Neural Networks vs. Random Forests – Does it always have to be Deep Learning?

by Prof. Dr. Peter Roßbach

Motivation

After publishing my blog post "Machine Learning, Modern Data Analytics and Artificial Intelligence – What's new?" in October 2017, a user named Franco posted the following comment:

"*Good article. In our experience though (finance), Deep Learning (DL) has a limited impact. With a few exceptions such as trading/language/money laundering, the datasets are too small and DL does not catch the important features.*
For example, a traditional Random Forest (RF) classification algorithm can be beat DL in financial compliance by a huge margin (e.g. >10 points in accuracy and recall).
Additionally, RF can be made "interpretable" which is infinitely useful for the upcoming EU Regulation General Data Protection Regulation."

I would like to take this comment as an impetus for this blog post. Neural Networks and especially Deep Learning are actually very popular and successful in many areas. However, my experience is that Random Forests are not generally inferior to Neural Networks. On the contrary, in my practical projects and applications, Random Forests often outperform Neural Networks.

This leads to two questions:

(1) What is the difference between the two approaches?

(2) In which situations should one prefer Neural Networks or Random Forests?

How do Neural Networks and Random Forests work?

Let's begin with a short description of both approaches. Both can be used for classification and regression purposes. While classification is used when the target to classify is of categorical type, like creditworthy (yes/no) or customer type (e.g. impulsive, discount, loyal), the target for regression problems is of numerical type, like an S&P500 forecast or a prediction of the quantity of sales.

Neural Networks

Neural Networks represent a universal calculation mechanism based on pattern recognition. The idea is to combine simple units to solve complex problems. These units, also called neurons, are usually organized into several layers that have specific roles. The basic architecture of a so-called multi-layer perceptron is shown in Figure 1.

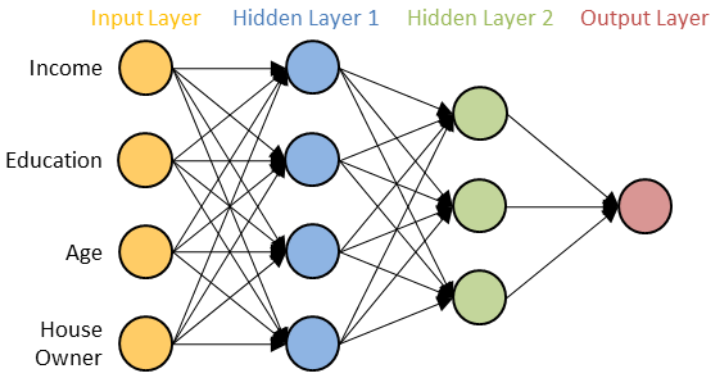

Figure 1: Architecture of a Neural Net

The Neural Network consists of an input and an output layer and in most cases one or more hidden layers that have the task to transform the inputs into something that the output layer can use. Neural Networks can process all kinds of data which is coded in numeric form. The data is inserted into the network via the input layer, transformed via the hidden layer(s) and finally scaled to the wanted outcome in the output layer. In the case of an assessment of creditworthiness, the input neurons would, for example, take values for income, education, age and house ownership. The output neuron would then provide the probability for creditworthiness.

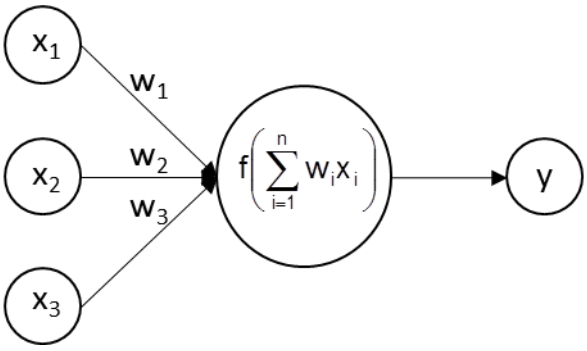

Figure 2: Functioning of a neuron

If we zoom into a hidden or output node, we see what is shown in figure 2. The node receives the outputs x_i of the previous nodes to which it is connected. These are multiplied by weights w_i and summed up. The sum is then transformed with a function f() and passed on to the nodes of the next layer or output as a result. After setting the architecture of a Neural Network (number of layers, number of neurons per layer and transformation function f() per neuron) the network is trained by searching for the weights that produce the desired output. Learning algorithms adjust the connection weights between the neurons according to minimize the (squared) differences between the real values of the target variables and those calculated by the network.

Thus, a Neural Network is a chain of trainable, numerical transformations that are applied to a set of input data and yield certain output data. With this very general paradigm we can build nearly anything: Image classification systems, speech recognition engines, trading systems, fraud detection systems, and so on.

A Neural Network can be made deeper by increasing the number of hidden layers. The more layers the more complex the representation of an application area can be. Deep Networks have thousands to a few million neurons and millions of connections.

Over time different variants of Neural Networks have been developed for specific application areas. Specifically for image processing, Convolutional Neural Networks were developed. Recurrent Neural Networks are designed to recognize patterns in sequences of data and Long Short-Term Memory Networks as a further development to learn long-term dependencies.

Random Forests

Random Forests belong to the family of decision tree algorithms. A Decision Tree represents a classification or regression model in a tree structure. Each node in the tree represents a feature from the input space, each branch a decision and each leaf at the end of a branch the corresponding output value (see figure 3).

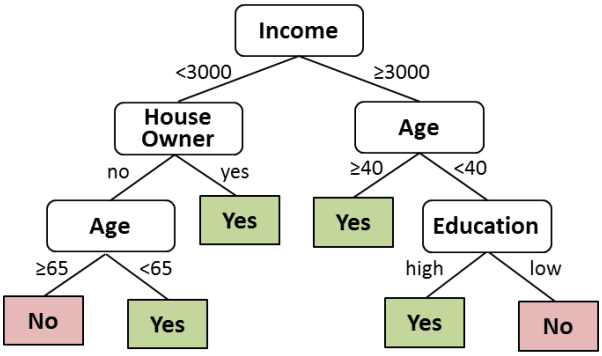

Figure 3: Architecture of a Decision Tree

To obtain a result for a specific input object, e.g. a person who applies for a credit, the decision process starts from the root node and walks through the tree until a leaf is reached which contains the result. At each node, the path to be followed depends on the value of the feature for the specific input object. In figure 3 for example the process walks to the left, if the person has an income lower than 3000.

Similar to Neural Networks, the tree is built via a learning process using training data. The learning process creates the tree step by step according to the importance of the input features in the context of the specific application. Using all training data objects, at first the most important feature is identified by comparing all of the features using a statistical measure. According to the resulting splitting value (3000 for income in figure 3), the training data is subdivided. For every resulting subset the second most important feature is identified and a new split is created. The chosen features can be different for every subset (see left and right branch in figure 3). The process is now repeated on each resulting subset until the leaf nodes in all the branches of the tree are found.

A Decision Tree is easy to create, is able to handle different types of input data (categorical and numerical) and is interpretable due to its kind of representation. However, Decision Trees often show a lack of reliability in their application to new data. One reason for this is their tendency to perfectly fit all samples in the training data. This results in poor application quality if the data is noisy or contains outliers.

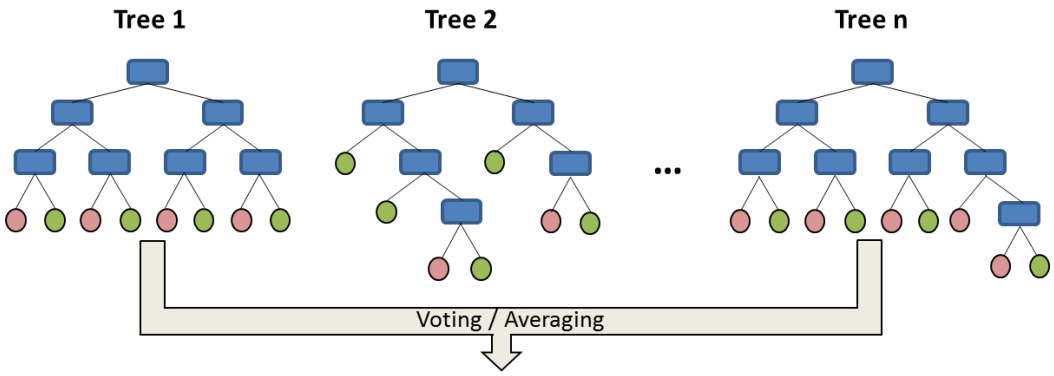

Figure 4: Random Forest as Tree Ensemble

A so-called Random Forest is the answer to this problem. By definition, a forest is a group of single trees. Every tree is different; they all together make a forest. According to this analogy, the Random Forest algorithm builds different trees based on the same training data source. Each Decision Tree is created with a different, randomly chosen subset of the training data and with a randomly chosen subset of the features at every node. Thus, the trees are always only built on the basis of a subset of the data.

Compared to the classical Decision Tree explained above, every tree in a Random Forest is a weak tree because he did not have access to the complete data during the build process. But all together, they build an ensemble of “decision makers” where the collective result can be created by majority voting in the case of classification or averaging in the case of regression (see figure 4). This compensates for potential errors of single trees in the forest so that the model is less likely to produce results further away from the real values. Many of the applications of Random Forests use tree sizes between 100 and 500. In their practical application they have proven to be of very high quality.

An important strength of Random Forests is that they are able to perform still well in the case of missing data. According to their construction principle, not every tree is using the same features. If there is any missing value for a feature during the application there usually are enough trees remaining that do not use this feature to produce accurate predictions.

On the other hand, when applied to regression problems, Random Forests have the limitation that they cannot exceed the range of values of the target variable used in training [1]. Thus, Random Forests may perform poorly with data that is out of the range of the original training data.

When to choose which algorithm?

With Neural Networks and Random Forests, we have two approaches which have the potential to produce classification and regression models of high quality. In practice, both approaches are widely and successfully used in different application areas. This raises the question, which of the methods (among all the others available in machine learning) one should prefer when starting a particular project.

An important finding in machine learning is that no single algorithm works best across all possible scenarios. Thus, no algorithm strictly dominates in all applications; the performance of machine learning algorithms varies wildly depending, for example, on the application and the dimensionality of the dataset. Accordingly, a good practice is to compare the performance of different learning

algorithms to find the best one for the particular problem. In some cases, it is also advantageous to build ensembles of multiple models created with different methods to combine strength and eliminate weaknesses.

However, often there is not enough time and/or money to test and optimize every algorithm in order to its quality in a specific context. On the other hand, particular weaknesses of an approach can lead to avoid a specific algorithm in a specific context. In these cases, a decision about an algorithm has to be made before starting the project.

In the following, I want to discuss some criteria which are important when choosing an algorithm. According to the subject of this blog, I will concentrate my discussion solely on Neural Networks and Random Forests.

Which criteria are important when choosing an algorithm?

Performance: Performance in the sense of the quality of classification or regression is a major determinant of the selection of an algorithm. An algorithm must be able to capture the underlying functionality and to represent it in the resulting model. Algorithms that can only capture linear relationships, for example, are always unsuitable if there are predominantly non-linear relationships in the underlying application area. Both, Neural Networks and Random Forests, have the ability to model linear as well as complex nonlinear relationships. Due to their construction, Neural Networks have a higher potential here.

Robustness: When assessing performance, the quality and reliability of the application must play the decisive role rather than the quality of fitting when creating the model. A model that performs very well with the data used to create it does not necessarily perform equally well with new data. Such a situation is called “overfitting”. In this case, the respective model does not provide the appropriate robustness which reflects its generalizability.

In times of Big Data, we often have to deal with high volumes of high-dimensional data coming from different sources. In contrast to the past, in which we mainly worked with pre-processed and aggregated data, we often use granular data in much larger quantities today. We are often not able to totally control the quality of the data. As a consequence, we must be aware that the data is not fully representative and may contain some errors and outliers.

A very similar situation arises for applications in a dynamic environment, in which at any time and often by chance (mostly slight) changes in the relationships can occur, for example the correlations between stock returns. This is very typical for economic applications. From a technical point of view, such changes lead to slightly different outputs with the same inputs. A model should react to this stably and not too sensitively.

As a consequence, the algorithm used to estimate the model must be able to find the underlying relationships and regularities in the given data and not try to fit the data as well as possible. One way to force this is to reduce the set of possible models that can be constructed by the algorithm. In literature, the term “complexity” is often used in this context. Simplified, we can describe this as the possible outputs that a model can generate in relation to the possible inputs. A linear model has therefore a lower complexity because it cannot represent the same output space compared to a nonlinear model. Thus, a linear model applied to a nonlinear case results in underfitting while a

nonlinear model applied to a linear case contains the risk of overfitting because it is able to produce output that deviates from a linear world.

A promising strategy in Machine Learning is to search for a model having a complexity corresponding to the complexity of the area of application. In this case, we reduce the risk of overfitting. This can be done by reducing the set of possible models that can be created by the respective algorithm. To do this, both approaches, Neural Networks and Random Forests, offer different opportunities. In Neural Networks, for example, the number of hidden neurons and the number of their layers have an influence on the complexity of the models or regularization techniques can be used when optimizing the weights during the learning process. In Random Forests, one can adjust the number of trees or the maximal size or depth of the single trees, for example. Thus, both approaches provide opportunities to handle complexity and overfitting issues. In comparison, Neural Networks are attributed to have a higher sensitivity to inputs, which tends to result in a higher risk of deviations in the case of “atypical” inputs.

Comprehensibility: Franco already stated in his comment, that models and their results should be “interpretable”. Humans have to understand, what models are doing, especially when they are responsible for the consequences of their application. In sectors like the finance area additionally regulation forces the companies to use algorithms that are transparent in order to understand their underlying risks. One of the main problems of Neural Networks is that trained models are difficult to interpret. They consist of hundreds to millions of different parameters depending on the size of the network, all interacting in a complex way. This black box problem makes it complicated to use Neural Networks in areas where trustiness and reliability of the predictions are of great importance.

Random Forests are also difficult to interpret because they consist of many (usually hundreds) of individual trees. Even if a single tree is easy to understand, the large number of trees makes the ensemble difficult to understand. More recently, however, approaches have been developed to identify the most representative trees in an ensemble [2]. By means of their analysis, the ensemble can finally be interpreted.

Cost and Time Expenditure: From an economic point of view, the costs and time required for the creation of a model play an important role. In this context, the training of Neural Networks is very time-consuming and computationally intensive. In addition to the learning process itself, a large amount of preparatory work is also necessary to bring the inputs into the required form. They must be in number format and normalized, for example. To find the best model, several variants have to be calculated and tested. Different so-called hyperparameters can be varied, e.g. the number of layers or neurons per layer or the learning rate. Different combinations of these hyperparameters are tested in a so-called grid search. For every combination a complete model has to be calculated and evaluated. The more hyperparameters exist, the more combinations have to be tested. In combination with the required time for training a model, this results in a considerable expenditure of time and cost.

Random Forests require much less input preparation. They can handle binary features, categorical features as well as numerical features and there is no need for feature normalization. Random Forests are quick to train and to optimize according to their hyperparameters [3]. Thus, the computational cost and time of training a Random Forest are comparatively low.

Furthermore, a Random Forest can be trained with a relative small amount of data. Neural Networks usually need more data to achieve the same level of accuracy. On the other hand, Random Forests

often have little performance gain when a certain amount of data is reached, while Neural Networks usually benefit from large amounts of data and continuously improve the accuracy.

Empirical Comparisons of Neural Networks and Random Forests

After discussing the differences between Neural Networks and Random Forests it makes sense to take a look into empirical studies comparing the performance of both. There exist many studies with this subject in several domains. I have selected three studies for this blog.

A very large study compares 179 classifiers applied to the 121 classification data sets of the UCI repository, which is one of the reference data bases for machine learning. According to the results, Random Forest is clearly the best classifier as it achieves the best classification results in over 90% of the cases [4]. The results of Neural Networks are on average worse, but close to those of Random Forests. One additional finding is that Neural Networks are getting better when the complexity of the data set increases.

Another study is concerned with the comparison of Neural Networks and Random Forests in predicting building energy consumption [3], which is a numerical prediction and not a classification case. According to the findings, Neural Networks performed marginally better than Random Forests. But the Random Forest models were able to effectively handle any missing values in the application. Thus, the Random Forests were able to accurately predict even when some of the input values were missing.

A further study deals with the prediction of carbon and nitrogen in the soil in agriculture, which is a nonlinear regression problem [5]. The study compares Neural Networks, Random Forests and another ensemble based tree algorithm called Gradient Boosted Machines. The results show, that the performance and the rank of the three methods varied according to the dataset used. In the majority of the cases the Random Forest models outperformed the other models.

Even if the above description of comparative studies is not complete, it shows that both approaches have similar performance potential in many applications. Other studies show similar results, sometimes Neural Networks are better sometimes Random Forests, but mostly only with minor differences.

Nevertheless, deep learning has been able to produce a huge improvement in quality in many, mostly very complex areas. This includes image classification, speech recognition, etc. In these areas Random Forests do not have the potential to compete.

Summary

The intention of this blog was to show that Neural Networks, despite their current high visibility in the media, not always need to be the first choice in selecting a machine learning methodology. Random Forests not only achieve (at least) similarly good performance results in practical application in many areas, they also have some advantages compared to Neural Networks in specific cases. This includes their robustness as well as benefits in cost and time. They are particularly advantageous in terms of interpretability. If we were faced with the choice of taking a model with 91% accuracy that we understand or a model with 93% accuracy that we don't currently understand, we would probably choose the first one for many applications, for example, if the model is supposed to be

responsible for investigating patients and suggesting medical treatment. These may be the reasons for the increasing popularity of Random Forests in practice.

Literature

[1] Ellis, P. (2016): Extrapolation is tough for trees! http://freerangestats.info/blog/2016/12/10/extrapolation

[2] Banerjee, M.; Ding, Y.; Noone, A. (2012): Identifying representative trees from ensembles. In: Statistics in medicine, volume 31, pp. 1601-1616.

[3] Ahmad, M.W.; Mourshed, M.; Rezgui, Y. (2017):Trees vs Neurons: Comparison between Random Forest and ANN for high-resolution prediction of building energy consumption. In: Energy and Buildings, volume 147, pp. 77–89.

[4] Fernández-Delgado, M.; Cernadas, E.; Barro, S. (2014): Do we Need Hundreds of Classifers to Solve Real World Classifcation Problems? In: Journal of Machine Learning Research, volume 15, pp. 3133-3181.

[5] Said Nawar, S.; Mouazen, A.M. (2017): Comparison between Random Forests, Artificial Neural Networks and Gradient Boosted Machines Methods of On-Line Vis-NIR Spectroscopy Measurements of Soil Total Nitrogen and Total Carbon. In: Sensor, volume 17, pp. 2428-2450.