Klara Schlüter
Machine Learning
NTNU - Department of Computer Science

November 6, 2019

ASSIGNMENT 3 — Theory

1 Deep vs. Shallow Learning

Deep Learning architectures are characterized by having several hidden layers, they are a "multilayer stack of simple modules" [1]. The architecture of the layers thereselves is simple. Every single layer represents a function, mapping the layer's input to a certain output. The layers are mostly non-linear [1], which means, that the represented function from input to output is non-linear. All layers are a subject of the learning process (are transformed by the learning process to improve the overall performance). Deep Learning architectures are able to learn from and process complex, high-dimensional data.

Deep Learning methods perform representation learning: Deep Learning systems are able to process raw data, as opposed to Shallow Learning systems, that only work when given only relevant features of the data (a good representation). The multilayered architecture in Deep Learning enables the system to discover those relevant features itself. During learning, the system abstracts from the raw input more and more layer by layer. Every layer represents the data in its own way, so the system as a whole finds the features (representations) that are important for the task. An example for Deep Learning would be a deep neural network accepting images and classifying them into "contains a dog" and "does not contain a dog". The network is fed with the raw image and has to find out how to detect dogs in it, instead of being given features like "picture contains two ears" and "picture contains for legs".

Shallow learning on the other hand requires preprocessed data. The relevant features have to be extracted beforehand to provide a good representation of the data to the system. An example would be linear regression as we implemented it in the first assignment: the system is fed with data represented as points in a coordinate system with axes representing only the relevant features. Given this representation, linear regression learns to classify the data, but it is not capable of discovering the representation itself.

2 Comparison k-NN, decision tree, SVM and deep learning

In the following, I am going to compare the four machine learning methodologies of the k-NN algorithm, decision tree learning, Support Vector Machines and Deep Learning. After contrasting different characteristics, I am going to discuss strengths and weaknesses of the approach as well as the question when and when not to apply them.

In table 1, four simple properties are summarized: As explained in the previous exercise, Machine learning can be **deep** or **shallow**, where (as the name already says) Deep Learning is the only deep one of the compared approaches. The other approaches don't work in multiple layers and need data in a certain (feature-extracted) form. Another classification can be done in **lazy** and **eager** learning. In the k-NN approach, the training examples are simply stored, and only when there is a query, a target function is computed for this special query. k-NN therefore is a lazy learner. The other three methods compute a target function while training and therefore are eager learners. Furthermore, we can examine the methodologies for their **robustness** to noisy training data (see table 1).

Comparing what type of **target functions** can be approximated, we can see that k-NN and Deep Learning are able to approximate discrete and continuous target functions (are able to do classification as well as regression). Decision trees are by the definition of a tree made

properties	**k-NN**	**decision tree**	**SVM**	**deep learning**
DEEP/SHALLOW [1]	shallow	shallow	shallow	deep
LAZY/EAGER	lazy	eager	eager	eager
ROBUST *to noisy training data*	yes [2]	yes	no [slides]	yes

Figure 1: Properties of machine learning approaches

for discrete functions. SVM is the most limited approach in matters of target functions: it learns a two-valued (discrete) classification [2]. In decision tree learning, the learned tree represents the target function, as well as the learned architecture in deep learning.

In the first chapters of the textbook [2], the **inductive bias** of a machine learning method is described. In k-NN, this bias is the idea that the solution of a problem must be similar to the solution of similar problems [2]. Both SVM and decision tree learning implement an inductive bias according to Occam's razor: in SVM, fewer support vectors (which means a simpler representation of the hyperplane) [lecture slides] are prefered, while in decision trees, smaller trees are searched first.

While Deep Learning can process raw data, the three shallow learning methods require a specific **input** representation. For decision tree learning, this is a data basa like set of instances with a fixed set of attributes of fixed (but possibly different) types. Both SVM and k-NN require the instances to be represented as points in the d-dimensional space $\mathbb{R}^n$ [2].

In the following, I discuss when and when not to use the four examined methodologies. First of all, two of the described characteristics result in very concrete limitations: the target function and the data representation. For example, decision trees can only be used to approximate discrete target functions and if the data is reprented as instances with attributes as described earlier. Also, the properties seen in table 1 should be consulted when choosing the best approach. Still, each method has its strengths and weaknesses, which are described in the following.

In k-NN, a new approximation of the target function is computed for every query [2]. This can be strength and weakness: the approximation can take special aspects of the query into account. However, if the target function is expensive to compute (if aggregating the neighbours solutions is expensive), k-NN isn't the best choice. In k-NN, all attributes are used to find similar instances. If not all attributes are relevant, the so-called *curse of dimensionality* prevents the system from performing well. The idea of k-NN is to use similar solutions for similar problems, so it should only be applied to a problem space where this holds. Finally, k-NN is extremly robust to noisy training data [2], and should therefore be considered when the given data contains a lot of missing values or errors.

Since the computation done for SVM uses only the support vectors, it is a good choice for large feature sets or large data sets. It does not depend on every single detail and therefore might perform faster than other approaches. However, it is extremely sensible to noisy data and therefore shouldn't be used if clean data is not guaranteed.

The speciality of decision tree learning is that per definition of a decision tree, it represents disjunctive descriptions, so it is the perfect option for such problems.

Deep learning is the only of the four approaches doing representation learning and therefore the best choice for tasks where the problem representation can't or shouldn't be prespecified, for example because it is computationally infeasible. Also, Deep Learning is perfect for complex tasks that the other approaches are not able to solve. Especially, it is good in finding new and better types of solutions, since it does not depend on any hints on how to solve a problem. Still, the resulting architecture is more complex than in the other approaches, which sometimes isn't necessary.

3 Ensemble methods

Ensemble methods can be used both for classification and regression. Still, a "necessary and sufficient condition" for the use of ensemble methods is that the single classifiers are accurate and diverse [3] [4]. An accurate classifier is better than a random guess. A set of classifiers is diverse if, when classifying a new set of data, they missclassify independently (i.e., the missclassified examples are not correlated between different classifiers. According to both [4] and [3], there are three main situations in which Ensemble methods perform better:

- **statistical**: From a small set of training data, many satisfying hypotheses can be inferred. By aggregating the results of all those classifiers, the propability of choosing a good final one is increased.
- **computational**: Instead of finding the global optimum, classifiers sometimes learn a local optimum. Using an ensemple method prevents the overall classifier from getting stuck in such local optima.
- **representational**: Some hypotheses can't be represented by the single classifiers. An aggregation of them provides more possible hypotheses and therefore might contain the target hypothesis, even if single classifiers could not reach it.

Ensemble methods also help when a single model overfits the data, since it combines different classifiers. Still, whenever applying an ensemble method, the better results should be worth the training overhead due to the training of several models. Now, I will explain three different ensemble methods.

Random Forest. A Random forest is made out of different decision trees. The trees are build on random subsets of the data set. Additionally, in every iteration of the build process, only a random subset of the features are taken into account. Like this, a forest of trees with random differences is build. Classification is then done by voting among the individual trees.

Bagging. The name is a combination of **B**ootstrapping and **agg**regating, and so is the methodology: Bootstrapping is used to produce several subsets of the training data (randomly drawing with duplicates). For each subset, a classifier is trained. When classifying a new example, the predictions of the different classifiers are aggretated to an overall prediction, for example by voting or averaging.

Boosting. In Boosting, different classfiers are trained on subsets of the data like in Bagging. The difference is that here, the subsets are not drawn independent: instead, in every iteration a classifier is trained and then tested. The next classifier is then trained with a focus on the examples the previous one misclassified in testing. For prediction, all classifiers are again combined. In the overall classifier, consisting of classifiers focussing alltogether on every part of the data set, the errors even out.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[2] T. M. Mitchell, *Machine Learning.* New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[3] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, (London, UK, UK), pp. 1–15, Springer-Verlag, 2000.

[4] Lecture slides: Machine Learning Autumn 2019.

Submitted by Klara Schlüter on November 6, 2019.