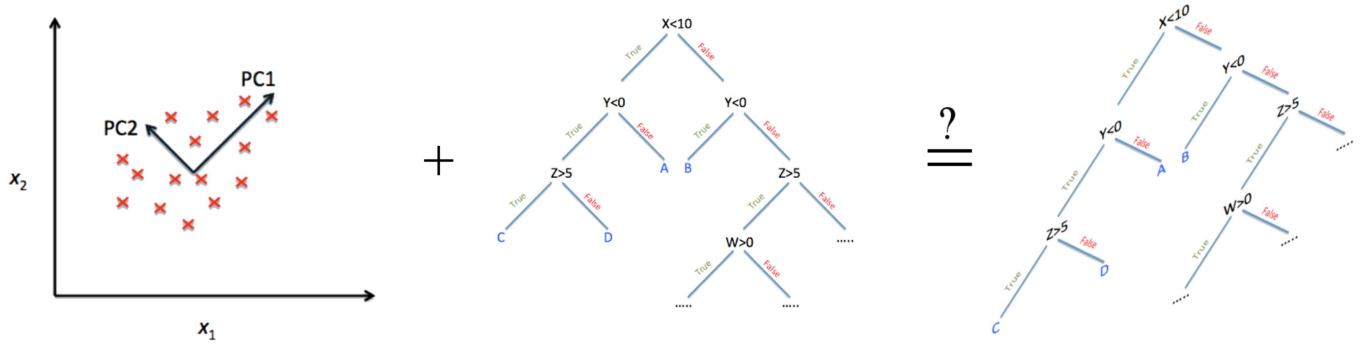
# Why you should use PCA before Decision Trees (https://dorukkilitcioglu.github.io/2018/08/11/pca-decision-tree.html)

4 minute read

Dimensionality Reduction techniques have been consistently useful in Data Science and Machine Learning. It can reduce training times, allow you to remove features that do not hold any predictive value, and it even works for noise reduction. In this blog post, we are going to focus on why it might even make your classifier perform better.

I have known these for quite a while now. However, seeing how much it helps in a low-compute, prototyping environment was a huge wake-up call for me.

Note: this post can be considered a part 1.5 of my Representation Learning series. See part 1 here.




*PCA (https://sebastianraschka.com/faq/docs/lda-vs-pca.html) + Decision Tree (https://medium.com/machine-learning-bites/machine-learning-decision-tree-classifier-9eb67cad263e) ?= Skewed Decision Tree.*

# The Setup

During the first 5 months of the year, I was working on improving various parts of a software solution we are building at NYU IT using Data Science techniques. Some of my work required me to prototype different avenues of possible improvements rapidly and relied heavily on existing Machine Learning models.

Although I usually have an intuition about which models to use on which datasets, I also know not to rely on that intuition too much. As a result, I try to run as many models as possible on any given dataset, but that is often a costly process. Running a tool like polyssifier (https://github.com/alvarouc/polyssifier) is the best way to make sure that you pick the right model for the job, but I prefer choosing only one of each 'class' of models. Ex: you probably do not need to run all of Ridge, Lasso, and Linear Regression (https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/) if you see that Tree Boosting (https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/) performs much better than one of them.

# Principal Component Analysis

Feature (or representation) extraction/learning is also a very important part of this process. Principal Component Analysis (PCA) is one of the most common ways of extracting features and reducing the dimensionality of your dataset.

I am not going to go in-depth and explain the exact workings of PCA. There are countless others on the internet that have already done so, probably better than I ever can. I would personally recommend chapter 11 of Mining of Massive Datasets (http://www.mmds.org/), which is a great resource for working with Big Data.

For our purposes, it is suffice to explain PCA as an algorithm that **linearly rotates** and **scales** a matrix, which is commonly your dataset. If your original dataset is $d$-dimensional, then PCA can give you another $d$-dimensional dataset with **no information loss**. Most importantly, these new dimensions are constructed in a way to **maximize the variance** in a greedy manner, wherein the first dimension has the highest possible variance across all linear transformations of your dataset.

It is this property that allows it to be used as a dimensionality reduction technique. After performing PCA, you can pick the top $k$ dimensions that contains the most variance, which reduces your dimensionality and keeps the highest amount of information (see below). If your data is noisy (ex: anything to do with signals), PCA might also have a noise-removal effect (https://stats.stackexchange.com/questions/247260/principal-component-analysis-eliminate-noise-in-the-data).

It should also be noted here that using PCA is not always a good thing (https://stats.stackexchange.com/questions/52773/what-can-cause-pca-to-worsen-results-of-a-classifier). As usual, you are encouraged to try your models both before and after applying PCA to see how well it performs.

# Information Gain and Decision Trees

Variance goes hand-in-hand with entropy (https://en.wikipedia.org/wiki/Entropy_(information_theory)), which can be considered to be an indication of how much **unexplained information** there is. They are not the same thing, but for many distributions, entropy relies on the variance of the distribution (https://math.stackexchange.com/questions/651077/is-standard-deviation-the-same-as-entropy/651213#651213). Entropy is used in calculating the Information Gain in Decision Trees (http://www.saedsayad.com/decision_tree.htm), which is how Decision Trees are trained.

To put it simply, Information Gain measures the difference in entropy if we were to create a decision point for a given dimension. For a binary feature, Information Gain would be the difference between the entropy of the original dataset, minus the entropy of dataset if we assume that feature is true, and the entropy of the dataset if we assume that feature is false. The original entropy is always larger, as the assumptions simplify the problem, decreasing the number of possibilities.

This makes PCA a natural fit to be applied before a Decision Tree is learned, as it explicitly transforms your dataset to highlight the directions that have the highest variance, which are often the directions that have the highest Information Gain while learning a Decision Tree. This makes the Decision Tree learn much faster, and achieve a high accuracy within the least number of decision layers.

# Conclusion

I would like to end this post with a disclaimer, because as with everything in Data Science and Machine Learning, your mileage may vary. Remember that PCA is simply a **linear transformation**, meaning it does not contain any extra information that could not be found in the original dataset.

Indeed, it is most useful because many learning algorithms are inexact, good-enough algorithms that produce good-enough estimates of the maximum likelihood estimate (https://en.wikipedia.org/wiki/Maximum_likelihood_estimation) of the parameters, which is often intractable and/or computationally infeasible. In theory, you could get the same performance out of a model whether you used PCA beforehand or not. In practice, having a better structured data can make or break a model.

In any case, give PCA a shot. It might be worth your while.

**Updated:** August 11, 2018