Notes on session 8

Wilhelmiina Hämäläinen

9th November 2005

1 Robustness

Robustness is a very important property of a model. Intuitively, it means that the model is stable and small errors in model construction do not have dramatic effects on results. The opposite is that the model is *sensitive* to either small variations in data or initial model parameters (domain knowledge). For example, the data can contain outliers (exceptional data points), noise (small errors in attribute values) or missing attribute values. On the other hand, some models require parameter values from the expert, which can be hard or impossible to estimate accurately (e.g. the ideal number of hidden nodes, type of activation function and stopping criterion in a neural network, the type of kernel function in SVM, or the measure for data purity and stopping criterion for decision trees).

Some modelling paradigms produce more robust models than others, but robustness depends on also the the size of data set and model complexity. Thus, it also related to *overfitting* problem: if the model is very sensitive to data, it overfits easily and describes even errors in the data set. On the other hand, a robust model can tolerate errors and doesn't overfit so easily. As a result, it generalizes better to new data.

Another aspect of robustness is the *inductive bias* or inductive principles. Inductive bias consists of the assumptions (conditions), under which the model works well. For example, in linear regression we demand that the relationship between Y and $X_1, ..., X_k$ is linear, the independent variables X_i are not correlated and there are no outliers. In naive Bayes, we assume that the explanatory variables $X_1, ..., X_k$ are conditionally independent, given the class value Y, and if we have nominal data, then the decision boundary should be approximately linear. In decision trees, we assume that the classes can be separated and there are no contradictory values (data

points, which have the same values in $X_1, ..., X_k$ but different values in Y). Modelling paradigms, which make less restricting assumptions or which tolerate better the violation of these assumptions are more robust.

Fedor gave a nice formalization for robustness concerning data: If the difference between two data sets D_1 and D_2 is small, then also the difference between resulting models is small. I.e. Let d_1 measure the difference between data sets and d_2 the difference between models. Let M_1 be a model produced from data set D_1 and M_2 from data set D_2 in some modelling paradigm. Then we say that the modelling paradigm is robust, if for any data sets D_1 and D_2 the following condition holds: If $d_1(D_1, D_2) < \delta$, then $d_2(M_1, M_2) < \epsilon$ for arbitrary $\epsilon, \delta > 0$.

We could also add the influence of domain knowledge (initial parameters) into this definition: Let θ stand for initial parameters estimated by an expert. Then the condition becomes: If $d_1(D_1, D_2) < \delta_1$ and $d_3(\theta_1, \theta_2) < \delta_2$, then $d_2(M_1, M_2) < \epsilon$.

Another note: It doesn't matter, if the models are very different in some irrelevant aspects, as long as they produce the same predictions. Thus, the difference function d_2 should measure the differences between predictions $Y_1 = M_1(X)$ and $Y_2 = M_2(X)$ for all possible attribute values $X = X_1, ..., X_k$.

2 Data vs. model parameters

Data can be described by attribute-value pairs, $A = v$. The possible values for attribute A are called *domain*, $dom(A)$. The type of domain defines the data type. The basic division is to numeric and categorial data. Numeric data can be further divided into discrete and continuous values, and categorial data into nominal and ordinal values. If these are unclear, I suggest to recall them from the first lecture slides.

Now the confusion appears easily, when we think about model parameters. The parameters are usually always numbers, but sometimes they can be boolean values. For example, in probabilistic models the parameters consist of probability distribution of form $P(A = v)$ or density function $f(A)$. In the first case, the model parameters are discrete values, while in the latter one they are continuous (but we can represent them only by finite precision – i.e. by dicrete numbers – unless we give an interval). In the same way, fuzzy logic and Dempster-Shafer theory assign numeric values for parameters. The data itself can have either numeric or categorial values, as long as you can express it by a boolean-valued propositions $A = v$.

If $dom(A)$ is continuous or discrete numeric, but very large, a common solution is

to discretize it first, i.e. create a new attribute A', which has only a small discrete domain. For example, if $dom(A) = \{a_1, ..., a_k\}$, then we have to define thresholds $v_1, ..., v_{k-1}$ such that $A' = a_1$, if $A < v_1$, ..., $A' = a_k$, if $A > v_{k-1}$. Notice also that we can always represent any kind of data as boolean valued (nominal) data!

Notice that the opposite transformation, from categorial to numeric data is much more difficult, because now we should change from less informative to more informative values. One common solution is to represent all data as boolean-valued and interprete the truth values as 0 and 1. Notice that in the same time the number of attributes can increases and the model can become too complex.

3 NP-hard problems

We noticed that most of the interesting problems concerning model construction and reasoning by models are NP-hard. NP-hard problem are even more difficult than NP-complete problems (which can be solved in polynomial time by a nondeterministic Turing machine), but they share one common property: if an NP-hard or an NP-complete problem could be solved in polynomial time (by deterministic Turing machine), then all problems in class NP could be solved in polynomial time, as well. NP-hard problems are harder, because they cannot be solved in polynomial time even by nondeterministic Turing machine (or nobody has invented such a solution), and thus they do not belong to class NP themselves.

Typically NP-hard problems are enlargements of NP-complete problems. For example, in $3SAT$ problem we should just decide, wheteher a logical clause of form $(v_{11} \vee v_{12} \vee v_{13}) \wedge ... \wedge (v_{k1} \vee v_{k2} \vee v_{k3})$ is true with some truth value assignment $v_1, ..., v_n$, where each $v_{ij} = v_k$ for some $k = 1, ..., n$. A more complex problem is to find **all** such truth value assignments. That is exactly, what an ATMS does: for all belief nodes we calculate the minimal sets of assumptions, under which it is true. This gives also the intuition, why reasoning by a general Bayesian network is NP-hard: to calculate probability $P(Y)$, for a non-root node Y, given parent nodes $X_1, ..., X_k$, we should calculate the probabilities $P(X_1), ..., P(X_k)P(Y|X_1, .., X_k)$ for all possible value combinations $X_1, ..., X_k$, and sum them together. If X_is are boolean valued, we have 2^k different value combinations, which means exponential time. However, in a special case, the networks has only 0 and 1 probabilities and $P(Y)$ can also be just 0 or 1. Now, we can nondeterministically calculate $P(Y)$ in linear time: we just guess a value combination (if such exists) and check that $P(Y) = 1$. This means that if we could calculate probabilities in Bayesian networks in polynomial time, we

could also solve 3SAT in polynomial time and win 1 000 000 dollars!

Other NP-hard problems concerning modelling paradigms are: learning an optimal Bayesian network, an optimal decision tree, and an optimal neural network. Reasoning by ATM-systems and reasoning by a Dempster-Shafer system containing no missing values (i.e. when beliefs define a complete probability model). Maybe others, too, inform if you know!

4 Reasoning tasks

Most of the reasoning tasks implemented by expert systems concern either classification or regression. Classification is a general name for all prediction tasks, where you have to predict a categorial value. If you predict a numeric value, it is called regression. Notice that we do not have to give deterministic prediction, but it can be probabilistic or otherwise uncertain. E.g. a probabilistic classifier produces a probability to belong to some class C, while Dempster-Shafer theory produces a belief and a plausibility to belong to a given class, and fuzzy systems, a fuzzy value.

A totally different task is involved in planning: we are given a starting point and a goal we want to reach and we should learn an optimal sequence of actions, which leads from the starting point to the goal. For example, genetic algorithms, case-based methods and rule-based systems (fuzzy systems, enlargements of TMS) suit for planning, too.

Optimization task (performed by genetic algorithms) is also different: we can search an optimal model among all possible models. Genetic algorithms can also solve other, more complex tasks, where the main idea is always to find an optimal solution among all alternatives.

Some methods (Bayesian networks, HMMs) can be used to calculate the probability of some state of affairs (i.e. attribute value combination). TMS can reveal contradictions and consequences of given assumptions.