Explaining Results of Neural Networks by Contextual Importance and Utility

Kary FRÄMLING

Dep. SIMADE, Ecole des Mines, 158 cours Fauriel, 42023 Saint-Etienne Cedex 2, FRANCE
framling@emse.fr, tel.: +33-77.42.66.09

Abstract: *The use of neural networks is still difficult in many application areas due to the lack of explanation facilities (the « black box » problem). The concepts of contextual importance and contextual utility presented make it possible to explain the results of neural networks in a user-understandable way. The explanations obtained are of same quality as those of expert systems, but they may be more flexible since the reasoning module and the explanation module are completely separated.*

The numerical complexity of estimating the contextual importance and contextual utility is to a great extent solved by the neural net proposed (INKA), which also has good function approximation and training properties.

1. Introduction

In a great number of application areas, the « black box » aspect of *neural networks* is one of the main reasons for using *expert systems* instead, since they can at least show the reasoning path (rules or other) leading to the result. The most common approach for explaining the results of a neural network is therefore to try to identify *rules* that correspond to the behaviour of the net.

This approach means implementing the reasoning of a neural network as a rule base. Unfortunately, rule bases don't have the same expression power as neural networks. This is especially true for continuous real-valued functions. The output of a rule base changes in a non-continuous way as the set of fired rules changes[1] when changing the values of the input variables. Rule bases also tend to grow too big to maintain and understand and it is difficult to ensure the coherence of the rules. These are the reasons why finding a corresponding rule-base (or some other symbolic reasoning model) is difficult.

The methods proposed in this paper offer capabilities for directly explaining the **results** (not the knowledge itself) of neural networks without transforming the

---

[1] Certainty factors and fuzzy sets are a potential solution to non-continuity problems, but they increase the complexity of the rule base and they make explanations difficult to understand.

knowledge of the neural net. They are based on the concepts of *importance* and *utility* of an input on the output variable, which are indicators that may be calculated directly from the neural net. Importance and utility are key concepts used in multiple criteria decision making, which is the main application area studied so far. However, the concepts and methods presented here are also applicable to other problem domains.

2. Multiple criteria decision making

One application area where it is difficult to use classical expert systems, with or without certainty factors, is *multiple criteria decision making* (MCDM). It is surprising to see that among the great number of MCDM methods, there is hardly any explanation facilities provided. Still, in typical selection problems, it is crucial to motivate the recommendation of the MCDM system.

MCDM problems require finding a model of the decision maker's preferences, which may be called his *preference function*. However, the decision maker is quite often a group of people or an abstract person (society, nature, economy, ...). This makes it very difficult to explicitly express the preference function, which is the reason for using machine learning instead. It is then sufficient to find examples of decisions already made by the decision maker, no matter what his nature is. A true preference function is usually non-linear and continuous, which makes its mathematical expression quite complex. Most MCDM methods are linear and don't support machine learning, while expert systems usually are non-continuous. This is the reason for using neural nets.

In MCDM methods, the importances of the *selection criteria* are expressed by *weights*, while the transformation of the values of the criteria into *utility values* is done with *utility functions*[2]. For a car selection problem, these concepts may be used for giving explanations like « The car is <u>good</u> because it has a <u>good</u> size, <u>decent</u> performances and a <u>reasonable</u> price, which are *very important* criteria », where words indicating <u>utilities</u> are underlined and words indicating the *importance* are in italics. The fact of using a linear model makes the definition of importance and utility quite easy. However, when using non-linear models like neural nets, the task becomes more difficult.

3. Contextual importance and utility as basic explanation components

For continuous, non-linear systems like fuzzy logic or neural nets, the importance of a criteria and the utility of a value depend on the context. The notions of *contextual importance* (*CI*) and *contextual utility* (*CU*) [Främling, 1992, 1996], [Främling & Graillot, 1995] are therefore introduced. The context is defined by the current values of the inputs (the selection criteria in the MCDM

---

[2] This is true for a great number of MCDM methods, but not all. Some methods use pseudo-criteria or other ways of avoiding utility functions.

case). The contextual importance of an input on an output is then defined as the ratio:

$$CI = \frac{dynmax_j(c_i) - dynmin_j(c_i)}{absmax - absmin} \qquad (1)$$

where *CI* is the contextual importance, $dynmax_j(c_i)$ and $dynmin_j(c_i)$ are the maximal and minimal output values observed by varying the value of the input *j*, c_i is the context studied and *absmax* and *absmin* define the whole output value range. *CU* is defined as:

$$CU = \frac{n_i - dynmin_j(c_i)}{dynmax_j(c_i) - dynmin_j(c_i)} \qquad (2)$$

where *CU* is the contextual utility and n_i is the current output value. Both *CI* and *CU* are continuous, numerical values that are easy to find for one input using Monte-Carlo simulation or other methods. They may further be transformed into symbolic values in order to produce explanations. The transformation may be done simply by defining value ranges for « good », « bad », « important », « little important », ..., or by using fuzzy sets.

The notions of *CI* and *CU* are illustrated by the example in Figure 1. The preference function learnt by the neural net is for choosing a car, where the inputs of the neural net are the characteristics of the car and the output value is the preference value (or « score ») obtained. The context is defined by the car studied, the « SAAB 900 S 2.0-16 », which gives the values of all inputs except the one studied. The preference value is indicated as a function of the price of the car, the cross indicating the current value. The scale indicated on the right shows some limits for translating *CU* into words. The same kind of scale is used for *CI*.

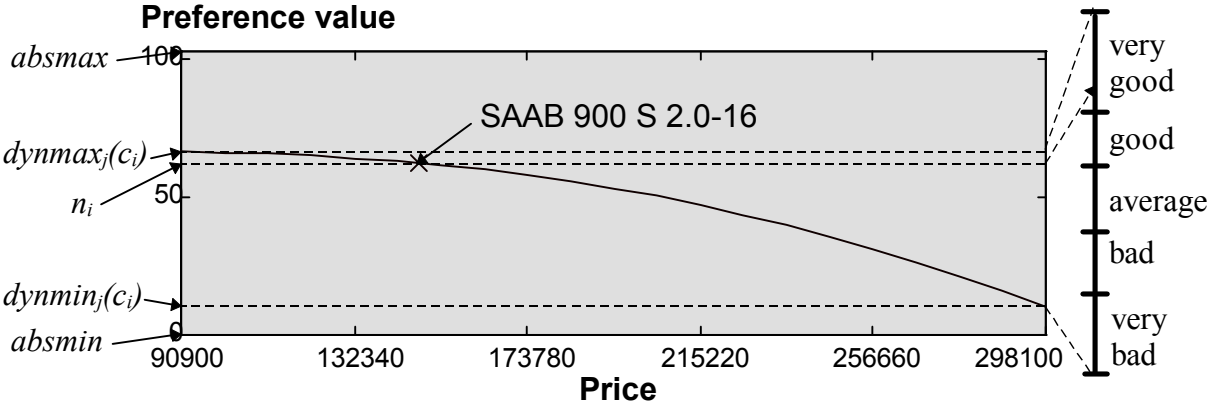

Figure 1. Illustration of the notions of contextual importance and contextual utility using the problem of choosing a car [Främling, 1996].

From the figure we can see that the price is very important for the preference value and that the current price is very good[3]. Looking at this example, it is obvious that the sum of the contextual importances of all criteria is not necessarily equal to one, as is the case for linear preference functions. Several criteria may even have an importance of one at the same time! This phenomena is also found in expert systems, where changing the value of one selection criteria may cause the fired rules to change, thus causing changes that may be up to 100% on the output value.

4. Intermediate concepts

Intermediate concepts are used for structuring the domain, which makes it possible to give explanations with several levels of detail. They correspond to the different levels of the inference tree of a rule-based expert system. A typical intermediate concept for choosing a car would be « performances », which groups together basic concepts like « power », « weight », « top speed » and « acceleration ».

For linear functions, importance of intermediate concepts is simply the sum of the importance of the basic criteria. In the case of non-linear functions, however, finding *CI* and *CU* of such concepts has to be done using the same methods as for basic criteria. Instead of changing the values of one input, the changes are made to the values of all basic criteria that the intermediate concept is composed of.

Criteria	***Performances***	***Max power***	***Weight***	***Top speed***	***Acceleration***
$dynmin_j$	45.3	54.2	54.8	54.0	52.8
$dynmax_j$	78.5	65.2	58.8	64.0	62.8
***CI* (%)**	32	11	4	10	10
CU	0.29	0.28	0.63	0.33	0.45
CI_{rel} **(%)**		33	12	30	30

Table 1. Values for $\mathrm{dynmax_j(c_i)}$ *and* $\mathrm{dynmin_j(c_i)}$ *obtained for the intermediate concept « performances » and its basic criteria.*

Table 1 indicates the values $dynmax_j(c_i)$ and $dynmin_j(c_i)$ obtained this way for « performances » and the concepts that it depends on, again for the context « SAAB 900 S 2.0-16 ». On the third line, we find the contextual importance of them all when calculated relative to the preference value (using *absmax* and

[3] This opinion of course depends on each decision maker's resources. Also, a scale like « expensive », « not expensive », ... would be more appropriate.

absmin). However, the importance of the sub-concepts relative to « performances » is more interesting (line five). It is calculated by using the values of $dynmin_j$ and $dynmax_j$ for « performances » instead of using *absmax* and *absmin* in the formula *(1)*. The values shown let us automatically[4] produce the following explanation:

The SAAB 900 S 2.0-16 has an average ranking, preference value 61 on 100 because:
- Price, which is *extremely important* has a very good value (147800).
- Performances, which is *very important*, has a rather bad value
- ... other selection criteria.

Performances are rather bad because:
- The maximum power, which is *important*, is rather bad.
- The acceleration, which is *important*, is average.
- The maximum speed, which is *important*, is rather bad.
- The weight, which is *less important*, is good.

The actual contents of the explanation depends on the strategy used (importance first, utility first, « rule base style », ...). It is also clear that the syntax of the automatically produced phrases could be improved quite easily.

4.1 Concept trees and inference trees

Intermediate concepts are used extensively in rule based expert systems for dividing the main problem into smaller sub-problems, solved by a subset of rules. As the rules are fired by the inference engine, an inference tree is produced where we can follow the reasoning used. At each node we see the hypothesis of a rule and the conditions which have caused the rule to be fired. This information is then used for explaining the result.

In the previous chapter there was no inference tree, only a concept tree. We can, however, always reconstruct the same inference tree as that of a rule based system. The contextual importance determines the nodes to include in the explanation by setting a minimal limit. The contextual utility corresponds to the conditions and hypothesis of the inference tree, where either the value or the contextual utility (or both) may be used in the explanations.

The advantages of using contextual importance and utility in explanations compared to using rules are numerous. The definition of a concept tree is much easier than defining (and maintaining) a complete rule base. Explanations also directly correspond to the actual reasoning of the neural net, without first having to translate it into some symbolic representation. This also means that it is possible to explain reasoning under uncertainty, since we may explain a certainty factor or a probability just as well as a preference value. It is therefore

[4] Originally produced in French, only translated here.

possible to use the presented methods even in diagnostics or other application areas, as indicated in the next chapter.

4.2 Concept networks and semantic networks

Figure 2 shows a small part of an example semantic network for diagnosing car problems. The links used for the actual diagnosis are dotted, while links used mainly for explanation purposes are indicated with solid lines. A neural net could be teached to do exactly the same kind of diagnosis, but without the need to specify limits like « > 5 minutes » or « < 0° C ». Neural nets also present the advantage of giving certainty factors (or probabilities or other indicators of uncertainty) instead of just giving boolean answers (yes/no).

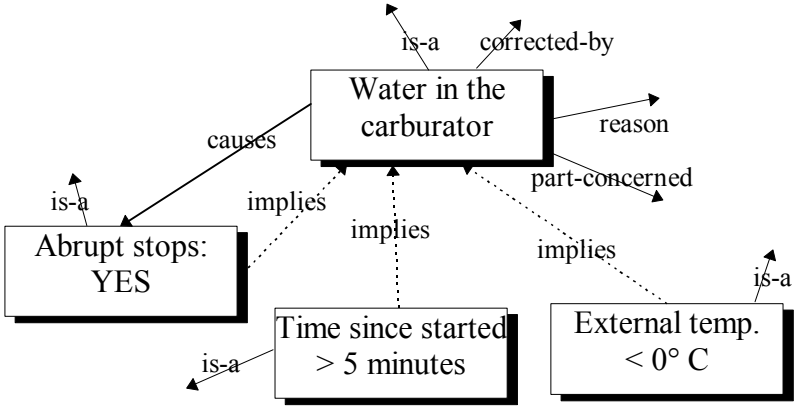

Figure 2. A part of an example semantic network for diagnosing car problems, shown for the diagnosis « water in the carburator ».

The same semantic network could be used for explaining the result (the certainty factor) of a neural net, except that there will initially be no connections between the input values and the diagnosis and that conditions like « Time since started > 5 minutes » are replaced by the concept itself (« Time since started » in this case). The connections are established by the contextual importance for the case explained. When we do have water in the carburator, we identify the same connections as the ones in Figure 2. The contextual utility replaces the limits in the explanations, since we can identify values like « favorable » and « unfavorable » for the diagnosis.

As shown by this example, the main techniques already used for explanations in symbolic reasoning are directly applicable to the methods presented here. Their use is even simplified by the fact that the reasoning and the explanation modules are completely separate, thus simplifying the construction and maintenance of both modules.

5. The INKA network

Finding the values of $dynmax_j(c_i)$ and $dynmin_j(c_i)$ easily gives problems of combinatorial explosion of the number of input value combinations to test. The more inputs we analyze, the more difficult the problem becomes. This problem is especially serious for inputs with discrete values (like symbolic concepts).

The INKA network described in [Främling, 1996] offers a solution to this problem.

The INKA network approximates functions by interpolation between the peaks of *radial basis functions* (RBF) instead of trying to make them fit directly to the approximated function, as is usually done in RBF networks. This is the main reason why the network is called the **Interpolating**, Normalising and Kernel Allocating (INKA) network. **Kernel Allocating** means that the number of hidden neurons is adjusted in order to improve the function approximation.

Like most RBF type networks, INKA uses three layers (*input layer*, *hidden layer* and *output layer*). Every neuron i of the hidden layer has a centroid $\vec{w}_i$ that is defined by its weights. The number of weights is the same as the number of inputs since no bias is used. The activation value of a hidden neuron is the squared *Euclidean distance* between the input vector $\vec{x}$ and $\vec{w}_i$. The output value of a hidden neuron is calculated by the *Inverse MultiQuadric Equations* (IMQE) function [Hardy, 1971]:

$$h_i = (1 + \frac{r_i^2}{d_i^2})^{-1/2} \qquad (3)$$

which has given better results than other functions tested (like the Gaussian function, for instance). The interpolation capabilities of the network are improved by the fact of normalizing the input values of the output neurons (also proposed in [Nowlan, 1990]):

$$hn_{j,i} = h_i / \sum_{x=1}^{m} h_x \qquad (4)$$

where $hn_{j,i}$ is the normalized output value of hidden neuron i for output neuron j, h_i is the output value of hidden neuron i and m is the number of hidden neurons connected to output neuron j. A weighted sum is used for calculating the final output value. Tests performed have shown that the interpolation capabilities are greatly improved by this normalisation, especially for sub-optimal network parameters. The normalisation also means that there is no need for any bias input in the output layer.

The learning algorithm used is based on the principle of starting with an empty hidden layer and adding new hidden units for input vectors where the output error is the greatest. New hidden units are added until the global error becomes small enough. This principle means that hidden units are first allocated for the « peaks » and « valleys » of the approximated function, which is a great advantage when searching for the maximal and minimal output values as a function of input values. Instead of testing all input value combinations, we only need to test input value combinations taken from the $\vec{w}_i$ vectors, which greatly reduces the combinatorial explosion when searching for $dynmax_j(c_i)$ and $dynmin_j(c_i)$ for several inputs. The normalisation at the output layer also filters

out excessive oscillations often found in RBF and backpropagation networks. This filtering makes it possible to get reliable estimations of $dynmax_j(c_i)$ and $dynmin_j(c_i)$.

6. Conclusion

The concepts and the definitions presented in this paper are simple and generally applicable. The explanations obtained can be more flexible than those of expert systems, since intermediate concepts are easy to define and modify depending on the user. Explaining reasoning under certainty is also well handled, which is usually a problem in expert systems.

As a conclusion, we think that a major obstacle for using neural networks, that of their incapability of explaining their results, hardly exists anymore. The neural net itself remains a « black box », but it becomes a black box capable of communicating and explaining its results just as well as or better than systems using symbolic reasoning.

The Matlab macro files and the data used in this article may be obtained by FTP from « ftp.emse.fr », directory « /pub/INKA » or by request from « framling@emse.fr ».

References

FRÄMLING Kary, *Les réseaux de neurones comme outils d'aide à la décision floue*, Thesis of D.E.A., École des Mines de Saint-Etienne, France, 1992, 55 p.

FRÄMLING Kary, *(in French) Learning and Explaining Preferences with Neural Networks for Multiple Criteria Decision Making*, Ph.D. thesis, École des Mines de Saint-Etienne, France, 1996, 304 p.

FRÄMLING Kary & GRAILLOT Didier, *Extracting Explanations from Neural Networks*, Proceedings of the ICANN'95 Conference, Paris, France, 1995, pp. 163-168.

Hardy R. L., *Multiquadric Equations of Topography and Other Irregular Surfaces*, Journal of Geophysical Research N° 71**,** 1971, pp. 1905-1915.

Nowlan S., *Maximum likelihood competitive learning*, Proceedings of Neural Information Processing Systems, 1990, pp. 574-582.