. Once we have these patterns, we can then construct features accordingly that can be used to detect these intrusions.



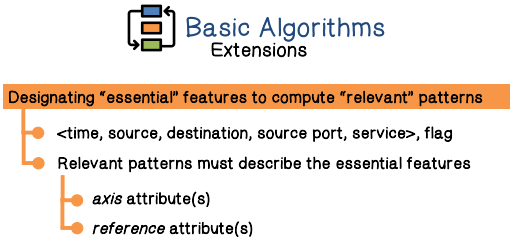
Here is a high-level overview of this process. We start with raw audit data. For example, packets. We summarize data into connection records. Then we find frequent patterns, and then we compare the patterns to determine the unique patterns associated with intuitions. We then construct features according to these patterns. With these features, we can learn classification models. This process is iterative. In fact, each step can iterate over and over to improve performance.



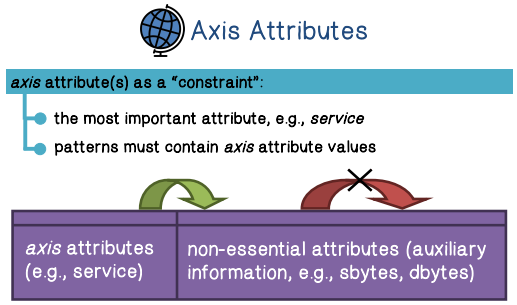
Now, let us discuss how we find patterns. We use data mining algorithms to find patterns. We first find the associations among the features. For example, this association tells us that there are a lot of S0 HTTP connections. We then find the patterns that describe how the associations tend to appear in a sequence. For example, this pattern says that, if you see the first two, there is 80 percent chance that you will see the third one, and all these are within a two second terminal.



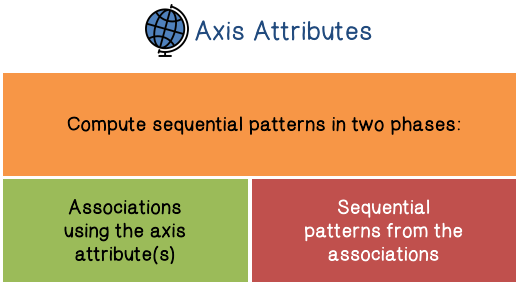
The basic algorithms are the association rules and frequent episodes. These algorithms can produce many useless patterns. For example, these algorithms can produce associations and frequent sequences involving source bytes and destination bytes. These patterns are not useful for intrusion detection. The reason is that, these patterns do not even involve more important attributes such as a service and the flag.



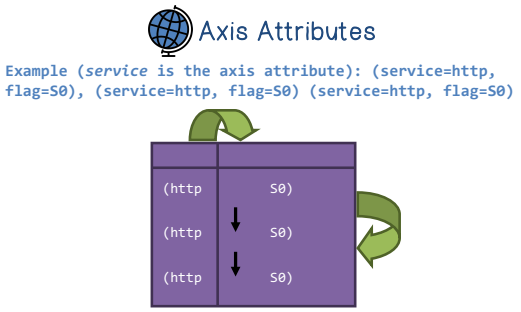
Therefore, we modify the basic algorithms, so that they only produce relevant or useful patterns. The intuition is that, the relevant patterns must describe these essential features. More specifically, we use some of these essential features as the so-called axis attributes, or reference attributes, to constrain how the patterns can be computed.



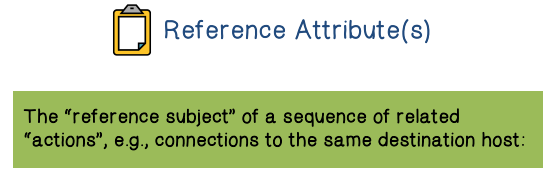
An axis attribute is typically the most important attribute, for example, the service. An association must contain an axis attribute. Therefore, we can eliminate patterns that describe associations among non-essential attributes.



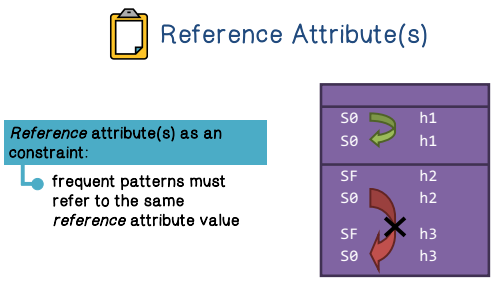
Once we compute associations involving axis attributes, we can then compute sequential patterns involving these associations, and that is how we compute sequential patterns.



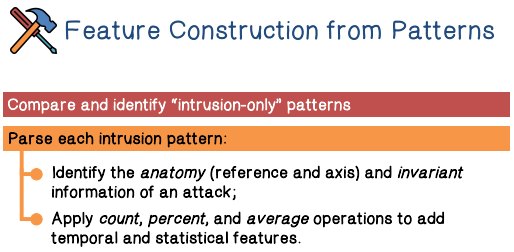
Here is an example. We use service as the axis attribute, we can then compute a number of associations. For example, there is frequent association of S0 and http.



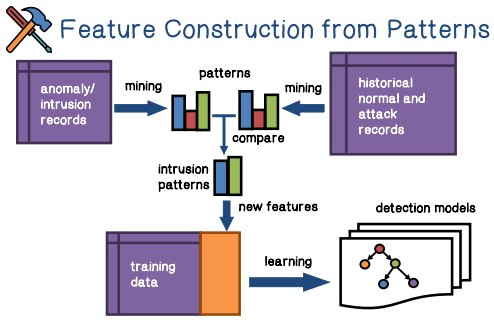
We then use reference attribute to compute a sequence of related associations. These associations in a sequence all refer to the same reference subject because they are essentially a sequence of related actions. For example, in this sequential pattern, all associations are referring to the same destination host.



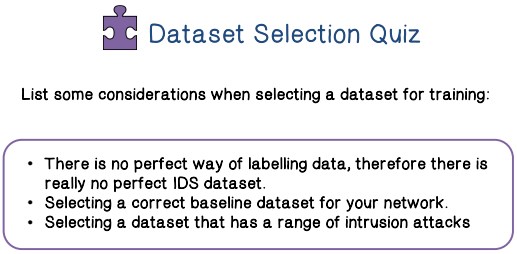
Again, the reference attribute can be used as a constraint to filter out patterns that are not useful for intrusion detection.



After we compute the frequent patterns from the normal data and from the intrusion data, we can then compare and identify the unique patterns that are from the intrusion data. We can then use these patterns to construct futures to build classifiers. We call the axis and reference attributes the anatomy of an attack and the intrisic attributes are those that are independent of the axis and reference attributes. Since our patterns are frequent and sequential in nature, we apply these operators to add the corresponding, temporal and statistical features.

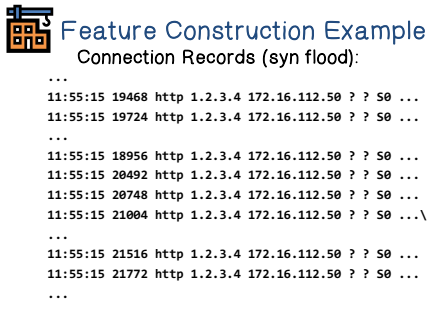


To summarize, we mine patterns from both the intrusion and a normal datasets and compare the patterns. We identify the patterns associated with only the intrusion dataset. We construct futures according to these patterns and add these features into the training dataset. We can then apply a machine learning algorithm to learn a classifier that can detect intrusions.

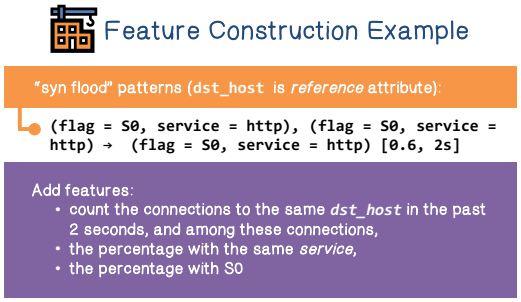


Now let us do a quiz. Suppose we want to select a dataset to train a classifier to detect intrusions, what are the considerations?

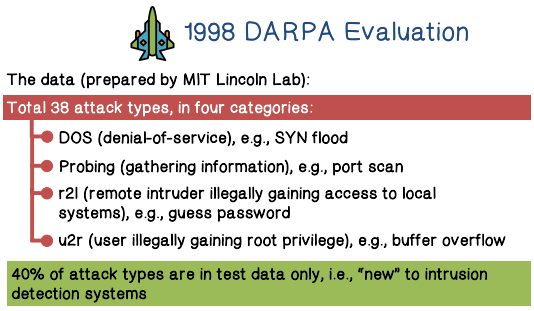
First, we should recognize that, there is no perfect way of labeling data and therefore there is really no perfect IDS dataset. What we can do is to select a baseline dataset for network. Then, we also use a dataset that has a range of intrusion examples.



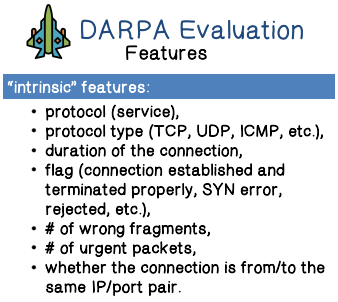
Let us consider an example of feature construction. Suppose here is the set of connection records associated with syn flood.



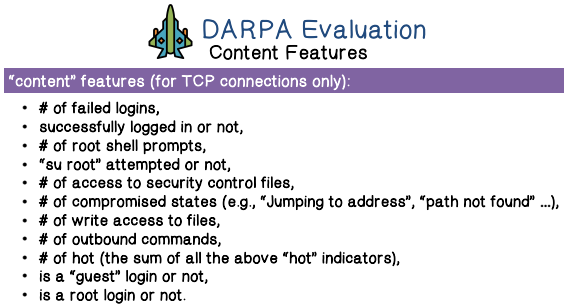
Using service as the axis attribute and destination host as a reference attribute, we can compute a frequent pattern. Based on this pattern, we can construct these following features, count the connections to the same destination host in the past two seconds, and among these connections, the percentage with the same service, and the percentage that has the S0 flag.



Now, let us discuss an evaluation of our approach. We use the DARPA evaluation dataset. There are 38 attack types in the data and these attack types are in four categories, Denial-of-Service, probing, remote-to-local, and user-to-root. In particular, remote-to-local means gaining local access and user-to-root means gaining superuser privilege. In this evaluation, training dataset is provided but 40 percent of the attack types are only in test data. That is, these attack types are new to the intrusion detection systems.



Here are the features that we defined and constructed for each connection record. The intrinsic features such as protocol, duration and flag. These are the ones that are inherent to any connection. In other words, they can be used for many applications, not just intrusion detection.



We construct a feature for the purpose of intrusion detection. The content features are based on analysis of the payload. For example, we can tell how many failed login attempts, whether the login was successful or not, number of root shells, whether a su root has been attempted, number of access to security control files, and so on.

DARPA Evaluation

Features constructed from mined patterns:

Temporal and statistical "traffic" features that describe connections within a time window:

- # of connections to the same *destination host* as the current connection in the past 2 seconds, and among these connections,
- # of rejected connections,
- # of connections with "SYN" errors,
- # of different services,
- % of connections that have the same service,
- % of different (unique) services.

We compute frequent patterns from the connection records and then we construct temporal and statistical features accordingly. These include number of connections to the same destination host as the current connection in the past two seconds and among these connections, number of rejected connections, number of connections with SYN errors, number of different services, the percentage of connections they have the same service, percentage of different or unique services.

DARPA Evaluation

Features constructed from mined patterns:

Temporal and statistical "traffic" features that describe connections within a time window (cont'd.):

- # of connections that have the same *service* as the current connection, and among these connections,
- # of rejected connections,
- # of connection with "SYN" errors,
- # of different destination hosts,
- % of the connections that have the same destination host,
- % of different (unique) destination hosts.

And the number of connections they have the same service as the current connection and among these connections, number of rejected conditions, number of connections with SYN errors, number of different destination hosts, percentage of connections that have the same destination host, percentage of different or unique designation hosts.

DARPA Evaluation

Features constructed from mined patterns:

Example "content" connection records:

dur	p_type	proto	flag	l_in	root	su	compromised	hot	…	label
92	tcp	telnet	SF	1	0	0	0	0	…	normal
26	tcp	telnet	SF	1	1	1	0	2	…	normal
2	tcp	http	SF	1	0	0	0	0	…	normal
149	tcp	telnet	SF	1	1	0	1	3	…	buffer
2	tcp	http	SF	1	0	0	1	1	…	back

Here we show examples of the values of content features in the connection records. We also label each connection as normal or one of the attacks.

DARPA Evaluation

Example rules produced by machine learning:

```
buffer_overflow :- hot >= 3, compromised >= 1,
su_attempted <= 0, root_shell >= 1.

back :- compromised >= 1, protocol = http.
```

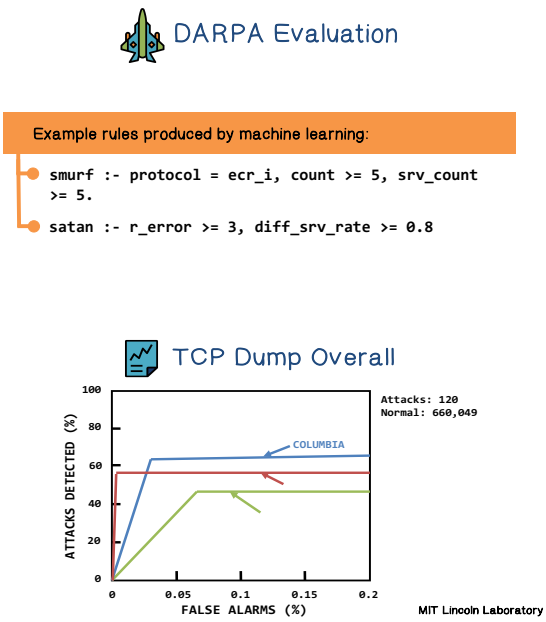
Here are some example rules produced by machine learning. For example, a buffer overflow attack can be detected because there are a number of indicators of compromise. A root shell has to be attained and su root has not be attempted.

DARPA Evaluation

Example "traffic" connection records:

dur	p_type	proto	flag	count	srv_count	r_error	diff_srv_rate	…	label
0	icmp	ecr_i	SF	1	1	0	1	…	normal
0	icmp	ecr_i	SF	350	350	0	0	…	smurf
0	tcp	other	REJ	231	1	198	1	…	satan
2	tcp	http	SF	1	0	0	1		normal

Here we show examples of the values of traffic features. Again, each connection record is labeled as normal or one of the attacks.



And here are some example rules produced by machine learning. For example, in a smurf attack, ICMP echo request has been sent to the same service on the same host. Satan is a probing attack and therefore there are many different ports or services being probed and since many of them are closed, many connection attempts are rejected.

This shows that our data mining and machine learning-based approach performed quite well. Other approaches are all based on rules hand-coded by experts.

A Framework for Constructing Features and Models for Intrusion Detection Systems

This paper describes the MADAM ID framework for Mining Audit Data for Automated Models for Intrusion Detection. The framework uses (a) data mining algorithms to compute frequent patterns and extract features in system audit data, and (b) applies ML classification algorithms to audit records in order to generate IDS rules. This helps create an IDS in an automated manner as compared to the traditional method of manual and rigorous analysis, and also ensures that the IDS is extensible and adaptable. Learned rules replace manually encoded intrusion patterns and profiles. Features are selected based on statistical patterns in the audit data. To produce a combined detection model from multiple detection models based on the correlation of intrusion evidence, meta-learning is proposed as a technique. It is a mechanism for inductively learning the correlation of predictions by a number of classifiers.

Data mining:

The data mining piece of the framework mines association rules and frequent episodes. Section 3 on 'Mining Audit Data' gives detailed explanations. Mining association rules involves extracting correlations from a database table and assigning "support" and "confidence" values based on the formal expression given in Section 3.2. Instead of searching for all possible associations, the data mining process is streamlined by utilizing schema-level information about the audit records. This helps in making use of only essential features and leaving out the auxiliary information. The paper argues that relevant association rules should describe patterns related to essential features, also called axis features. Frequent episodes represent sequential patterns in audit records. The formal expression is given in Section 3.3. The original frequent episodes algorithm is modified to run in two phases: (a) find frequent associations using axis features, (b) generate frequent serial patterns from these associations. So, this combines the associations among features and sequential patterns into a single rule.

Feature construction:

The mined frequent episodes which also contain associations among the features are used to construct temporal statistical features for building classification models. The process involves (a) identifying "intrusion only" patterns, and (b) parsing the intrusion patterns to construct features. These are described in Sections 4.1 and 4.2, respectively. Heuristic algorithms are used to identify intrusion only patterns. The paper describes an encoding procedure that represents each pattern as a number, where the order of significant digits corresponds to feature importance. The

ordering used was flag, axis feature, reference feature, rest of the essential features, and rest of the attributes. The distance between two patterns is a number computed as the digit-wise absolute difference between the encodings. An automatic procedure is used to parse a frequent episode and construct features, using three operators – count, percent, and average. The algorithm is described in detail in Section 4.2. These measure connections within a time window, and sharing a reference value, so this is temporal in nature.

Section 5 on 'Experiments' describes the methodology in some detail, but the takeaway is that frequent patterns mined from audit data can be reliably used as user anomaly detection models.