Computational Complexity of ML Models

Paritosh Kumar
Dec 15, 2019 · 3 min read

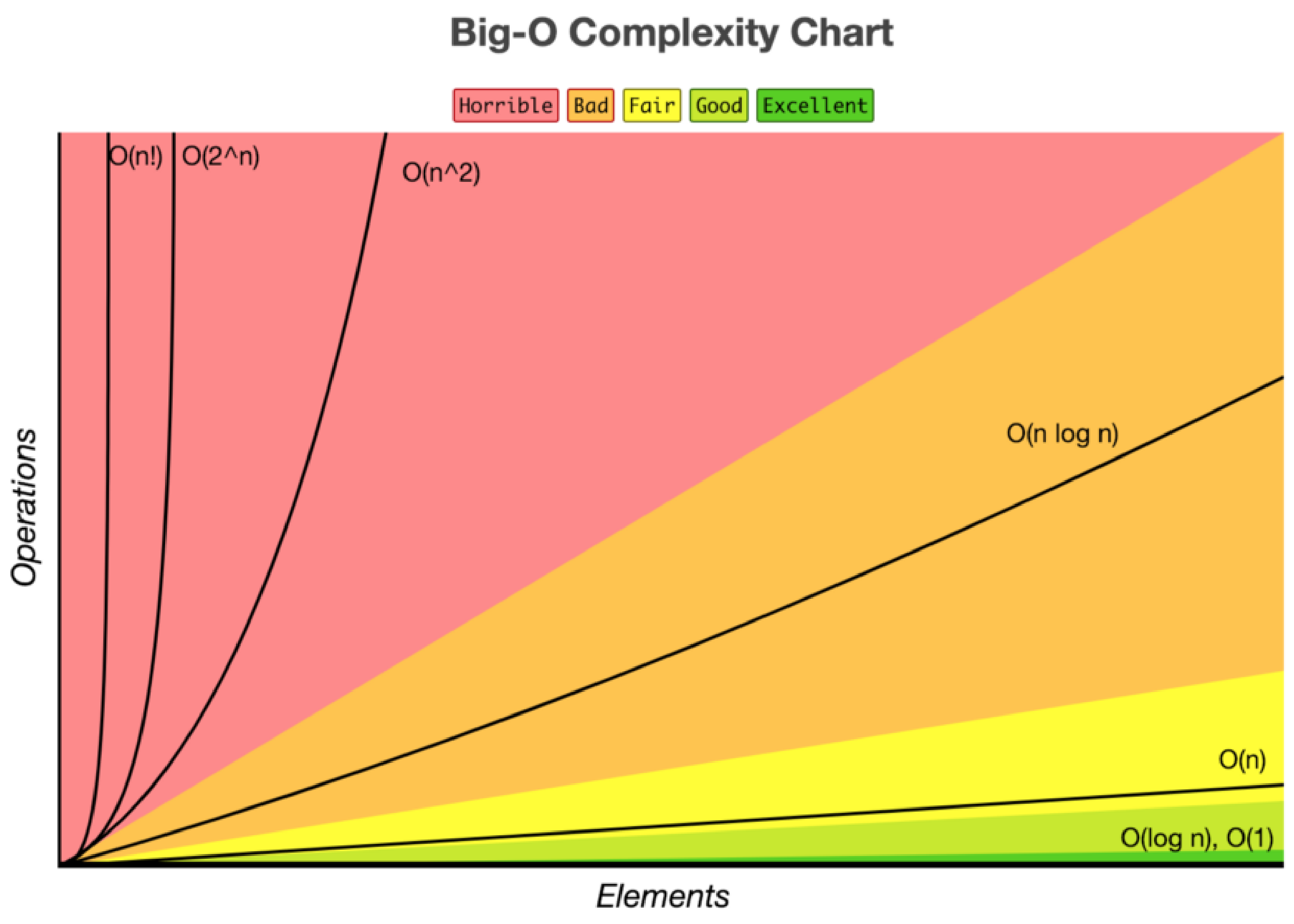
If you ever face a scenario like this, Congrats it means you have huge data :D :D. Knowing the Computational complexity is very important in Machine Learning.

So my topic is, **"What are the computational complexities of ML Models"**

Time complexity can be seen as the measure of how fast or slow an algorithm will perform for the input size. Time complexity is always given with respect to some input size (say n).
Space complexity can be seen as the amount of extra memory you require to execute

your algorithm. Like the time complexity, it is also given with respect to some input size (n).

The complexity of an algorithm/model is often expressed using the **Big O Notation,** which defines an upper bound of an algorithm, it bounds a function only from above. The graph below visualizes different cases of complexities for algorithms.



http://bigocheatsheet.com/

To write computation complexity we are assuming as,
n= number of training examples, d= number of dimensions of the data,
k= number of neighbors

The complexity of K Nearest Neighbors to find the k closest neighbor

Train Time Complexity *= O(knd)*
Loops through every training observation and computes the distance *d* between the training set observation and new observation.

Time is linear with respect to the number of instances (n) and dimensions (d).

Space Complexity *= O(nd)*
K Nearest Neighbors store the data.
Testing takes longer because you have to compare every test instance to the whole training data.

The complexity of Logistic Regression

Training Time Complexity means in logistic regression, it means solving the optimization problem.
Train Time Complexity=*O(nd)*

Space Complexity *= O(d)*
Note: Logistic regression is very good for low latency applications.

The complexity of SVM

Training Time Complexity=O(n^2)
Note: if n is large, avoid using SVM.

Run-time Complexity= O(k*d)
K= number of Support Vectors,d=dimentionality of the data

The complexity of Decision Tree

Training Time Complexity= O(n*log(n)*d)
n= number of points in the Training set
d=dimentionality of the data

Run-time Complexity= O(maximum depth of the tree)
Note: We use Decision Tree when we have large data with low dimentionality.

The complexity of Random Forest

Training Time Complexity= O(n*log(n)*d*k)
k=number of Decision Trees
Notes: When we have a large number of data with reasonable features. Then we can use multi-core to parallelize our model to train different Decision Trees.

Run-time Complexity= O(depth of tree* k)
Space Complexity= O(depth of tree *k)

Note: Random Forest is comparatively faster than other algorithms.

> ***The complexity of Naive Bayes***

Training Time Complexity = O(n*d)
Run-time Complexity = O(c*d)
We have to retrieve feature for each classes 'c'

Conclusion:
If you have a large amount of data , choosing the algorithm is based on the business problem that you are going to solve. Try to reduce the dimension of the data if required to decrease the computation complexities.
If you guys have any doubts please feel free to comment down below.

References :

https://www.appliedaicourse.com/
https://stats.stackexchange.com/questions/96995/machine-learning-classifiers-big-o-or-complexity

Algorithms Data Science Machine Learning Analytics Artificial Intelligence