DJS Synapse Learning Period

ML Week 3 Resources

Decision Tree:

Theory : https://youtu.be/7VeUPuFGJHk
Theory: https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/
Implementation : https://youtu.be/HY2DcBhgwm0

Overfitting and Underfitting [VERY IMPORTANT]:

https://www.youtube.com/watch?v=T9NtOa-IITo
Note : This concept can be seen in K Means and decision trees. In fact, we will look deeper into this when we begin deep learning.

Ensemble Methods: Boosting (adaboost):

Theory : https://www.youtube.com/watch?v=NLRO1-jp5F8&t=724s
Theory :
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
Implementation : https://www.youtube.com/watch?v=7xHM93WXOu8

Ensemble Methods: Bagging (random forest):

Theory : https://www.youtube.com/watch?v=KIOeZ5cFZ50
Theory :
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
Implementation : https://www.youtube.com/watch?v=MxiktOPmhV8&t=2s

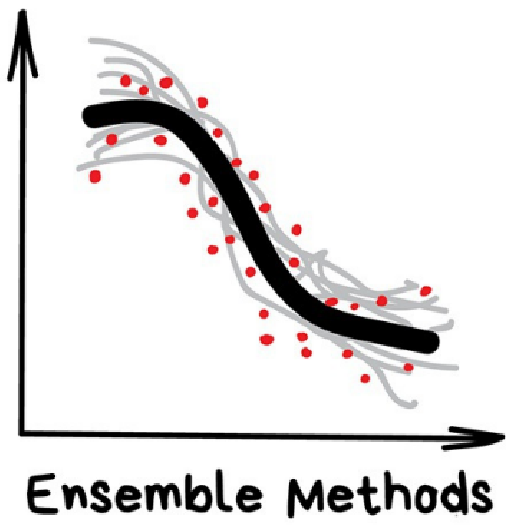
Decision Tree Pruning:

https://towardsdatascience.com/build-better-decision-trees-with-pruning-8f467e73b107

Naive Bayes:

Theory : https://youtu.be/jS1CKhALUBQ
Theory:
https://www.analyticsvidhya.com/blog/2021/09/naive-bayes-algorithm-a-complete-guide-for-data
-science enthusiasts/
Implementation : https://youtu.be/nHIUYwN-5rM

Part 3. Ensemble Methods



"Bunch of stupid trees learning to correct errors of each other"

Nowadays is used for:

- Everything that fits classical algorithm approaches (but works better)
- Search systems (★)
- Computer vision
- Object detection

Popular algorithms: Random Forest, Gradient Boosting

It's time for modern, grown-up methods. Ensembles and neural networks are two main fighters paving our path to a singularity. Today they are producing the most accurate results and are widely used in production.

However, the neural networks got all the hype today, while the words like "boosting" or "bagging" are scarce hipsters on TechCrunch.

Despite all the effectiveness the idea behind these is overly simple. If you take a bunch of inefficient algorithms and force them to correct each other's mistakes, the overall quality of a system will be higher than even the best individual algorithms.

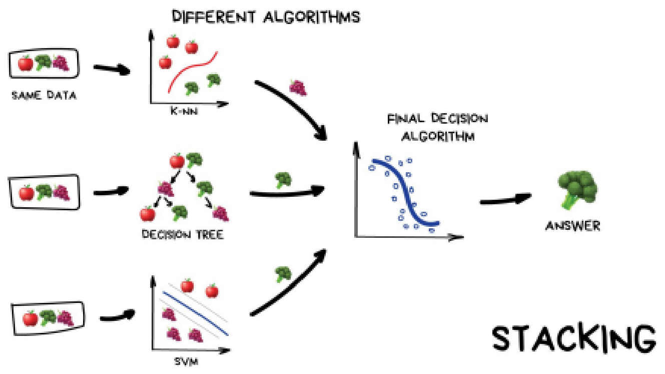
You'll get even better results if you take the most unstable algorithms that are predicting completely different results on small noise in input data. Like Regression and Decision Trees. These algorithms are so sensitive to even a single outlier in input data to have models go mad.

In fact, this is what we need.

We can use any algorithm we know to create an ensemble. Just throw a bunch of classifiers, spice it up with regression and don't forget to measure accuracy. From my experience: don't even try a Bayes or kNN here. Although "dumb", they are really stable. That's boring and predictable. Like your ex.

Instead, there are three battle-tested methods to create ensembles.

Stacking Output of several parallel models is passed as input to the last one which makes a final decision. Like that girl who asks her girlfriends whether to meet with you in order to make the final decision herself.

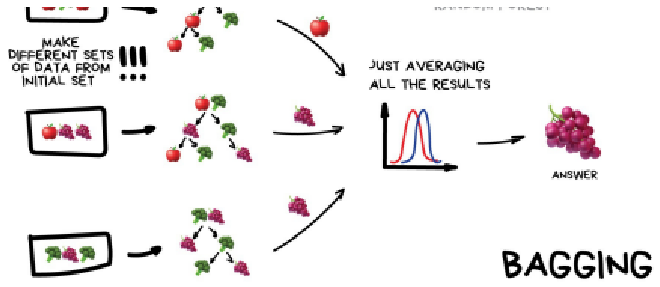


Emphasis here on the word "different". Mixing the same algorithms on the same data would make no sense. The choice of algorithms is completely up to you. However, for final decision-making model, regression is usually a good choice.

Based on my experience stacking is less popular in practice, because two other methods are giving better accuracy.

Bagging aka [Bootstrap AGGregatING](). Use the same algorithm but train it on different subsets of original data. In the end — just average answers.

Data in random subsets may repeat. For example, from a set like "1-2-3" we can get subsets like "2-2-3", "1-2-2", "3-1-2" and so on. We use these new datasets to teach the same algorithm several times and then predict the final answer via simple majority voting.

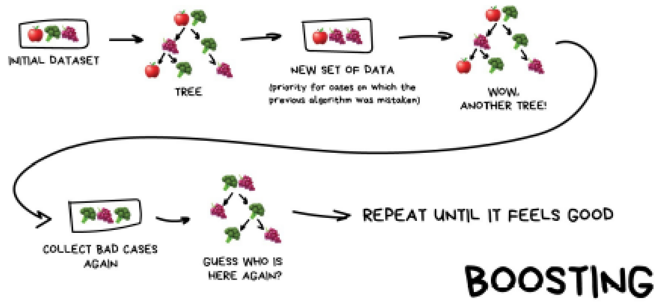

MAKE DIFFERENT SETS OF DATA FROM INITIAL SET
JUST AVERAGING ALL THE RESULTS
ANSWER
BAGGING

The most famous example of bagging is the [Random Forest]() algorithm, which is simply bagging on the decision trees (which were illustrated above). When you open your phone's camera app and see it drawing boxes around people's faces — it's probably the results of Random Forest work. Neural networks would be too slow to run real-time yet bagging is ideal given it can calculate trees on all the shaders of a video card or on these new fancy ML processors.

In some tasks, the ability of the Random Forest to run in parallel is more important than a small loss in accuracy to the boosting, for example. Especially in real-time processing. There is always a trade-off.

Boosting Algorithms are trained one by one sequentially. Each subsequent one paying most of its attention to data points that were mispredicted by the previous one. Repeat until you are happy.

Same as in bagging, we use subsets of our data but this time they are not randomly generated. Now, in each subsample we take a part of the data the previous algorithm failed to process. Thus, we make a new algorithm learn to fix the errors of the previous one.



The main advantage here — a very high, even illegal in some countries precision of classification that all cool kids can envy. The cons were already called out — it doesn't parallelize. But it's still faster than neural networks. It's like a race between a dump truck and a racecar. The truck can do more, but if you want to go fast — take a car.

If you want a real example of boosting — open Facebook or Google and start typing in a search query. Can you hear an army of trees roaring and smashing together to sort results by relevancy? That's because [they are using boosting].