Guide to SWAG

What is Machine Learning?

Machine learning (ML) is a powerful emerging field of computer science and mathematics that has had huge impacts in a variety of fields recently. A few technologies that utilize ML are: the Netflix movie recommendation system, voice recognition, and the Google self-driving car.

How does a Machine Learning problem look?

The way machine learning works is as follows:

1. Acquire training data
2. Train your ML classifier on this training data
3. Feed your ML classifier data that needs to be classified

Much of these terms are jargon to an everyday person, and that's fine. There is nothing too complicated about these steps, and I will explain it fully. ML represents objects that are going to be classified as a pair of two "things"—a vector x and label y. The vectors x is simply a set of attributes about the object you are trying to represent, and the label y is a +1 or -1 depending on the object. Training data are instances where you have both x and y for each object you are representing. Data that needs to be classified has an x, but the y is unknown. Thus, it is the job of our classifier to inform us what the y should be. Still not making any sense? Let's give a tangible example.

Example: Apples and Oranges:

Suppose we have a crate of apples and oranges. Say we wanted to explain these apples and oranges in a ML friendly way. We could set the vector x to hold the volume and weight of the fruit, and set the label y equal to +1 if it was an orange and -1 if it was an apple. We do this for every fruit, and soon enough, we have our training data. Say we are now given another crate of fruit, but we can't see the fruit. However, someone is telling us the size and weight of each fruit in the crate. We simply load our training data into a machine learning classifier, and every time this person tells us information about a fruit in the crate, we give our ML classifier that information and we should be able to tell the person whether the fruit is an apple or orange.

Using SWAG:

One can use SWAG two different ways, the GUI or simply by accessing the code directly (for more advanced users. The GUI offers the user an interface that allows use of SWAG with complete automatic configuration, and also leaves open the option of customizability, if needed. At the bare minimum, the user must have a set of training data with which to train your classifier(s). The training data should be organized into a CSV file(comma separated value) file. SWAG expects data to be organized as follows: each row should represent one instance like such:

1, 2, 7, 2, 2, 3, 8, 9, 5, 4, 6, -1

Where the last element is the label (-1 or +1, 0 if unknown) and all preceding elements are attributes. SWAG automatically caches any classifiers that are created, to remove the need to retrain classifiers and easily reuse them in the future.

Algorithms present:

k-Nearest Neighbors (kNN):

kNN is one of the simplest ML algorithms available, however do not its simplicity fool you—it is a very accurate algorithm as well. The parameter k is usually defined by the user, and the algorithms essentially tries to find the k closest vectors (closest is defined by some distance metric, usually Euclidean distance), and simply returns the label that occurred most frequently in those k closest neighbors. Training the classifier initially is very quick; however classification takes time linearly proportional to the total training size. It makes sense to use kNN when the number of attributes and data set size is low (< 100 attributes and < 1000 instances), because classification can quickly start taking a large amount of time.

Support Vector Machine (SVM):

SVMs are one of the most modern and accurate ML classifiers available today. One of the most important aspects of the SVM is its ability to transform your data with various functions to more accurately classify it. The SVM performs classification based on the support vectors (vectors that lie at the boundary between the +1's and -1's). Training the classifier initially can take a reasonable amount of time, however classification can be quickly (faster than kNN). SVMs are almost always good for any classification task. Their versatile and powerful nature make them a must have in any classification context.

Naïve Bayes (NB):

NB classifiers use probability (more specifically Bayes' rule) to solve classification tasks. They are very good with classification tasks that have discrete attributes, such as spam filtering. Naïve Bayes takes a reasonable amount of time to construct the classifier; however classification of vectors is done in constant time, making it a fairly quick algorithm. Naïve Bayes is also another

AdaBoost (AB):

AdaBoost is not an actual classifier algorithm; however it is something called a meta-algorithm. It essentially wraps itself around a set of classifiers and weights them according to their training error. It is a very useful algorithm that helps increase the accuracy of multiple classifiers. The time it takes to construct and classify vectors depends on the constituent algorithms within.

k-Means (kM):

k-Means is an unsupervised classifier, which means it can take in a set of vectors and no labels, and it attempts to find clusters of vectors that most similar to each other. The number of clusters it tries to find is defined by the user as k. This can potentially take quite a bit of time, as it can take time exponentially proportional to the data set size; however the implementation included in SWAG has a maximum amount of iterations before it stops.