**Curse of Dimensionality in Machine Learning**

Hello all, this is my first attempt at writing a technical blog and please excuse me if you find it a little vague. I will try my best to write a precise and simple blog to understand. So, let’s get started without wasting too much time!

***What is Curse of Dimensionality — Intuition behind it.***

First Question: What are *dimensions* in the dataset? So, each *feature or column*in dataset is considered as a dimension. Hence, if we have n features in dataset, then it means we have n-dimensional data in our hands. Very easy!

Now, whenever we talk about machine learning, the first thing that comes to mind is, how *insightful* our data is ! The more information our data can provide, the better will be the performance of the model in building and testing on real time data. So, this leads us to think that more dimensions or features in the dataset means more information and that means model will be quite accurate in this scenario since it has more information! But here’s the catch. It is **not** always the case that **more features=>more information=>more accurate model**. The analogy “more features=>more information=>more accurate model**”**holds true only until a certain *threshold.*We can imagine this threshold as a deadline till which we can increase the number of features. If we increase the number of features beyond this threshold value, our model starts suffering drastically. So, how exactly can we explain this scenario of less accurate model when we provide model with more features than threshold? Let’s dive deep.

Let’s say we select 2 features in the dataset and we build a model using those 2 features. Let’s say we get accuracy as *a1 on test data. (I have considered accuracy as a parameter to compare different models. However, you can select any other metric as parameter).* Consider I increase the number of features from 2 to 3, 3 to 6, 6 to 12 iteratively and calculate accuracies each time I select different number of features. Let’s say we get accuracies a2, a3 and a4 respectively on same set of test data. What we can observe is exactly this **a3>a2.>a1.**However, **a4<a3**. This is what curse of dimensionality exactly is! When we took 6 features and calculated accuracy, it was the highest at that time because we have reached a *threshold (discussed in earlier paragraph)*value of number of features and after this point, accuracies will start to go down horribly. However, the most obvious question you can get is, **HOW?**The next paragraph tries to answer this everlasting question! (Please note that number of features selected as threshold value changes each time depending on the dataset provided to the model. It is not a static value and I have considered a hypothetical dataset and threshold as 6 to explain things in simpler way.)

Accuracy of model increases till we increase features at certain threshold because model is able to gain information from data provided **till that point only.**However, beyond this point, when we increase features **exponentially**, model misbehaves and basically, it is unable to process that much information to generalize some hypothesis based on the data provided. This can be due to the redundant and irrelevant features that we provided to the model, which do not add value to the model predictions and generalization. Uff, too much theoretical? Let’s see practical implications of curse of dimensionality on one of the famous supervised learning algorithm, K-Nearest Neighbors (KNN).

**Curse of Dimensionality on KNN Algorithm:**

You will be aware of the fact that, KNN works on principle of distance metrics and K number of close neighbors, which are calculated based on distance metrics used. (Euclidean, Manhattan or Minkowski distance). Let’ say we have 2-dimensional data (You know what I mean when I say 2-dimensional data!). Suppose we have provided this 2D training data to the KNN model and then we provide model with a test point. KNN comes in action and labels this data point based on majority of K nearest neighbors. It calculates Euclidean distance between test data point and training points to perform such classification. We know Euclidean distance between 2 points in 2D space is given by *distance ((x1, y1), (x2, y2)) = √ (x2 — x1) ² + (y2–y1) ².*KNN will perform quite well and will classify data point almost perfectly as dimensions of data are only 2. Now, if we add one more dimension and make our data 3 dimensional, Euclidean distance will be then given as ,

*distance ((x1, y1, z1), (x2, y2, z2)) = √ (x2 — x1) ² + (y 2— y1) ² + (z2 — z1) ².*

It is quite obvious from above formula that distance calculated in 3-dimensional feature space will be higher than 2-dimensional feature space. The point to be noted here is that, we haven’t changed the number of training data examples fed to the KNN model as compared to 2-dimensional feature space and we have added only one dimension. Now, if we keep adding more and more dimensions of features to the dataset without increasing number of training data examples, Euclidean distance will increase drastically between test data point and training data points. This will result in sparsity (a lot of empty space in feature space) of training data points. Due to this sparsity, we can’t find TRUE neighbors of test data point because all training data points will be more or less at same distance from test data point. Hence, the concept of small distances from training examples, which is the backbone of KNN algorithm will suffer and this will result in very less accuracies on test data. This is how curse of dimensionality acts as a real CURSE to the model.

Now, what will be the workaround to avoid curse of dimensionality? Our general sense prevails here and we can obviously say, JUST ADD MORE NUMBER OF TRAINING EXAMPLES SO THAT FEATURE SPACE DON’T REMAIN SPARSE AND THEN WE CAN GET TRUE NEIGHBORS OF TEST DATA. It’s actually true, but we must remember one fact that data is not that easily available. Here, we are talking about addition of *meaningful and insightful*data, data with almost no outliers and data which can provide information to the model. We can’t pick ANY data at random and feed to the model. We have seen implications of BAD (Redundant and Irrelevant data) data in fourth paragraph above. So, adding more data is a one possibility to avoid curse of dimensionality; however, this is not at all feasible at this point of time.

So, the best approach to avoid curse of dimensionality is **Feature Selection.**We will learn more about it in, may be later at some stage.

Please do let me know, whether you liked this blog or not. Suggestions for improvement are always welcome:) and there is no one who don’t want appreciation of the work put in!!!