Pros and Cons of Decision Trees and the Nearest Neighbors Method

Decision trees

Pros:

* Generation of clear human-understandable classification rules, e.g. “if age <25 and is interested in motorcycles, deny the loan”. This property is called interpretability of the model.
* Decision trees can be easily visualized, i.e. both the model itself (the tree) and prediction for a certain test object (a path in the tree) can “be interpreted”.
* Fast training and forecasting.
* Small number of model parameters.
* Supports both numerical and categorical features.

Cons:

* The trees are very sensitive to the noise in input data; the whole model could change if the training set is slightly modified (e.g. remove a feature, add some objects). This impairs the interpretability of the model.
* Separating border built by a decision tree has its limitations – it consists of hyperplanes perpendicular to one of the coordinate axes, which is inferior in quality to some other methods, in practice.
* We need to avoid overfitting by pruning, setting a minimum number of samples in each leaf, or defining a maximum depth for the tree. Note that overfitting is an issue for all machine learning methods.
* Instability. Small changes to the data can significantly change the decision tree. This problem is tackled with decision tree ensembles (discussed next time).
* The optimal decision tree search problem is NP-complete. Some heuristics are used in practice such as greedy search for a feature with maximum information gain, but it does not guarantee finding the globally optimal tree.
* Difficulties to support missing values in the data. Friedman estimated that it took about 50% of the code to support gaps in data in CART (an improved version of this algorithm is implemented in sklearn).
* The model can only interpolate but not extrapolate (the same is true for random forests and tree boosting). That is, a decision tree makes constant prediction for the objects that lie beyond the bounding box set by the training set in the feature space. In our example with the yellow and blue balls, it would mean that the model gives the same predictions for all balls with positions >19 or <0.

The nearest neighbors method

Pros:

* Simple implementation;
* Well studied;
* Typically, the method is a good first solution not only for classification or regression, but also recommendations;
* It can be adapted to a certain problem by choosing the right metrics or kernel (in a nutshell, the kernel may set the similarity operation for complex objects such as graphs while keeping the k-NN approach the same). By the way, [Alexander Dyakonov](https://www.kaggle.com/dyakonov), a former top-1 kaggler, loves the simplest k-NN but with the tuned object similarity metric;
* Good interpretability. There are exceptions: if the number of neighbors is large, the interpretability deteriorates (“We did not give him a loan, because he is similar to the 350 clients, of which 70 are the bad, and that is 12% higher than the average for the dataset”).

Cons:

* Method considered fast in comparison with compositions of algorithms, but the number of neighbors used for classification is usually large (100-150) in real life, in which case the algorithm will not operate as fast as a decision tree.
* If a dataset has many variables, it is difficult to find the right weights and to determine which features are not important for classification/regression.
* Dependency on the selected distance metric between the objects. Selecting the Euclidean distance by default is often unfounded. You can find a good solution by grid searching over parameters, but this becomes very time consuming for large datasets.
* There are no theoretical ways to choose the number of neighbors – only grid search (though this is often true for all hyperparameters of all models). In the case of a small number of neighbors, the method is sensitive to outliers, that is, it is inclined to overfit.
* As a rule, it does not work well when there are a lot of features due to the “curse of dimensionality”. Professor Pedro Domingos, a well-known member in the ML community, talks about it [here](https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf) in his popular paper, “A Few Useful Things to Know about Machine Learning”; also “the curse of dimensionality” is described in the Deep Learning book in [this chapter](http://www.deeplearningbook.org/contents/ml.html).