# Machine Learning

Machine learning (ML) is a subfield of Artificial Intelligence. Algorithms in ML, learns to solve problems and generalizes from samples, observations, environments, etc. with no specific programming instructions. According to Tom Mitchell (1997), machine learning means "a computer program that improves its performance at some task through experience."

Machine learning has been widely applied in numerous applications including control systems (e.g. no-driver vehicles), search engines, medical diagnosis, bioinformatics, natural language processing, computer visions, stock market analysis, etc.

Machine learning tasks can be broadly divided into 3 main categories:

- Supervised learning

  The learner is trained with known outputs (targets) or labeled examples. In summary, the learner must find the function that map inputs to outputs. Examples of supervised learning include artificial neural networks, decision trees, Naïve Bayes, Bayesian believe network, etc.

- Unsupervised learning

  The learner is trained with unlabeled data; this means that the learner strives to find hidden structures in the data. Typically, input data are clustered based on their feature properties, in which the clusters are not known in advance. Unsupervised learning includes self-organizing maps (SOM), K-means.

- Reinforcement learning

  The learner learns without an explicit teacher, instead it learns from interaction with the environment. The learner makes the adjustment according to the environmental feedback. Hence, performance is evolved over time. Reinforcement learning includes robot control systems, Q-learning, etc.

Methodologies such as decision trees, neural networks, Naïve Bayes, can be found in Introduction to Data Science course, World Class, Regis University (2014) and other sources such as [Patel, 2003], [Piatesky-Shapiro]. The contents in this chapter present some other interesting approaches.

### *Ensemble Methods*

The weak classifier/learner is referred to a classifier, where it accuracy is only slightly better than a random guessing. The motivation of ensemble methods is based on the idea of using multiple learners and combining their predictions, with the hope to improve the performance of the individual classifier. However, more data are required for training and testing. The often-used techniques to generate more data are bootstrap.

*Bootstrap* is a statistical resampling method. The sample points are drawn randomly with **replacement**.
*Bagging* (**B**ootstrap **agg**regat**ing**) the sample is drawn with replacements from the data set. A classifier learns from each *bootstrap* sample. Then, all classifiers are aggregated (e.g. average for regression, and majority vote for classification) and then used to classify sample points from the test set.

Generally, it applies on one type of classifier. Decision trees are a widely used classifier for bagging. There are other advantages such as reducing the over-fitting. In addition, it is easy to train the classifier in parallel. Bagging is suitable for unstable learners (e.g. small change in the training sets results in large variation in the classification such as decision tree and neural network) [Langseth, 2012]. The random forest decision tree is an example of bagging.

*Boosting* is an ensemble method like bagging. But, each successive classifier increases weights for misclassified examples and decreases weights for well classified examples (from the last classifier). The weak learners will evolve over time. Finally, a weight vote is utilized for prediction. In many applications, boosting performs better than bagging. Boosting is a very popular method. It can combine both different models and adjusted (unequal) weights in a new model. There are many variations of boosting. The most popular one is Adaboost.

- ***Adaboost***

In 1995, Freund and Schapire introduced the AdaBoost (**Ada**ptive **Boost**ing) algorithm. The performance is very impressive and tends not to overfit. Other advantages include simple, fast, easy to program, flexible (can use with any learning algorithm), no prior knowledge about weak learner, etc. However, it is prone to (uniform) noise. Adaboost is among the top ten algorithms for data mining.

The idea of Adaboost is adjusting the weights of examples up or down. The weights of examples that have been misclassified previously or harder to classify will be increased. Contrary, if the examples are easy to classify, the weights will be decreased. The focus is on the hard to classify examples. In summary, successive weights in training sets are the results of re-weighting using past ensemble history. Then, the weighted or majority vote is combined into a single predictor. Adaboost employs exponential error loss as an error function, instead of typical squared Euclidian distance. Adaboost can be used for binary classification problems (two possible classes) and multiclass (more than two classes).

Adaboost Algorithm [Montillo, 2009]: 
Initialize weights for data points
- Repeat this for each iteration:
  - Fit classifier to training data
  - Compute weight classification error
  - Compute weight for classifier from the error

    - Update weights for data points
- Final classifier is weighted sum of all single classifiers

This is an example of using AdaBoost in R:

```
> library(ada)
> boost_ada <- ada(x=X, y =labels)
```

ada is in the 'ada' packgage. Here, assume that decision tree is used as a classifier so the 'rpart' package must be loaded. X is the features matrix, and labels are vectors with 0-1 class labels. After the object 'ada' is created, commands such as predict, summary, update, or plot can be invoked. Note that there are variations for multiclass Adaboost.

- ***Random Forests*** (Breiman, 2001)

Random Forest (RF) is an ensemble learning for classification and regression. It is an extension of decision trees. The idea is based on bagging, where each tree is built from a bootstrap sample. Bagging helps to reduce prediction variance, with bias remaining the same. Random forest is expected to have better accuracy than a single tree.

In RF, each tree is built upon independent random sample, but with the same distribution for all trees in the forest. Trees in random forest are allowed to grow fully without pruning. After a large number of trees are generated, the most popular class is voted. Hence, the name of these procedures is called random forests.

The generalization error rate depends on correlation between trees (lower is better) and strength of single trees in the forest (higher is better). Two important parameters are involved: number of trees and number of features. Increasing number of features for each split will result in increasing correlation as well as strength of single trees. Brieman (2001) claimed that error rates from random split feature selections compare favorably to Adaboost but are more robust to noise and/or outliers. In addition, a large number of trees (aka Law of Large number) always converge, so there is no problem of overfitting. Trees can grow in parallel.

Using Random Forest in R: [Zhao, 2013]

1) with function randomForest() in package randomForest This function cannot handle missing data so an imputation process is required. Also, the maximum number of levels is limited to 32 for ach categorical level.
2) cforest () from package party using conditional inference trees as base learners.

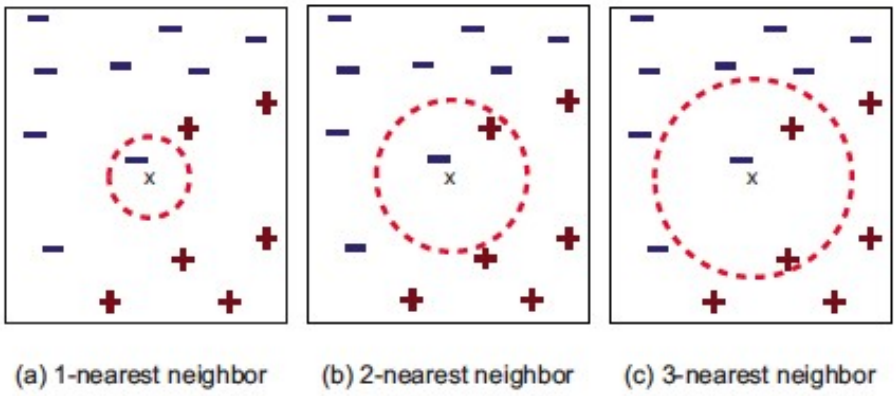
For example:

```
> library(randomForest)
> rf <- randomForest(outcome~ pred1+pred2, data=trainData,ntree=500,importance=TRUE)
> print(rf) # view results
> importance(rf) # each predictor importance
```

> rf.predict = predict (rf,newdata=data$val)

For more details, please refer to: http://cran.r-project.org/web/packages/party/party.pdf

 ***K- Nearest Neighbors Classification***




Source: Rudin (2012) MIT OCW

This learning algorithm can be used for classification and regression. It is one of the simplest algorithms in machine learning. The main idea is that a new data item will be classified using the same label as the majority of the k nearest neighbors in the feature space. The boundaries between different classes are in the shape of a Voronoi diagram of the training data.

Choosing the right K is critical for the performance of the algorithm. If K is too small, we just model the noise. If the K is too big, neighbors may include many points from other classes.
In case that the distances are spread among the neighbors, a distance-based voting scheme where closer neighbors are more influence should be applied. To make the distance measure more meaningful (e.g income has a wider range than height), attributes should be normalized.

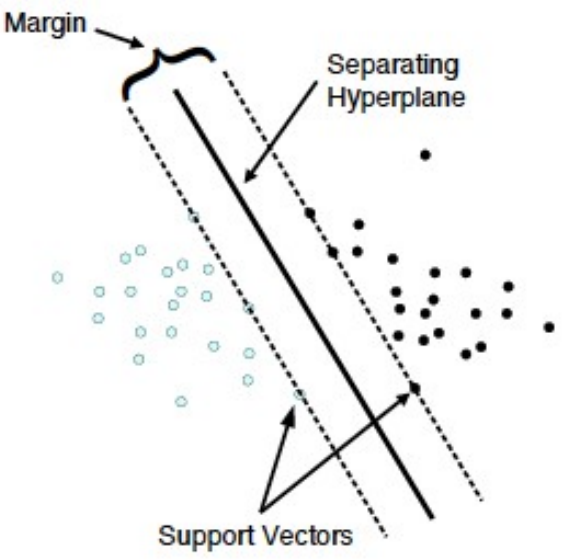
K-NN advantages are simple with no complex parameters but yet it is powerful. There are many applications such as handwritten character classifications, intrusion detection, and fault detection, etc. However, the algorithm can be slow and expensive (e.g. when there are new data points, the distance for all examples must be recomputed)

An example of K-Nearest Neighbor Classification in R:
>library(class)
> knn_model <- knn(train=X.train, test=X.test, cl= Resp.Label,k=K)

Need to install and load class package Here, X.train and X.test are training and testing matrices (or data frame), respectively. cl represents class attributes (correct answers) for the training examples. Finally, k is K neighbor. Learn more about KNN and its options from: http://cran.r-project.org/web/packages/class/class.pdf

 ***Support Vector Machines*** Why SVM?

Neural networks for supervised and unsupervised learning show promising results. In addition, MLP contains universal approximation properties that can literally map any functions. Nevertheless, there are critical issues for employing neural networks such as local minima; optimal number of the neurons is unknown, no unique solutions (e.g. there are many linear classifiers or hyperplanes for data separation but only one is considered the best)




SVM: Classification (linear separable case)

Source: http://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf

SVM main focus is about maximum margin. SVM is known as large margin classifier. When the margin of an example is referred to the distance from the example to the decision boundary. The points lying on the boundaries are known as *support vectors* with the reason that they support the separation boundary lines.

The margin is positive when the example is on the correct side of the decision boundary, or else the margin is negative. Furthermore, it is anticipated that all examples should have large margins so that they are further away from the decision boundary, and thus we can be certain about our classification decisions. The margin therefore, is used as a confident

measurement for the predictions. In other words, SVM selects the hyperplane that has the largest distance or greatest margin from the examples.

The simple case of SVM called linear SVM using linear classifier. However, the separation can be extended to non-linear boundaries, where the kernel trick applied.
The idea is that the original feature space can be mapped to some higher-dimensional feature space via some transformation function (e.g. $\phi: x \rightarrow \varphi(x)$) where the training data is separable. Most of real world problems are non-linear separable, therefore, a curved decision boundary is needed.

The kernel function is a function that is equal to an inner product in some feature space. Kernel Trick's concept is to convert the problem to the optimization problem in dual form. Then, solve the problem by trying to maximize Lagrangian with respect to the function lambda. Kernel function plays an important role for the SVM performance. There are different types of kernel functions with different purposes such as polynomial, Gaussian radial basis function, etc.

SVM can be used for classification and regression. SVM Applications are text/image classification, hand-written character recognition, spam filtering, etc. Advantages of using SVM include no local minima and scaling well with high dimensional data. Choosing the right kernel for the task, however, can be complicated. Nevertheless, SVM is among the top performance classifiers.

The other algorithm that produces a large margin is Adaboost. However, the algorithms such as decision trees, logistic regression, and perceptron do not produce a large margin.

An example of using SVM in R:
> library(e1071)
> svm.model <- svm(PredV, data=Traindata, kernel="radial", cost=C)
> svm.pred <- predict(svm.model, Testdata)
> summary(svm.model)

The SVM algorithm is in the 'e1071' package. Here, PredV is the dependent variable or model to be fit, kernel is the kernel type and can be changed (e.g. linear, polynomial, sigmoid), cost is the cost of constraints violation used in the Lagrange formulation. Check the documents on how to specify different parameters or how to change them.
More detail about SVM can be found at:
http://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf and
http://www.jstatsoft.org/v15/i09/paper.

- ***Principle Components Analysis (PCA)***

The goal of using PCA includes reducing the dimensionality of the data, decreasing redundancy, compressing the data, and preparing data for further analysis by other

techniques. Examples of applications include computer graphics, neuroscience, gene expressions, etc.

To reduce dimension, data are projected on the principal directions, which capture most of variability of the original data. At the beginning, the number of principal components is equal to the number of variables in the original data. However, only the first few principal components are used since the variances of most components may be too low, and will be discarded. Therefore, the number of variables can be reduced in further analysis. It is important to note that, if the original variables are not correlated, the PCA will not reduce the dimension of the data.

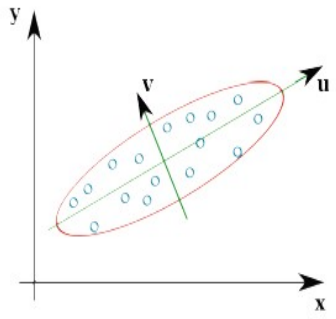



Figure 1: PCA for Data Representation

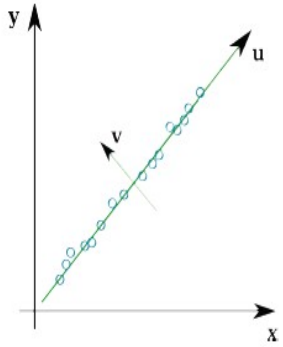



Figure 2: PCA for Dimension Reduction

Source: Gillies. F.D. (2011)

Figure 1, a data set of two variables on the X-Y coordinate. Here, data vary the most (principal direction) in the U axis, and V axis as the secondary direction. The directions U and V are known as principal components. Figure 2, each data point in (X,Y) coordinate is transformed into (U,V) coordinate. In the case of U-V axis, all data in V coordinates are close to zero, so data can be presented using only variable U. As a result, the dimension is reduced by 1.

In R, the PCA can be performed using prcomp() or princomp(). Data is expected to be in the form of data frame (observations in rows, and variables in columns)

- princomp( ) # uses eigenvectors (prone to fail if number of variables is greater than number of observations)
- prcomp( ) # uses a similar (but not identical) technique called singular value decomposition (SVD)

```
> pca <- princomp(data, cor=T)
> summary(pca, loadings=T)
```

For PCA details in R, please refer to:
http://cran.r-project.org/web/packages/HSAUR/vignettes/Ch_principal_components_analysis.pdf

 ***Association Rules***

The aim of the learning association is to find the associations between items stored in the database. Market basket database, for instance, is a collection of data collected while barcodes of products are scanned into the system. In this database, a large record of purchasing transaction is stored. Each record contains lists of all items bought by one customer on a single purchase. The products bought together information can be used for cross selling, promotion, store layouts, customer segments, etc.

Assuming that X and Y are the products.

$P(Y|X)$ is the probability that a customer who buys $X$ also buys $Y$

Example: $P(chips|salsa) = 0.8$

The association rules present the association between item-sets. In general, rules are in the form of if-then statements $X \rightarrow Y$. Three common measures that express the degree of uncertainly of a rule are: support, confidence, and lift.

Support ($X \rightarrow Y$) = $P(X \cup Y)$

Confidence ($X \rightarrow Y$) = $P(Y|X) = \frac{P(X \cup Y)}{P(X)}$

Lift ($X \rightarrow Y$) = $\frac{\text{Confidence } (X \rightarrow Y)}{P(Y)} = \frac{P(X \cup Y)}{P(X)P(Y)}$

In summary, support is the probability of a transaction containing both $X$ and $Y$. Confidence is the conditional probability that a transaction containing $X$ also contains $Y$ or probability that $Y$ appear in transactions that contain $X$. Lift is the ratio of confidence to probability that contain $Y$.

Apriori is a classic algorithm for mining the association rules. The idea of the algorithm is counting number of transactions, finding frequent item-sets, and generating association's rules.

## Apriori Algorithm

- Method:
  - Let k=1
  - Generate frequent itemsets of length 1
  - Repeat until no new frequent itemsets are identified
    - Generate length (k+1) candidate itemsets from length k frequent itemsets
    - Prune candidate itemsets containing subsets of length k that are infrequent
    - Count the support of each candidate by scanning the DB
    - Eliminate candidates that are infrequent, leaving only those that are frequent

Source: Tan et.al (2006)

In R, implement the Apriori algorithm using function *apriori ( )* in package *arules*. For example:

```
> library (arules)
> rules <- apriori(dataset, parameter=list(supp=0.8,conf=0.7)) # show arbitrary sup, conf
> inspect(rules)    # list rules (lhs, rhs) including support, confidence, and lift
> summary(rules) # get a summary of rules characteristics
```

More examples using R, please refer to [Zhao, 2013].

Reference:


*Applied Data Mining and Statistical Learning* (2015). Pennsylvania State University, Department of Statistics Online Programs. Retrieved from: https://onlinecourses.science.psu.edu/stat857/node/129

Breiman, L. (2001) *Random Forests*. Machine Learning. Vol. 45, Issue 1. pp. 5-32. Retrieved from: http://link.springer.com/article/10.1023%2FA%3A1010933404324

Chang, A. *R for machine learning*. MIT OpenCourseWare. Retrieved from: http://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/lecture-notes/MIT15_097S12_lec02.pdf

Freund, Y. & Schapire, E.E (1995). A decision-theoretic generalization of on-line learning and an application to boosting. Proceedings of the Second European Conference on Computational Learning Theory. 1995, pp.23-37.

Freund, Y. & Schapire, E.E (1999). *A Short Introduction to Boosting*. Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, September, 1999. Retrieved from: http://www.site.uottawa.ca/~stan/csi5387/boost-tut-ppr.pdf

Gillies, F.D. (2011) Principal Component Analysis. Probabilistic Inference. Imperial College. London. Retrieved from: http://www.doc.ic.ac.uk/~dfg/ProbabilisticInference/IDAPILecture15.pdf

Patel, N. (2003) *Data Mining*. MIT OCW. Retrieved from: http://ocw.mit.edu/courses/sloan-school-of-management/15-062-data-mining-spring-2003/lecture-notes/

Kaynig-Fittkau, V. Practical Machine Learning. Harvard University. Retrieved from 13-PracticalMachineLearning.pdf

Langseth, H. (nd) Lecture 3 *Bagging &Boosting+SVMs* TDT33173 Machine Learning. Retrieved from: http://www.cs.bham.ac.uk/~axk/MLE/boost/Lecture3.pdf

Liaw, A. and Wiener, M.(2002) Classification and Regression by randomForest. RNews. Vol.2/3 Dec. Retrieved from: http://cogns.northwestern.edu/cbmg/LiawAndWiener2002.pdf

Mitchell, T. (2011). Machine Learning. Carnegie Mellon University. Spring 2011. Retrieved from: http://www.cs.cmu.edu/~tom/10701_sp11/lectures.shtml

Montillo, A.A. (2009). *Random Forests* (Statistical Foundations of Data Analysis), Temple University. Retrieved from: http://www.dabi.temple.edu/~hbling/8590.002/Montillo_RandomForests_4-2-2009.pdf

Piatesky-Shapiro, G. (nd). Machine Learning and Data Mining – Course Notes. Retrieved from: http://kdnuggets.com/data_mining_course/course_notes.pdf

Rudin, C. (2012). Prediction: Machine Learning and Statistics. Retrieved from: http://ocw.mit.edu/courses/sloan-school-of-management/15-097-predictionmachine-learning-and-statistics-spring-2012/lecture-notes/

Schapire, R. (2007) *Theory and Applications of Boosting*. NIPS conf. Retrieved from: http://media.nips.cc/Conferences/2007/Tutorials/Slides/schapire-NIPS-07-tutorial.pdf

Sewell, M (2007). Ensemble Methods. Retrieved from: http://mleg.cse.sc.edu/edu/csce822/uploads/Main.ReadingList/ML_ensemble.pdf

Tan, P. Steinback, M and Kumar, V. (2006) *Introduction to Data Mining.* Pearson. Education, Inc. (2006). Retrieved from: http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap6_basic_association_analysis.pdf

Zhao, Y. (2013) R and Data Mining: Examples and Case Studies. Retrieved from: http://cran.r-project.org/doc/contrib/Zhao_R_and_data_mining.pdf