Klara Schlüter
Machine Learning
NTNU - Department of Computer Science

September 30, 2019

ASSIGNMENT 1 — Theory

Problem 1. Concept learning

In concept learning, the task is to learn a categorization of a set from a training subset: which elements belong to a certain subset, the concept, and which do not? For example, consider the set of all pullovers in my wardrobe. A concept in the set of these pullovers could be "Klaras prefered pullovers". Performing concept learning on this task means to learn how to decide for an arbitrary pullover if it belongs to this concept. For learning, there is a training subset given, consisting of positive and negative examples together with their classification given, i.e. a blue Hoodie as a negative example, and a red hoodless pullover as a positive example.

Problem 2. Function approximation

The term *function approximation* is another name for the process of learning, as approximating a certain function is what is actually done: The lecture's textbook explains machine learning as the task to learn an operational description of some ideal function defining the optimal behaviour (operational in this context means computationally feasable). The function actually learned by the machine often has to be an approximation, since learning a perfect operational description might for example take too much time or space or require knowledge the machine doesn't have.

Problem 3. Inductive bias

What is inductive bias in the context of machine learning...
To generalize from the given training examples, every concept learning algorithm needs to make some assumptions, for example about a certain structure in the possible hypotheses or about rules for building hypotheses. The set of these assumptions are called inductive bias of a concept learning algorithm.

...and why is ist so important?
If no assumptions are made at all, then no generalization beyond the training examples is possible, which means that no prediction can be derived deductively for any inputs that are not contained in the training set.

Still, we want to obey as few assumptions as possible. Therefore, the lecture's textbook defines the inductive bias as any set of assumptions that restricts the learning as little as possible (is minimal) but still allows for every input to deductively derive a classification from the training set.

[...] What can you say about the inductive bias for each of them?

For the Candidate Elimination algorithm, the textbook defines the inductive bias as "The target concept c is contained in the given hypothesis space H." This means that since we only consider conjunctions and therefore choose the hypothesis space as containing all possible conjunctions, the algorithm can't derive any other hypotheses (and therefore is biased).

To decision trees, Occam's razor can be applied: "Prefer the simplest hypothesis that fits the data." [Textbook]. For example, the ID3 algorithm searches the hypotheses space from simple to complex (hill-climbing approach) and stops as soon as it has found a hypothesis that is consistent with the training data. As a result, the simplest fitting hypothesis is returned (was obviously prefered over more complex ones).

In comparison, the following can be observed: The inductive bias in Candidate Elimination introduces hard restrictions to the result, since it excludes certain hypotheses from the search space even if they may be (the only) consistent ones, and thus searches through an incomplete hypothesis space. This type of inductive bias is called restriction bias or language bias.Still, the algorithm returns all consistent hypotheses contained in the searchspace, the inductive bias does not affect the search. While decision tree algorithms work on a complete hypothesis space, simpler solutions are prefered while searching, so the search is affected by the inductive bias. This type is called preference bias or search bias. Typically, search bias are prefered over restriction bias.

Problem 4. Validation

What is overfitting...
In the textbook, a learned hypothesis is defined as overfitting, if there is another (distinct) hypothesis with a greater error on the training instances, but a smaller error on the set of all instances. This means: The algorithm has learned a hypothesis that is "too complex", as it considers patterns that are not important for the task since they are coincidentally in the training data. This can be for example due to noise or to a training set that is too small.

...and how does it differ from underfitting?
While an overfitting hypothesis covers too much and therefore irrelevant patterns or regularities in the training data, an algorithm learning an underfitting hypthesis does not capture enough of the relevant patterns in the data. An overfitting hypothesis is too special, too close to the training data as it follows every smallest movement; an underfitting hypothesis is too general, it abstracts from important structures in the training data. Both do perform much worse on the set of all instances than a perfectly fitting model.

Briefly explain what a validation set is.
A validation set is a subset of the available data. Instead of being used for training like the subset's complement, the training set, it is used to validate the learned function by running it on the elements of the validation set and comparing the result to the given, desired result. Particularly, in decision tree learning, it is furthermore used to decide, if a hypothesis should be pruned or not.

How can cross-validation be used to mitigate overfitting?
Cross validation is a technique to estimate how well a learned function performs on the set of all instances, or even to identify the best function among several learned functions. For cross validation, the available example data is split into a training set and a validation set. The training set is used to learn a function that is then applied to the validation set to calculate its error. In different types of cross validation, several functions can be learned and validated to choose the best one or merged to a even smarter model.

Problem 5. Candidate Elimination algorithm

Assuming that the column *Sex* should be taken into account even if it isn't listed in the argument explanation of the exercise specification, the Candidate Elimination algorithm yields the following version spaces after each step:

state after	**most generic hypotheses**	**most specific hypotheses**
initialisation	$G = \{<?,\ ?,\ ?,\ ?>\}$	$S = \{<\varnothing,\ \varnothing,\ \varnothing,\ \varnothing>\}$
example 1	$G = \{<?,\ ?,\ ?,\ ?>\}$	$S = \{<Female,\ Back,\ Medium,\ Medium>\}$
example 2	$G = \{<?,\ ?,\ ?,\ ?>\}$	$S = \{<Female,\ ?,\ Medium,\ ?>\}$
example 3	$G = \{<?,\ ?,\ Medium,\ ?>\}$	$S = \{<Female,\ ?,\ Medium,\ ?>\}$
example 4	$G = \varnothing$	$S = \varnothing$
example 5	$G = \varnothing$	$S = \varnothing$

The fourth example, which is a positive example, is excluded both by the only hypothesis in G and the only hypothesis in S of the version space as defined after processing the third example. Therefore both are deleted, and no new elements can be derived. Hence, after processing the fourth example, both the set of the most generic hypotheses and the set of the most specific hypotheses are empty.

The hypothesis space does not include any hypotheses consistent with the first four hypotheses (and therefore obviously no hypotheses consistent with all five hypotheses either). The reason for this is that it only contains conjunctive hypotheses, which are unable to modle this specific task.

Submitted by Klara Schlüter on September 30, 2019.