Neural Networks with Adaptive Activation Functions

Ilaria Castelli

Università degli Studi di Siena, Via Roma 56, 53100 Siena, Italy

e-mail: castelli@dii.unisi.it

Tutor: Edmondo Trentin

Introduction

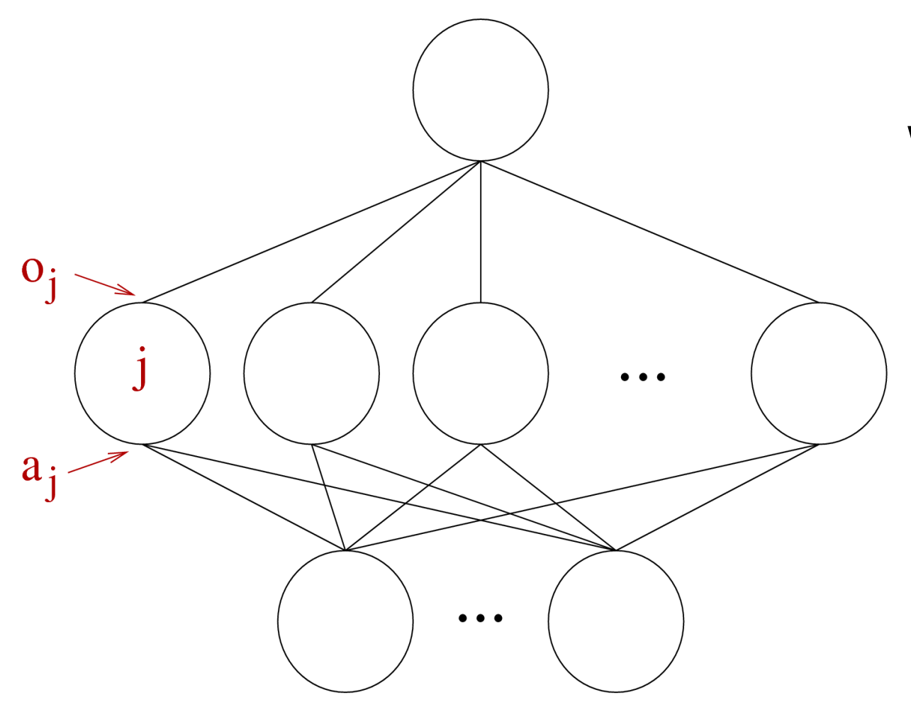
In the most common framework for training neural networks in a supervised setting, we are given a training set $D=\{(x^k,y^k):k=1,\dots,N\}$ of examples sampled from an unknown distribution $p(X,Y)$, and we want to minimize the expected value of a loss function under $p(X,Y)$. Neural networks are usually trained through a gradient descent procedure [1] to minimize

$$L(f(W),(X,Y)) \;=\; \frac{1}{2}\sum_{k=1}^{N}\left(y^k-\tilde{f}(x^k;W)\right)^2$$

and, since *D* is given, this reduces in minimizing

$$C(W) \;=\; \frac{1}{2}\sum_{k=1}^{N}\left(y^k-\tilde{f}^k(W)\right)^2$$

w.r.t. the network weights *W*.



Input patterns are forwarded through the network, up to the output units, computing $\tilde{f}(W)$.

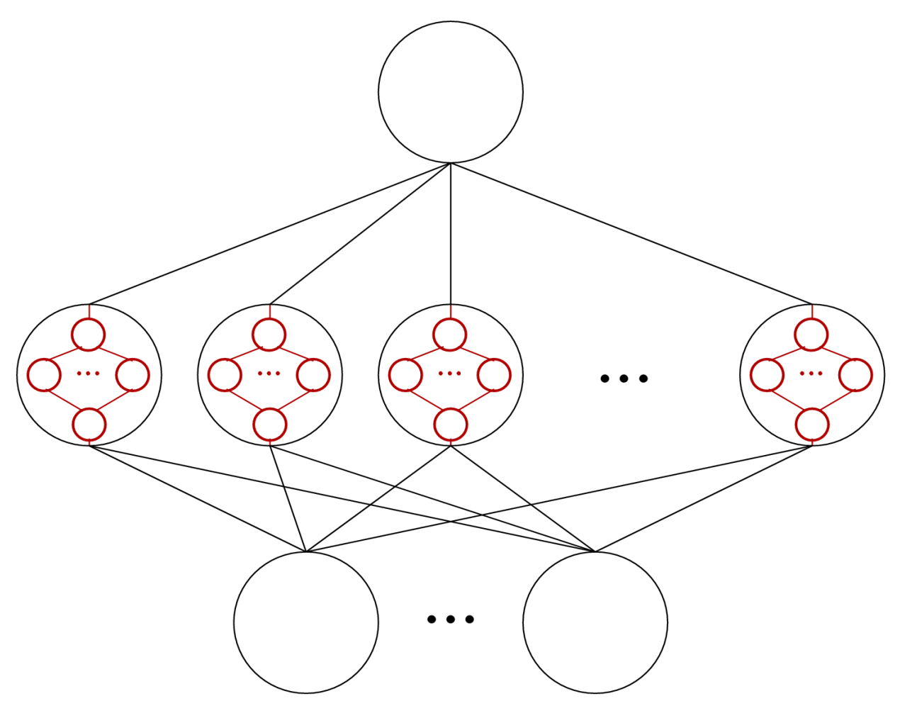
Unit activation $a_j=\sum_{u\in L_{l-1}} o_u * w_{ju}$

Unit output $o_j=f_j(a_j)$

Neural activation functions: motivation

According to [3], MLPs with one sigmoidal hidden layer are universal function approximator, i.e. $\forall \varepsilon>0, \forall h:R^p\rightarrow R^m, \forall x\in R^p \;:\; \|\tilde{f}(x,W)-h(x)\|<\varepsilon$

Although the theorem guarantees the existence of a network able to approximate any function, from a practical point of view, the number of nodes required in the hidden layer may grow very large, even exponentially in the number of inputs [4]. This may lead to big architectures, difficult to train and prone to stuck into poor local minima.



The idea of this work is to reduce the training complexity of the networks using a small number of hidden units, but allowing them to compute a more flexible activation function. **Our activation functions are implemented by neural networks** and adaptively learned during training.

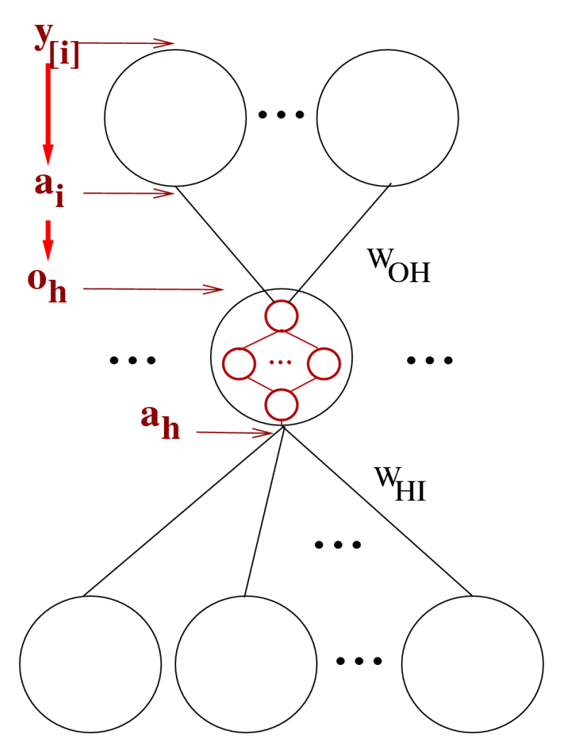
Hidden units training

Each hidden activation function f_h is trained on a dataset $D^h=\{(x_h^k,y_h^k):k=1,\dots,N\}$ obtained propagating the inputs and the outputs from the external network:

- $x_h^k \;=\; a_h \;=\; \sum_{u\in L_0} o_u * w_{hu} \;=\; \sum_{u\in L_0} x_{[u]}^k * w_{hu}$

where $x_{[u]}^k$ is the *u*-th entry of input x^k.

- Computing y_h^k the following condition should be satisfied for each output unit *i:*



$$o_i \;=\; f_i(a_i) \;=\; f_i\left(\sum_{h=1}^{m} y_h^k * w_{ih}\right) = y_{[i]}^k$$

This can be achieved:

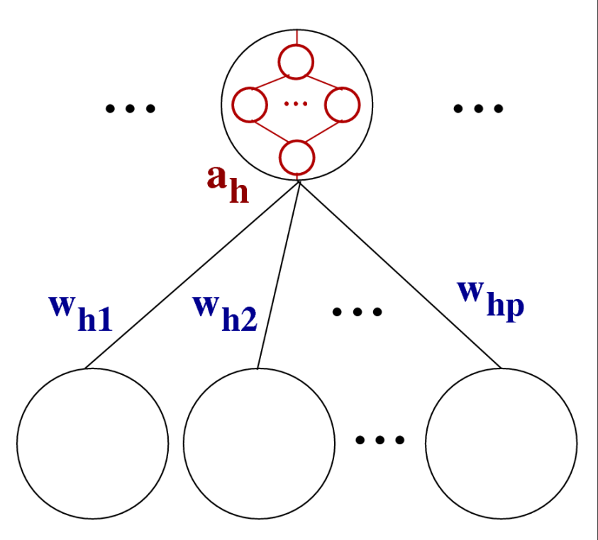
- through gradient descent
- through inversion of the weight matrix W_{OH} : $O_H=W_{OH}^{-1}A_O$

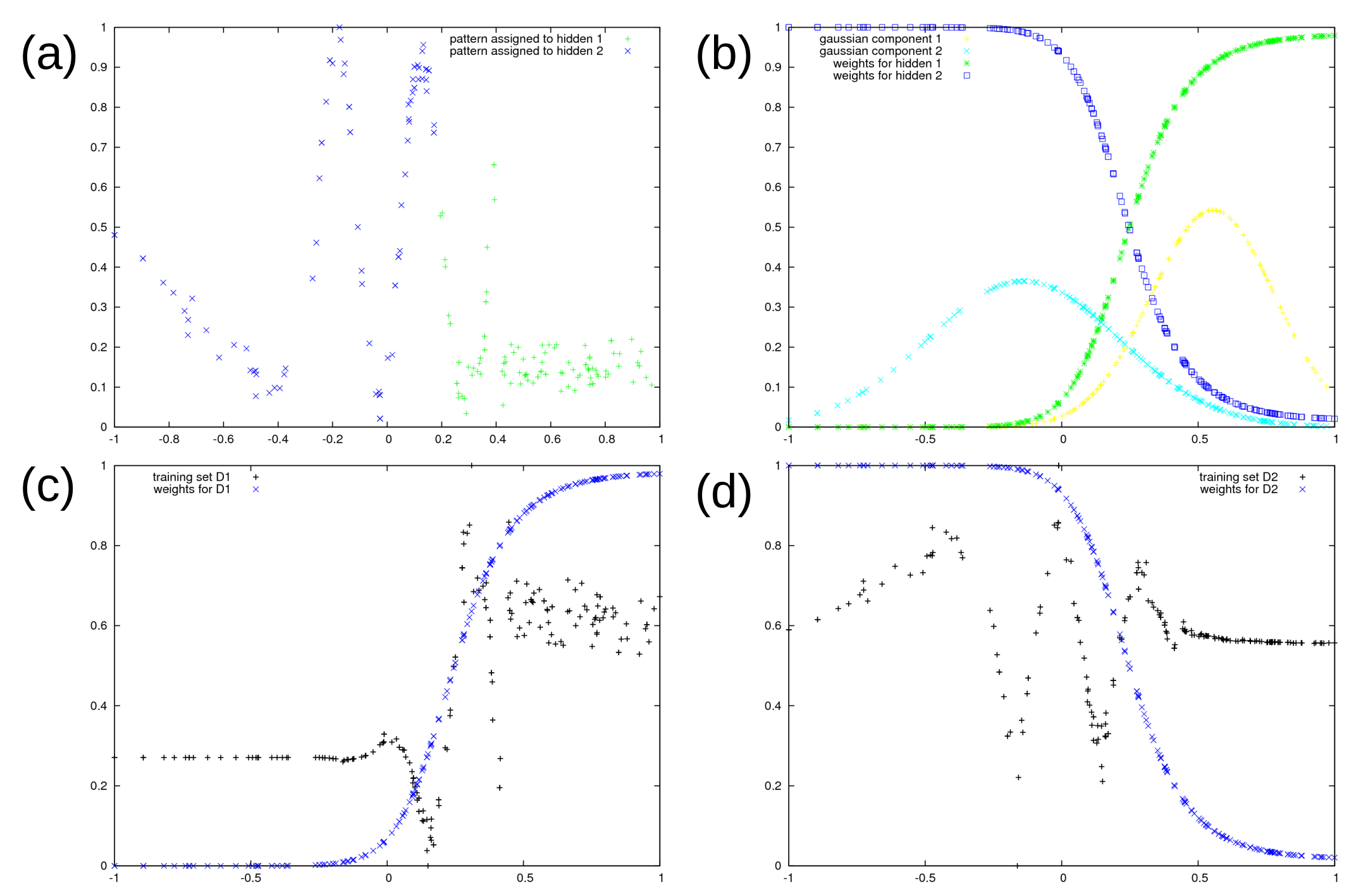
Probabilistic pattern weighting

- $\forall h=1,\dots,m$ and $\forall x^k$, we compute $P(\omega_h|x^k)$, i.e. the probability for hidden unit *h* of being competent on pattern x^k
- The *h*-th hidden unit is trained on D_h to minimize
$$C_h(W_h) \;=\; \frac{1}{2}\sum_{k=1}^{N} P(\omega_h|x^k)\left(y_h^k-\tilde{f}_h^k(W_h)\right)^2$$
- Every hidden unit specializes on a part of the problem
- The estimate $P(\omega_h|x^k)$ is used to weight the contribute of *h* to the activation of the *i*-th output unit:
$$a_i \;=\; \sum_{h=1}^{m} o_h w_{ih} P(\omega_h|x^k) \;=\; \sum_{h=1}^{m} o_h \tilde{w}_{ih}$$

$P(\omega|x^k)$ is computed through a maximum likelihood density estimation joint over the input pattern projected in hidden spaces and the target output

$$p((x^k,y^k)|\theta) \;=\; \sum_{h=1}^{m} P(\omega_h)\, p((w_h x^k,y^k)|\omega_h,\theta_h)$$





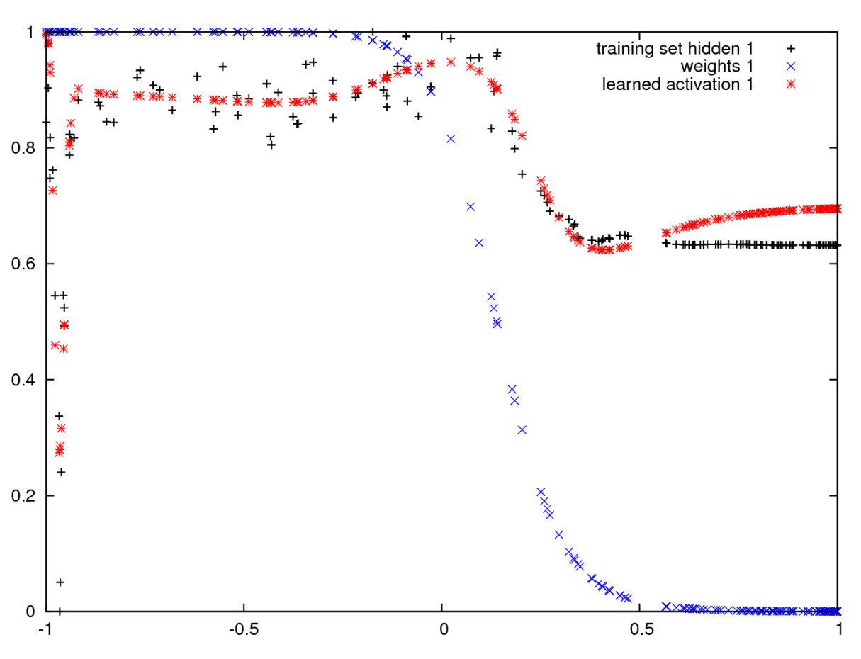
(a) pattern assigned to each hidden after ML estimation

(b) likelihood mixture components and associated posterior weights

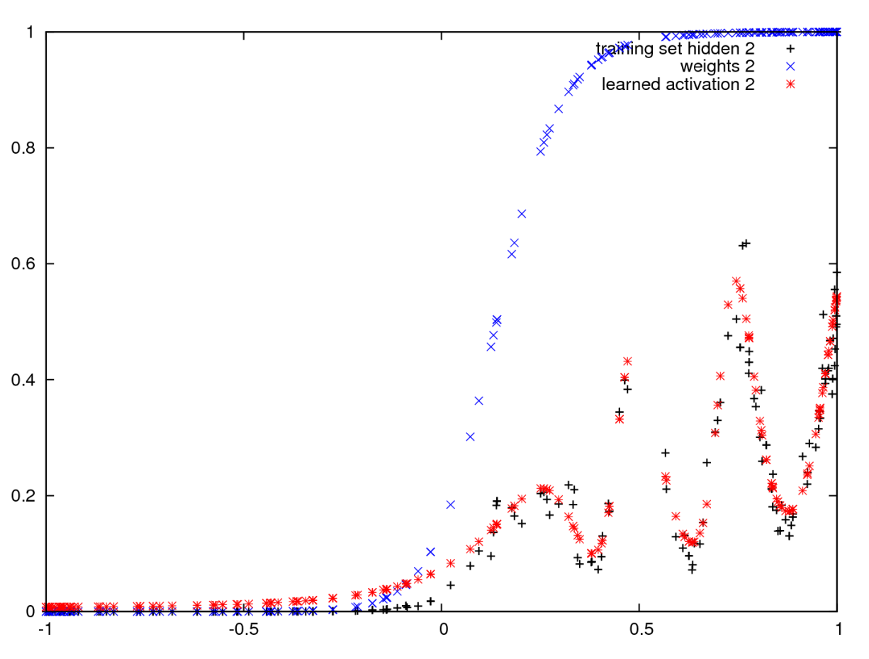
(c) weighted training set for 1st hidden unit

(d) weighted training set for 2nd hidden unit

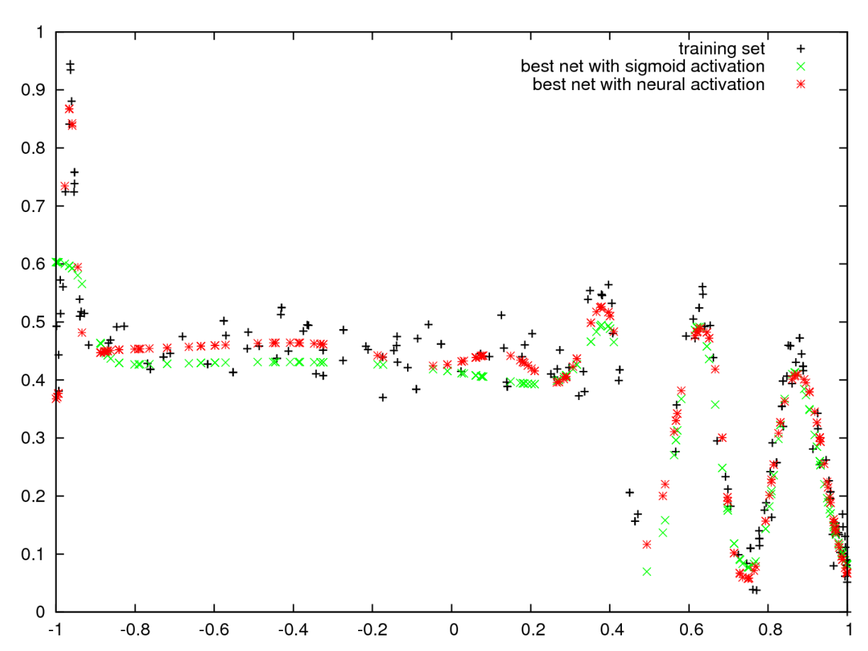
A regression example



(e) activation learned for 1st unit



(f) activation learned for 2nd unit



(g) comparison against classical MLP

MLP with sigmoidal activation f.				MLP with adaptive activation f.			
#Param	#Hidden	ISE test	MSE test	#Param	#Hidden	ISE test	MSE test
40	13	0.00137	0.037	40	2 (5-6)	9.25e-4	0.035
49	16	0.00390	0.052	37	2 (4-6)	9.42e-4	0.035
46	15	0.00393	0.052	34	2 (4-5)	9.55e-4	0.038

Both classical MLPs and MLPs with adaptive activation functions, have been trained using the same hyper-parameters (i.e. learning rate, sigmoid smoothness, total number of training epochs)

Ongoing work

- Evaluation of this model for classification problems
- Recursive unrolling of hidden units in order to expand the network in depth
- An alternative way of dealing with the difficulties in training networks that require a large number of hidden units has been recently proposed [5], and consists in using *"deep architectures".* I'm currently exploiting such deep models to understand their disentangling capabilities in difficult classification problems (i.e. related to face analysis, in large scale settings with millions of images)

References

[1] Christopher M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, USA, 1996.
[2] Richard O. Duda, Peter E. Hart, and David G. Stork. Pattern Classification. Wiley, New York, 2nd edition, 2001.
[3] G. Cybenko. Approximation by Superposition of Sigmoidal Functions. Mathematics of Control, Signals and Systems, vol 2, pp. 303-314, 1989.
[4] Johan Håstad. Almost Optimal Lower Bounds for Small Depth Circuits. In Proceedings of the 18th annual ACM Symposium on Theory of Computing, 1986.
[5] Yoshua Bengio. Learning Deep Architectures for AI. Foundations and Trends in Machine Learning, 2009.