# Extracting Explanations from Neural Networks

Kary FRÄMLING, Didier GRAILLOT

Dep. SIMADE, Ecole des Mines, 158 cours Fauriel, 42023 Saint-Etienne Cedex 2, FRANCE
framling@emse.fr, graillot@emse.fr, tel.: +33-77.42.66.09, +33-77.42.01.30

***Abstract:*** *The use of neural networks is still difficult in many application areas due to the lack of explanation facilities (the « black box » problem). An example of such applications is multiple criteria decision making (MCDM), applied to location problems having environmental impact. However, the concepts and methods presented are also applicable to other problem domains. These concepts show how to extract explanations from neural networks that are easily understandable for the user. Explanations obtained may in many cases even be better than those of expert systems. The INKA network presented in this paper is well adapted for MCDM problems, while also having properties that simplify the extraction of explanations compared to most other neural networks.*

## 1. Introduction

The use of *neural networks* is still difficult in many application areas due to the lack of explanation facilities (the « black box » problem). This is often 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. This approach to explanations works especially well for deterministic reasoning, but it is also applicable to probabilities or certainty factors, like in MYCIN [Shortliffe, 1976] which could explain its medical diagnosis using certainty factors. However, if there are multiple conclusions reached by several reasoning paths to explain, the task quickly becomes difficult. This is one of the reasons why fuzzy logic systems are just about as bad in explaining their behaviour as neural nets despite the fact that they use the same kind of rules as expert systems.

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 [Pomerol, 1993]. Still, in typical problems having environmental impact, like the selection of waste disposal sites, it is crucial to motivate the decision to the local population and other actors concerned.

MCDM problems require finding a model of the decision maker's preferences, which is usually called his *preference function*. However, in environmental location problems the decision maker is usually 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 is his nature. The 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 (except for fuzzy logic, for instance). This is the reason for using neural nets instead.

The methods presented in this paper are applicable to most neural nets, but they are numerically feasible only for certain of them. The INKA network presented in chapter 3 is highly suitable for learning preference functions as well as for overcoming the numerical problems. However, the concepts and methods presented here are also applicable to other problem domains.

## 2. Importance and utility as basic explanation components

In MCDM methods, the *importance* of a *selection criteria* is expressed by a *weight*, while the transformation of the values of the criteria into *utility values* is done with a static *utility function* [1] [Pomerol, 1993]. It is therefore easy to give explanations like « The disposal site for industrial waste has been selected because it has a very *good* geology and is *far* from the closest town, which are very important criteria » [Främling & Graillot, 1994], where words indicating *utilities* are in italics and words indicating the importance are underlined. 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.

### 2.1 Dynamic importance and utility

For continuous, non-linear systems like fuzzy logic or neural nets, it is necessary to introduce new notions of *dynamic importance* (DI) and *dynamic utility* (DU). Dynamic refers to the fact that the importance and utility depend on the current values of the selection criteria. The DI of an input on an output is then defined as the ratio

$$DI = \frac{dynmax - dynmin}{absmax - absmin} \qquad (1)$$

where *DI* is the importance, *dynmax* and *dynmin* are the maximal and minimal output values observed by varying the value of the input and *absmax* and *absmin* define the whole output value range. DU is defined as

$$DU = \frac{curval - dynmin}{dynmax - dynmin} \qquad (2)$$

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

It is also important to point out that the sum of the importances of all criteria is not necessarily one. Several criteria may, in fact, 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 activated rule to change, thus causing changes that may be up to 100% on the output value.

### 2.2 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 depends on the basic concepts of « 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 *DI* and *DU* of such concepts has to be done using the same methods as for basic criteria. Instead of changing the values of one input,

---

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

the changes are made to the values of all basic criteria that the intermediate concept is composed of. However, this easily gives a problem of combinatorial explosion of the number of input value combinations to test in order to find the maximal and minimal values. This problem is especially serious for inputs with discrete values (like symbolic concepts).

### 2.3 Explanations by similarity

Another kind of explanation facility that is possible with certain neural nets is explanation by similarity, which however will not be treated in detail in this paper. The principle is to use known cases as a base for explanations. These cases are the examples in the training set, for which there exists pre-defined explanations. It is then enough to identify which case is the « closest » to the current input and apply the existing explanation, together with an indication of the degree of similarity. Most numeric classification methods offer a direct way of defining the concept of « closest ». Feature (or Kohonen) maps and *Radial Basis Function* (RBF) networks also offer the same concept, as well as the INKA network presented later. Rule-based expert systems do not, however, offer this kind of possibilities.

## 3. The INKA network

The *Interpolating, Normalising and Kernel Allocating* (INKA) network is based on the principles of RBF networks, which means that it is a three layer fully connected feedforward network. The *input layer* distributes the input values to the neurons of the *hidden layer*. Every neuron $i$ of the hidden layer has a centroid $\vec{w}_i$ defined by its weights. For an input vector $\vec{x} = (x_1, x_2, ..., x_n)$, the hidden neuron computes its activation value as the squared *Euclidean distance* between $\vec{x}$ and $\vec{w}_i$ as

$$r_i^2 = (\vec{x} - \vec{w}_i)(\vec{x} - \vec{w}_i)^T \qquad (3)$$

where $r_i^2$ is the activation value of hidden neuron $i$, $\vec{x}$ and $\vec{w}_i$ are the 1 x $n$ input and weight vectors and $T$ means matrix transposing. The output function used in this paper is the *Inverse MultiQuadric Equations* (IMQE) function [Hardy, 1971]

$$h_i = (1 + \frac{r_i^2}{d_i^2})^{-1/2} \qquad (4)$$

where $h_i$ is the output of hidden neuron $i$ and $d_i^2$ is a positive scalar constant, that may be individual or the same for all hidden neurons. The *output layer* neurons have an activation function which is the normalized weighted sum of their inputs as in [Nowlan, 1990; Benaim, 1994]. The normalisation makes it possible to largely reduce the problems of finding an optimal $d_i^2$-value and gives better generalisation (interpolation) properties than classical RBF networks. It is done by

$$hn_{j,i} = h_i / \sum_{x=1}^{m} h_x \qquad (5)$$

where $hn_{j,i}$ is the normalized output value of hidden neuron $i$ for output neuron $j$ and $m$ is the number of hidden neurons connected to output neuron $j$. In the case of a fully connected network, $hn_{j,i}$ is the same for all $j$. Then the output value of output neuron $j$ is calculated as

$$o_j = \sum_{i=1}^{k} b_{j,i} hn_i \qquad (6)$$

where $o_j$ is the value of network output $j$, $b_{j,i}$ is the weight between output neuron $j$ and hidden neuron $i$ and $k$ is the number of hidden neurons connected to output neuron $j$.

### 3.1 The training algorithm

The main principle of the training algorithm proposed in this paper is to allocate new hidden neurons where the output error is the greatest, thus trying to reduce the global error as much as possible at each step. However, distance between centroids of two hidden neurons is not allowed to be smaller than the constant *c*.

1. Choose the values for the $d^2$ and *c* parameters.
2. Start off with an empty hidden layer, which means the network output is always zero.
3. Initialize the set of resting examples *R* to the whole training set.
4. Select the example in *R* for which the output error is the greatest, add it to the set of used examples *U* and remove it from *R*.
5. If there is no hidden neuron closer than *c* to the example, allocate a new hidden neuron *i* with $\vec{w}_i$ equal to $\vec{x}$.
6. Recalculate the *B* matrix for the examples in *U*.
7. Test if the global error measure is small enough. If not, go back to step 4. If it is small enough, recalculate the *B* matrix for the whole training set.

Figure 1. The INKA training algorithm.

Recalculating *B* on the whole training set is done only at the end of the training in order to avoid premature regression. *B* may be calculated in several ways, but in INKA it is done by the matrix inversion

$$Y = HN * B \qquad (7)$$

where *Y* is the matrix with the correct output values, *HN* is the matrix of normalized outputs of the hidden neurons and *B* is the output layer weight matrix to be found.

Some redundant hidden neurons are nearly always generated during the training. A simple optimization algorithm has therefore been developed that permits the removal of them, but the number of neurons removed is not always significant. It is, however, possible to be sure of not allocating much more neurons than absolutely necessary.

### 3.2 Solving the problem of combinatory explosion

The training algorithm tends to first allocate neurons for *significant training examples* (the extreme output values) [Hardy, 1971] and somewhat resembles *stepwise forward selection* discussed for instance in [Boyce et al., 1974]. Hidden neurons are always first created at the peaks of the approximated function. In order to find the maximum and minimum values of the INKA output function it is then normally sufficient to use hidden neuron centroid values for the inputs that are analyzed[2]. This is very fortunate, because it allows us to easily find *DI* and *DU* even for multiple inputs, as is the case for intermediate concepts used in explanations.

[2] « Normally » here means that there is no excessive extrapolation phenomena, which occurs when $d_i^2$ is too big.

## 4. Test results

The classical MCDM problem of choosing a car is used for testing the methods on a non-linear preference function. The preference function used is a weighted sum, with non-linearities introduced by the following rules:

- IF Price > 200.000,- FF THEN exponentially reduce the weight of all criteria and make it zero for 300.000,- FF
- IF Fuel consumption > 11l/100km THEN exponentially reduce the weight of all criteria and make it zero for 14l/100km
- IF maximum power > 200hp but no four-wheel drive THEN exponentially reduce the utility of maximum power to 30% for 240hp or more

The thirteen selection criteria are *price*, *power*, *transmission type*, *traction type*, *car size*, *trunk size*, *weight*, *top speed*, *acceleration 1000m*, *fuel consumption*, *esthetic score*, *image score* and *equipment score*. The criteria are of three different types, continuous numeric, discrete numeric and discrete numeric representing symbolic values. Many criteria are also highly correlated. The training set consists of twelve real cars plus twenty-six « pseudo-cars ». Two of the pseudo-cars correspond to the ideal and anti-ideal car, while the others have been interactively introduced by the decision maker in order to correct obvious errors in the preference function. Pseudo-examples are usually added for extreme values of the selection criteria, where the error is the biggest due to the small number of real examples available. The INKA parameter values were $d^2 = 1$, $c = 0$ and RMSE (*Root Mean Square Error*) error goal = 0.05. The number of hidden neurons created was quite high, thirty-five for thirty-eight examples. This was done on purpose by using a small RMSE limit in order to get a very high precision and thus simplify the comparisons of Table 1.

| Criteria # | **1** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | **10** | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *DI* (%) | **58** | 11 | 6 | 5 | 6 | 6 | 4 | 10 | 10 | **39** | 8 | 5 | 10 |
| Linear weight | **12** | 11 | 6 | 5 | 6 | 6 | 4 | 10 | 10 | **10** | 8 | 5 | 10 |
| *DU* | **0.94** | 0.28 | 1.00 | 0.00 | 0.74 | 0.60 | 0.62 | 0.33 | 0.44 | **0.90** | 0.66 | 0.99 | 0.45 |
| Linear utility | **0.72** | 0.29 | 1.00 | 0.00 | 0.76 | 0.61 | 0.63 | 0.33 | 0.45 | **0.61** | 0.67 | 1.00 | 0.45 |

Table 1. Dynamic importances and utilities obtained for the example car (a SAAB 900 S 2.0-16). The linear weights and utilities indicate the correct values for *DI* and *DU* if there would be no non-linearities neither in the importance nor in the utility functions of the selection criteria. The columns for price and fuel consumption are marked in bold.

For criteria not affected by the rules, we find the original weights and utility values. As expected, the importance of price and fuel consumption is significantly higher than the original weights. The utilities obtained for these criteria also show that the utility function is highly non-linear. For the intermediate concept « performances » mentioned before, we obtain an importance of 33 and a utility of 0.29. The values obtained can then be transformed into explanatory phrases like:

« The SAAB 900 S 2.0-16 has an *average* ranking, 61 / 100 (69th of 113 cars). Good points with this car are that it has a *low* price (147800 Francs) and a *low* fuel consumption (10.2 l/100km), which are both very important criteria, 60% and 40%. It has a value that is *quite bad* for performance, which is an important criteria, 34%. Among the criteria that are important for performance, there is the maximum power that is *bad* and the maximum speed that is *quite bad*. Among the more or less important criteria, there is the equipment level for which the value is *average*. »

The contents of the explanation may vary a lot depending on the value ranges defined for importance and utility values. Different strategies representing various points of view may also be used, like explanations by the most important criteria first, by the criteria having the best or worst values, …. Sometimes it is also useful to compare two cars at a time in order to determine why one is better than the other one, as in [Främling & Graillot, 1994]. In order to avoid giving too many details it is necessary to limit the backtracking through the intermediate concepts. Determining the best combination of all these possibilities and parameters is a difficult problem that concerns man-computer interaction in general and is not the subject of this paper. More details on this aspect may, for instance, be found in [Swartout & Moore, 1993].

## 5. Conclusion

The concepts presented in this paper are simple and generally applicable, yet providing possibilities to provide explanations with several levels of detail and from several points of view. The explanations obtained are more flexible than those of expert systems, since there is no need for a rigid, pre-defined structure defining intermediate concepts. Uncertainty is also handled better than in expert systems using certainty factors, whose propagation through the inference tree is difficult to follow and understand for the final user, not to speak about the work of the expert and the knowledge engineer who have to define them and their propagation method.

The INKA network presented makes it possible to use neural nets for MCDM, where the data sets are usually very small. It also makes obtaining explanations numerically feasible using the concepts presented. The combination of automatically learning non-linear preference functions and providing explanations on the recommendations given are both significant improvements to the state-of-art in multiple criteria decision making, where explanation facilities of the kind presented hardly exist.

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

## References


**Benaim M. 1994**. On Functional Approximation with Normalized Gaussian Units. *Neural Comp.* **6**, 319-333.

**Boyce D. E., Farhi A., Weischedel R. 1974**. *Optimal Subset Selection - Multiple Regression, Interdependence and Optimal Network Algorithms*. Springer-Verlag, Berlin.

**Främling Kary, Graillot Didier 1994**. *Système d'Aide à la Décision pour le choix de centres de stockage de déchets - StocDech, Manuels d'installation et d'utilisation*. Ecole Nationale Supérieure des Mines de Saint-Etienne, Saint-Etienne, France. 91 p.

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

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

**Pomerol Jean-Charles 1993**. *Choix multicritère dans l'entreprise*. Éditions Hermès, Paris. 391 p.

**Shortliffe E. 1976**. *Computer-based Medical Consultations: MYCIN*. Elsevier, Amsterdam.

**Swartout William R., Moore Johanna D. 1993**. Explanation in Second Generation Expert Systems. In: *Second Generation Expert Systems*. Edited by Jean-Marc David, Jean-Paul Krivine and Reid Simmons. Springer-Verlag, New-York, 543-585.