The Complexity of Learning Concept

How much data do we need to use a ML algorithm?

Although this is the most common question, it is hard to answer since the amount of data mainly depends on how complex the learning concept is. In Machine Learning (ML), the learning complexity can be broken down into informational and computational complexities. Further, informational complexity considers two aspects, how many training examples are needed (sample complexity) and how fast a learner/model's estimate can converge to the true population parameters (rate of convergence). Computational complexity refers the types of algorithms and the computational resources to extract the learner/model's prediction within a reasonable time. As you can guess now, this blog will cover informational complexity to answer the question.

Learn from an example – 'To be or Not to be banana'

Let's try to learn what banana is. In this example, banana is the learning concept (one hypothesis, that is 'to be' or 'not to be banana'), and the various descriptions associated with banana can be features[i] such as colors and shapes. Unlike the way human can process the concept of banana – the human does not require non-banana information to classify a banana, typical machine learning algorithm requires counter-examples. Although there is One Class Classification (OCC) which has been widely used for outlier or anomaly detection, this is harder than the problem of conventional binary/multi-class classification.

Let's place another concept 'Apple' into this example and make this practice as a binary-class classification. By doing this, we just made the learning concept simpler, 'to be banana = not apple' and 'not to be banana = apple'. This is little counter-intuitive since adding an additional learning concept into a model makes the model simpler: however, OCC basically refers one versus all others, and the number of all other cases are pretty much infinite. This is where we are in terms of ML; one of the simplest learning activities for human is the most difficult problem to solve in ML. Before generating some data for banana, we need to define some terms.

- Instances[ii] $\boldsymbol{X}$ describes banana with features such as color ($\boldsymbol{f_1}$ = yellow, green or red, $|\boldsymbol{f_1}|=3$), shape ($\boldsymbol{f_2}$ = cylinder or sphere, $|\boldsymbol{f_2}|=2$) and class label ($\boldsymbol{C} \rightarrow$ {banana, apple}, $|\boldsymbol{C}|=2$). These values for color and shape need to be enumerated. For examples, we can assign integers to each value like (Yellow=1, Green=2, Red=3), (Cylinder=1, Sphere=2) and (banana=0, apple=1) (**Table 1**)
- The target function $\boldsymbol{t}$ generates a prediction for 'is this banana or apple' as a number ranging between $0 \leq \boldsymbol{t}(\boldsymbol{x_i}) \leq 1$. Typically, we want to have a prediction, $\boldsymbol{t}(\boldsymbol{x_i})$ as close as $\boldsymbol{c}(\boldsymbol{x_i})$, $0 \leq \boldsymbol{i} \leq \boldsymbol{n}$, where $\boldsymbol{n}$ is the total number of samples.
- The hypothesis space $\boldsymbol{H}$ can be defined as the conjunction of features and target function $\boldsymbol{h}(\boldsymbol{x_i}) = (\boldsymbol{f_{1i}}, \boldsymbol{f_{2i}}, \boldsymbol{t}(\boldsymbol{x_i}))$.
- Training examples $\boldsymbol{S}$ must contain roughly the same number of banana (0) and apple (1) examples. A sample is described as $\boldsymbol{s}(\boldsymbol{x_i}) = (\boldsymbol{f_{1i}}, \boldsymbol{f_{2i}}, \boldsymbol{c}(\boldsymbol{x_i}))$

Sample complexity – estimate the size of training data set in a quick and dirty way

Ideally, we want to have all the instances in the training sample set $\boldsymbol{S}$ covering all the possible combinations of features with respect to $\boldsymbol{t}$ as you can see in **Table 1**. There are three possible values for $\boldsymbol{f_1}$ and two possible values for $\boldsymbol{f_2}$. Also, there are two classes in this example. Therefore, the number of all the possible instances $|\boldsymbol{X}| = |\boldsymbol{f_1}| \times |\boldsymbol{f_2}| \times |\boldsymbol{C}| = 3 \times 2 \times 2 = 12$. However, $\boldsymbol{f_2}$ is a lucky feature[iii] that is mutually exclusive between banana and apple. Hence, $|\boldsymbol{f_2}|$ is considered as 1 in this case. In addition to that, we can subtract one case because there is no red banana. For this example, only 5 instances can exhaust the entire sample space $\boldsymbol{H}$. In general, the number of features (columns) in a data set is exponentially proportional to the required number of training samples ($|\boldsymbol{S}| = \boldsymbol{n}$). If we assume that all

features are binary like a simple value of yes or no, then $|\boldsymbol{X}| = 2 \times 2 \times 2 = 2^3$. Two to the power of the number of columns is the minimum $\boldsymbol{n}$ in the simplest case. This example only works when the values in all the features are discrete values. If we use the gradient color values for Color (RGB 256 color pallet ranges from 0 to 16777215 in decimal), the required number of training samples will increas quite significantly because now you need to multiply 16777216 for $\boldsymbol{f_1}$ if all the possible colors exist in $\boldsymbol{H}$.
It is worth noting that the number of instances we calculate here does not always guarantee that a learner/model can converge properly. If you have the number of data equal or below this number, the amount of data is simply too small for the most of algorithm except a ML algorithm evaluating one feature at a time such as a decision tree. As a rough rule of thumb, many statisticians say that a sample size of 30 is large enough. This rule can be applied for a regression based ML algorithm that assumes one smooth linear decision boundary. Although an optimal $\boldsymbol{n}$ could be different on a case-by-case basis, it is not a bad idea to start from $\boldsymbol{n} = |\boldsymbol{X}| \times 30$.

Table 1 Training sample set *S*

Instances	$\boldsymbol{f_1}$ Color (Yellow=1, Green=2, Red=3)	$\boldsymbol{f_2}$ Shape (Cylinder=1, Sphere=2)	$\boldsymbol{C_i}$ Class = Banana (0) or Apple (1)
$\boldsymbol{x_1}$, Banana 1	1	1	0
$\boldsymbol{x_2}$, Banana 2	2	1	0
$\boldsymbol{x_3}$, Apple 1	1^{iv}	2	1
$\boldsymbol{x_4}$, Apple 2	2	2	1
$\boldsymbol{x_5}$, Apple 3	3	2	1

Learning curve – an accurate way to estimate the size of training data set

In ML, learning curve refers a plot of the classification accuracy against the training set size. This is not an estimation method; it requires building a classification model multiple times with the different size of training data set. This is a good technique for sanity-checks (underfitting – high bias and overfitting – high variance) for a model. It also can be utilized to optimize/improve the performance.
Figure 1 shows two examples of learning curves that represent underfitting (left-side plot) and overfitting (right-side plot). Basically, these are the plots of the training and cross-validation error (root mean square error in this example) when the size of training set increases. For underfitting case (left-side plot), both the training and cross-validation errors are very high, and the increment of training set size does not help. This indicates that the features in the data set are not quite relevant to the target learning concept. The examples are confusing the model. On the other hand, the right-side plot shows an overfitting case. The validation error is much higher than the training error in this case. The model trains on the identical samples repeatedly, and the training error climbs continuously while the cross-validation error continue to decrease. The performance for an overfitted model usually looks good; however, it will fail miserably for the real-life data.

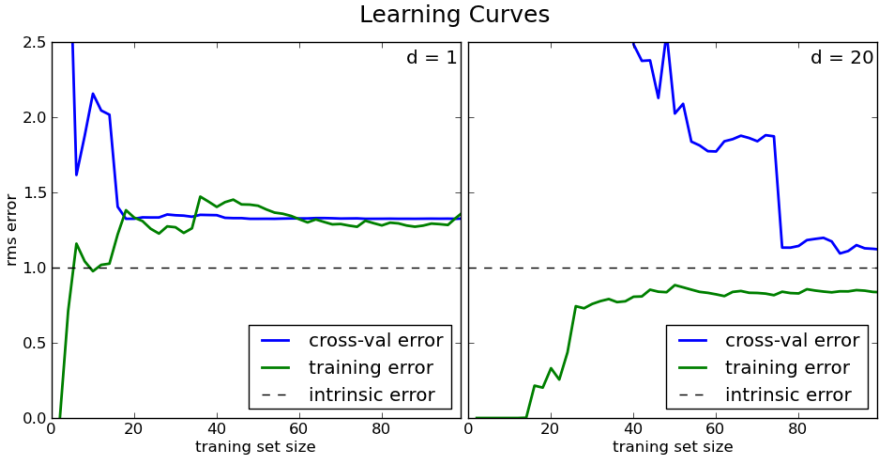

Figure 1 An example of bias plot:
https://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/tutorial/astronomy/practical.html

Back in the day, not a single ML paper was accepted without a learning curve. Without this simple plot, the entire performance claim will be unverifiable.

Resources

Internal web page
External web page

Contacts

Americas

Kihoon Yoon

Sr. Principal Systems Dev Eng

Kihoon.Yoon@dell.com

+1 512 728 4191

---

[i] Or attributes. A feature is an individual measurable property or characteristic of a phenomenon being observed.
[ii] Instance is also referred as example or sample
[iii] If a feature like f_2 exists in a data set, we could make 100% accurate prediction simply by looking at f_2.
[iv] Is there yellow apple? Yes, Golden Delicious is yellow.