

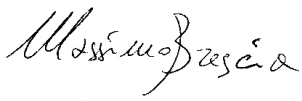
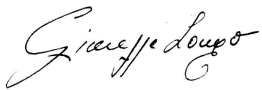
VONeural Project

Why use softmax activation function and cross-entropy error function

Doc. : softmax_entropy_VONEURAL-SPE-NA-0004-Rel1.0.odt

Issue: 1.0

Date: 1/10/2007

What	Who	When	Signatures
Prepared by	B. Skordovski	01/10/2007	
Approved by	M. Brescia	02/10/2007	
Released by	G. Longo	02/10/2007	

Revision Matrix

Issue	Author	Date	Section/Paragraph Affected	Reason/Initiation/Documents/Remarks

INDEX

.....	1
1 Reference Documents.....	5
2 Introduction.....	5
3 Activation function.....	5
4 1-of-C classification.....	7
5 Softmax activation function.....	9
6 Cross-entropy error function.....	10
6.1 Cross-entropy for two classes.....	10
6.2 Cross-entropy for multiple classes.....	12
7 Conclusion.....	13

TABLE INDEX

Tab. 1 – Reference Documents.....	4
-----------------------------------	---

EQUATION INDEX

Equation 1 – Threshold function.....	5
Equation 2 – Logistic sigmoid function.....	5
Equation 3 – Softmax function.....	9
Equation 4 – Softmax-logistic equation.....	9
Equation 7 – Cross-entropy error function.....	11
Equation 16 – Cross-entropy error function for multiple classes.....	13

1 Reference Documents

REF. ID	TITLE	DATE	AUTHOR	COMMENTS
REF. 1	Neural networks for pattern recognition	First published 1995	Christoper M. Bishop	Page: 225-245 (Chapter 6)
REF. 2	Obtaining Accurate and Comprehensible Data Mining Models – An Evolutionary Approach	2007	Ulf Johansson	Page: 44-53 (Chapter 3)
REF. 3	ftp://ftp.sas.com/pub/neural/FAQ.html	1997	Sarle W.S.	

Tab. 1 – Reference Documents

2 Introduction

This document contains a description of the main characteristics of the softmax activation function and the cross-entropy error function. Furthermore the reasons for using both functions together is derived. Some basic terms are briefly reported and described. paragraph 3 and 4 describe the activation function and the 1-of-C coding scheme. This document does not provide a general introduction on Neural Networks and their main properties and characteristics

3 The activation function

The activation function can be either linear or non-linear, depending on whether the network must learn a regression problem or should perform a classification. Activation functions, for the hidden units, introduce the non linearity into the network. Without non linearity, the hidden units would not render the NN more powerful than just the plain perceptrons with only input and output units (the linear function of linear functions is again a linear function). In other words, it is the non linearity (i.e., the capability to represent non linear functions) that makes multilayer networks so powerful.

For the hidden units, sigmoid activation functions (for binary problems), see equation (2), or softmax (for multi class problem; see paragraph 5), are usually better to use instead of the threshold activation functions, see equation (1).

$$g(a) = \begin{cases} 0 & \text{when } a < 0 \\ 1 & \text{when } a \geq 0 \end{cases} \quad (1)$$

$$g(a) \equiv \frac{1}{1 + \exp(-a)} \quad (2)$$

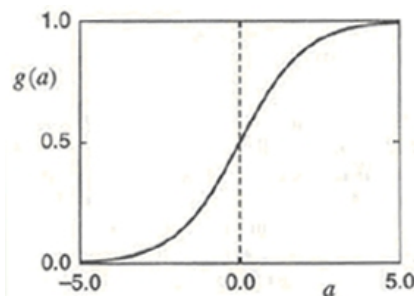


Figure 1: Plot of sigmoid function clearly shows that the outputs are between 0 and 1

Networks with threshold units are difficult to train, because the error function is stepwise constant, hence the gradient either does not exist or is zero, thus making it impossible to use back propagation (a powerful and computationally efficient algorithm for finding the derivatives of an error function with respect to the weights and biases in the network) or the more efficient gradient-based training methods.

With sigmoid units, a small change in the weights will usually produce a large change in the outputs, which makes it possible to tell whether that change in the weights is good or

useless. With threshold units, a small change in the weights will often produce no change in the outputs.

For the output units, activation functions suited to the distribution of the target values are:

- For binary (0/1) targets, the logistic sigmoid function is an excellent choice
- \emptyset For categorical targets using 1-of-C coding, the softmax activation function is the natural extension of the logistic function.
- For continuous-valued targets with a bounded range, the logistic and hyperbolic tangent functions can be used, where you either scale the outputs to the range of the targets or scale the targets to the range of the output activation function ("scaling" means multiplying by and adding appropriate constants).
- If the target values are positive but have no known upper bound, you can use an exponential output activation function, but you must beware of overflow.
- For continuous-valued targets with no bounds, use the identity or "linear" activation function (which amounts to no activation function) unless you have a very good reason to do otherwise.

There are certain natural associations between output activation functions and various noise distributions. The output activation function is the inverse of what statisticians call the "link function".

4 1-of-C classification

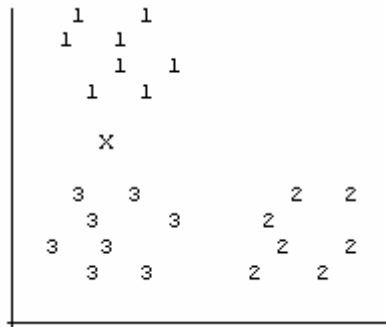
To classify targets into one out of C categories (i.e. you have a categorical target variable), 1-of-C coding is used. That means that you code C binary (0/1) target variables corresponding to the C categories. Statisticians call these "dummy" variables. The dummy variable corresponding to the correct category is given the value one and all the others have value zero. The use of a softmax output activation function ensures that the net, if properly trained, will produce valid posterior probability estimates (McCullagh and Nelder, 1989; Finke and Muller, 1994).

If the categories are Red, Green, and Blue, then the data would look like this:

Category	Dummy variables		
Red	1	0	0
Green	0	1	0
Blue	0	0	1

When there are only two categories, it is simpler to use just one dummy variable with a logistic sigmoid output activation function; this is equivalent to using softmax with two dummy variables.

It is very important not to use a single variable for an unordered categorical target. Suppose you used a single variable with values 1, 2, and 3 for red, green, and blue, and the training data with two inputs looked like this:



Consider a test point located at the X. The correct output would be that X has about a 50-50 chance of being a 1 or a 3. But if you train with a single target variable with values of 1, 2, and 3, the output for X will be the average of 1 and 3, so the net will say that X is definitely a 2!

For an input with categorical values, 1-of-(C-1) coding can be used if the network has a bias unit. This is just like 1-of-C coding, except that one of the dummy variables is omitted. Using all C dummy variables creates a linear dependency on the bias unit, which is not advisable unless you are using weight decay or Bayesian learning, which requires all C weights to be treated on an equal basis.

1-of-(C-1) coding looks like this:

Category	Dummy variables	
Red	1	0
Green	0	1
Blue	0	0

If you use 1-of-C or 1-of-(C-1) coding, it is important to standardize the dummy inputs; see "Should I standardize the input variables?" "Why not code binary inputs as 0 and 1?" for details.

If you are using weight decay, you want to make sure that shrinking the weights toward zero biases ('bias' in the statistical sense) the net in a sensible, usually smooth, way. If you use 1-of-(C-1) coding for an input, weight decay biases the output for the C-1 categories towards the output for the 1 omitted category, which is probably not what you want, although there might be special cases where it would make sense. If you use 1-of-C coding for an input, weight decay biases the output for all C categories roughly towards the mean output for all the categories, which is smoother and usually a reasonable thing to do.

5 Softmax activation function

In order to ensure that the outputs can be interpreted as posterior probabilities, they must be comprised between zero and one, and their sum must be equal to one. This constraint also ensures that the distribution is correctly normalized. In practice this is, for multi-class problems, achieved by using a softmax activation function in the output layer. The purpose of the softmax activation function is to enforce these constraints on the outputs. Let the

network input to each output unit be q_i , $i=1, \dots, c$, where c is the number of categories. Then the softmax output p_i is:

$$p_i = \frac{\exp(q_i)}{\sum_{j=1}^c \exp(q_j)} \quad (3)$$

Statisticians usually call softmax a "multiple logistic" function. Equation (3) is also known as normalized exponential function. It reduces to the simple logistic function when there are only two categories. Suppose you choose to set q_2 to 0.

$$p_1 = \frac{\exp(q_1)}{\sum_{j=1}^c \exp(q_j)} = \frac{\exp(q_1)}{\exp(q_1) - \exp(0)} = \frac{1}{1 + \exp(-q_1)} \quad (4)$$

The term softmax is used because this activation function represents a smooth version of the winner-takes-all activation model in which the unit with the largest input has output +1 while all other units have output 0.

The softmax function is also used in the hidden layer of normalized radial-basis-function networks, but in the interest of this document we would not enter into their description.

6 Cross-entropy error function

6.1 Cross-entropy for two classes

Learning in the neural networks is based on the definition of a suitable error function, which is then minimized with respect to the weights and biases in the network. Error functions play an important role in the use of neural networks. A variety of different error functions exist.

For regression problems the basic goal is to model the conditional distribution of the output variables, conditioned on the input variables. This motivates the use of a sum-of-squares error function. But for classification problems the sum-of-squares error function is not the most appropriate choice. In the case of a 1-of-C coding scheme, the target values sum to unity for each pattern and so the network outputs will also always sum to unity. However, there is no guarantee that they will lie in the range (0,1).

In fact, the outputs of the network trained by minimizing a sum-of-squares error function approximate the posterior probabilities of class membership, conditioned on the input vector, using the maximum likelihood principle by assuming that the target data was generated from a smooth deterministic function with added Gaussian noise. For classification problems, however, the targets are binary variables and hence far from having a Gaussian distribution, so their description can not be given by using Gaussian noise model.

Therefore a more appropriate choice of error function is needed.

Let us now consider problems involving two classes. One approach to such problems would be to use a network with two output units, one for each class. First let's discuss an alternative approach in which we consider a network with a single output y . We would like the value of y to represent the posterior probability $P(C_1|x)$ for class C_1 . The posterior probability of class C_2 will then be given by $P(C_2|x)=1-y$. This can be achieved if we consider a target coding scheme for which $t = 1$ if the input vector belongs to class C_1 and $t = 0$ if it belongs to class C_2 . We can combine these into a single expression, so that the probability of observing either target value is

$$p(t|x)=y^t(1-y)^{1-t} \quad (5)$$

This equation is the equation for a binomial distribution known as Bernoulli distribution. With this, interpretation of the output unit activations, the likelihood of observing the training data set, assuming the data points are drawn independently from this distribution, is then given by

$$\prod_n (y^n)^{t^n} (1-y^n)^{1-t^n} \quad (6)$$

By minimizing the negative logarithm of the likelihood we get to the cross-entropy error function (Hopfield, 1987; Baum and Wilczek, 1988; Solla et al., 1988; Hinton, 1989; Hampshire and Pearlmutter, 1990) in the form

$$E=-\sum \{t^n \ln y^n + (1-t^n) \ln (1-y^n)\} \quad (7)$$

Let's consider some elementary properties of this error function. Differentiating this error function with respect to y^n we obtain

$$\frac{\partial E}{\partial y^n} = \frac{(y^n - t^n)}{y^n(1-y^n)} \quad (8)$$

The absolute minimum of the error function occurs when

$$y^n = t^n \quad \forall n \quad (9)$$

The considering network has one output whose value is to be interpreted as a probability, so it is appropriate to consider the logistic sigmoid activation function, equation (2), which has the property

$$g'(a) = g(a)(1-g(a)) \quad (10)$$

Combining equations (8) and (10) we see that the derivative of the error with respect to a takes a simple form

$$\delta^n \equiv \frac{\partial E}{\partial a^n} = y^n - t^n \quad (11)$$

This equation gives the error quantity which is back propagated through the network in order to compute the derivatives of the error function with respect to the network weights. The same equation form can be obtained for the sum-of-squares error function and linear output units. This shows that there is a natural pairing of error function and output unit activation function.

From the equations (7) and (9) the value of the cross entropy error function at its minimum is given by

$$E_{min} = - \sum \{ t^n \ln t^n + (1-t^n) \ln(1-t^n) \} \quad (12)$$

The last equation becomes zero for 1-of-C coding scheme. However, when t^n is a continuous variable in the range (0,1) representing the probability of the input vector x^n belonging to class C, the error function (7) is also the correct one to use, In this case the minimum value (12) of the error does not become 0. In this case it is appropriate by subtracting this value from the original error function to get a modified error function of the form

$$E = - \sum_n \left\{ t^n \ln \frac{y^n}{t^n} + (1-t^n) \ln \frac{(1-y^n)}{(1-t^n)} \right\} \quad (13)$$

But before moving to cross-entropy for multiple classes let us describe more in detail its properties. Assume the network output for a particular pattern n , written in the form $y^n = t^n + \epsilon^n$. Then the cross-entropy error function (13) can be transformed to the form

$$E = - \sum_n \left\{ t^n \ln \left(1 + \frac{\epsilon^n}{t^n} \right) + (1-t^n) \ln \left(1 - \frac{\epsilon^n}{1-t^n} \right) \right\} \quad (14)$$

so that the error function depends on the relative errors of the network outputs. Knowing that the sum-of-squares error function depends on the squares of the absolute errors, we can make comparisons. Minimization of the cross-entropy error function will tend to result in similar relative errors on both small and large target values. By contrast, the sum-of-squares error function tends to give similar absolute errors for each pattern, and will give large relative errors for small output values. This result suggests the better functionality of the cross-entropy error function over the sum-of-squares error function at estimating small probabilities. Another advantage over the sum-of-squares error function, is that the cross-entropy error function gives much stronger weight to smaller errors.

6.2 Cross-entropy for multiple classes

Let's return to the classification problem involving mutually exclusive classes, where the number of classes is greater than two. For this problem we should seek the form which the error function should take. The network now has one output y_k , for each class, and target data which has a 1-of-c coding scheme, so that we have $t_k^n = b_{kl}$ for a pattern n from class C_l . The probability of observing the set of target values $t_k^n = b_{kl}$, given an input vector x^n , is just $p(C_l|x) = y_l$. Therefore the conditional distribution for this pattern can be written as

$$p(t^n|x^n) = \prod_{k=1}^c (y_k^n)^{t_k^n} \quad (15)$$

As before, starting from the likelihood function, by taking the negative logarithm, we obtain an error function of the form

$$E = - \sum_n \sum_{k=1}^c t_k^n \ln y_k^n \quad (16)$$

For 1-of-C coding scheme the minimum value of the error function (16) equals 0. But the error function is still valid when t_k^n is a continuous variable in the range (0,1) representing the probability that x^n belongs to C_k . To get the proper target variable the softmax activation function is used. So for the cross-entropy error function for multiple classes, equation (16), to be efficient the softmax activation function must be used.

By evaluating the derivatives of the softmax error function considering all inputs to all output units, (for pattern n) it can be obtained

$$\frac{\partial E^n}{\partial a_k} = y_k - t_k \quad (17)$$

which is the same result as found for the two-class cross-entropy error (with a logistic activation function), equation (11). The same result is valid for the sum-of-squares error (with a linear activation function).

This can be considered as an additional proof that there is a natural pairing of error function and activation function.

7 Conclusion

Referring to the last results we see the correctness of the expression “pairing of error function and activation function”.

As mentioned before for every activation function we get a proper error function, and as shown for the soft max activation function we must use the cross-entropy error function.

It is obvious that by using non proper pairs of activation and error function the network would not perform as we would like to, giving results without sense.

__oOo__