

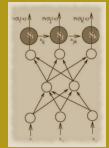
Multilayer Perceptron (MLP): the Backpropagation (BP) Algorithm

Guest Speaker: Edmondo Trentin

Dipartimento di Ingegneria dell'Informazione
Università di Siena, V. Roma, 56 - Siena (Italy)

{trentin}@dii.unisi.it

October 7, 2008



[Title Page](#)



[Page 1 of 26](#)

[Go Back](#)

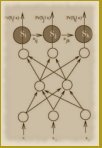
[Full Screen](#)

[Close](#)

[Quit](#)

Representation of Inputs (Patterns)

- In order to carry out the learning task, we need to extract a digital representation \mathbf{x} of any given object (/events) that has to be fed into the MLP
- \mathbf{x} is called a *pattern*
- \mathbf{x} is real-valued (i.e. $\mathbf{x} \in \mathcal{R}^d$):
 - \mathbf{x} is also known as “feature vector”
 - The components of \mathbf{x} are known as the *features*
 - d is the dimensionality of the *feature space* \mathcal{R}^d
- The (problem-specific) process of extracting representative features x_1, \dots, x_d is known as *fea-*



Title Page



Page 2 of 26

Go Back

Full Screen

Close

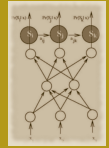
Quit

ture extraction. It should satisfy two requirements:

1. \mathbf{x} contains (most of) the information needed for the learning task
2. d is as small as possible

Further processing steps, if needed:

- *Feature selection/reduction* (e.g. Principal Component Analysis) may reduce the dimensionality, preserving only the “relevant” information
- *Normalization* (/standardization) transforms the feature values into homogeneous and well-behaved values that yield numerical stability



Title Page



Page 3 of 26

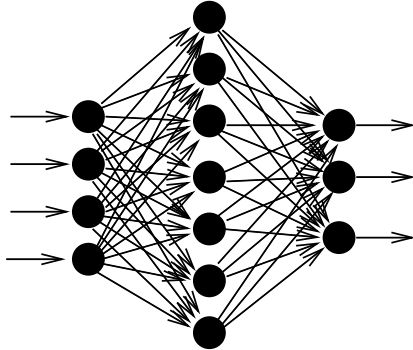
Go Back

Full Screen

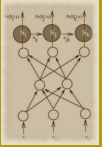
Close

Quit

Multilayer Perceptron (MLP)



- Feedforward (and full) connections between pairs of adjacent *layers*
- Continuous and differentiable activation functions
- Realizes a multidimensional function $\mathbf{y} = \varphi(\mathbf{x})$ between input $\mathbf{x} \in \mathcal{R}^{d_i}$ and output $\mathbf{y} \in \mathcal{R}^{d_o}$



Title Page



Page 4 of 26

Go Back

Full Screen

Close

Quit

MLP: Dynamics (forward propagation)

- Each unit realizes a transformation of the signal via application of its activation function $f(\cdot)$ to its argument a .
- The argument a is obtained as a weighted sum of the signals that feed the neuron through the incoming connections, i.e. $a = \sum_k w_k z_k$ where w_k is the weight associated with k -th connection, and z_k is the k -th component of the signal (either input signal, or yield by other neurons in the MLP).



[Title Page](#)



Page 5 of 26

[Go Back](#)

[Full Screen](#)

[Close](#)

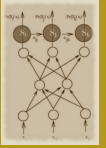
[Quit](#)

MLP: Learning

A learning rule is applied in order to improve the value of the MLP weights over a training set \mathcal{T} according to a given *criterion function*.

MLP: Generalization

The MLP must infer a general law from \mathcal{T} (a raw memorization of the training examples is not sought!) that, in turn, can be applied to novel data that are distributed according to the same probability laws. *Regularization techniques* help improving generalization capabilities.



[Title Page](#)



Page 6 of 26

[Go Back](#)

[Full Screen](#)

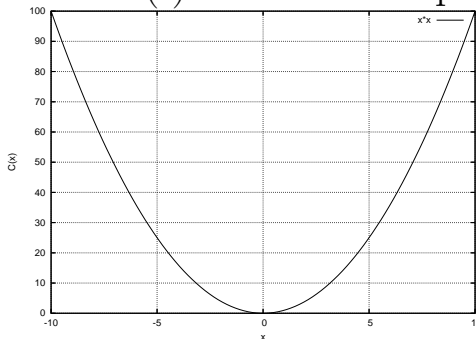
[Close](#)

[Quit](#)

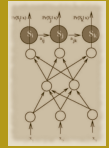
MLP Training: the Idea

Given an example (\mathbf{x}, \mathbf{y}) , modify the weights w s.t. the output $\hat{\mathbf{y}}$ yielded by the MLP (when fed with input \mathbf{x}) gets **closer** to the target \mathbf{y} .

- Criterion function $C(\cdot)$: minimum squared error $(y - \hat{y})^2$



- Advantages:
 - Convex & Non-negative (search for minimum)
 - Penalizes large errors
 - Differentiable (**gradient-descent** is viable)



Title Page



Page 7 of 26

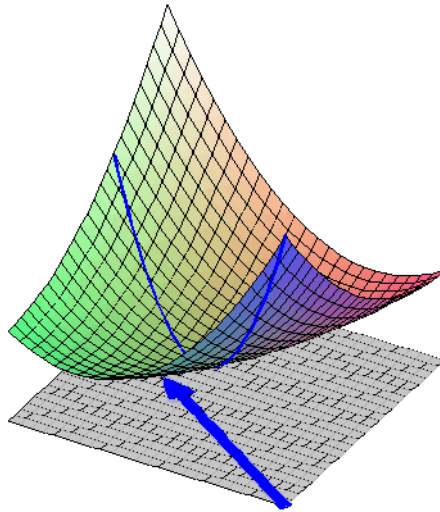
Go Back

Full Screen

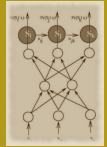
Close

Quit

Gradient Descent



The criterion $C(\cdot)$ is a function of the MLP weights w . Method: iterate slight modification of the weights in order to move in the opposite way w.r.t. the gradient (steepest direction).



[Title Page](#)



Page 8 of 26

[Go Back](#)

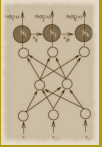
[Full Screen](#)

[Close](#)

[Quit](#)

Backpropagation Algorithm (BP)

- Labeled (*supervised*) training set: $T = \{(\mathbf{x}_k, \mathbf{y}_k) \mid k = 1, \dots, N\}$
- Online **criterion function**: $C = \frac{1}{2} \sum_{n=1}^{d_o} (y_n - \hat{y}_n)^2$ where \hat{y}_n is n -th MLP output
- Weight-update rule: $\Delta w_{ij} = -\eta \frac{\partial C}{\partial w_{ij}}$ (*Note*: w_{ij} is the connection weight between j -th unit in a given layer and i -th unit in the following layer)
- Activation function for i -th unit: $f_i(a_i)$, where:
 - $f_i : \mathcal{R} \rightarrow \mathcal{R}$
 - $a_i = \sum_j w_{ij} f_j(a_j)$ is the input to i -th unit (*Note*: the sum is extended to all the units in the previous layer)



Title Page



Page 9 of 26

Go Back

Full Screen

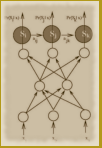
Close

Quit

BP Case 1: i is in the output layer

$$\begin{aligned}\frac{\partial C}{\partial w_{ij}} &= \frac{1}{2} \sum_{n=1}^{d_o} \frac{\partial (y_n - \hat{y}_n)^2}{\partial w_{ij}} & (1) \\ &= \frac{1}{2} \frac{\partial (y_i - \hat{y}_i)^2}{\partial w_{ij}} \\ &= -(y_i - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial w_{ij}}\end{aligned}$$

$$\begin{aligned}\frac{\partial \hat{y}_i}{\partial w_{ij}} &= \frac{\partial f_i(a_i)}{\partial w_{ij}} & (2) \\ &= \frac{\partial f_i(a_i)}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} \\ &= f'_i(a_i) \frac{\partial \sum_l w_{il} \hat{y}_l}{\partial w_{ij}} \\ &= f'_i(a_i) \hat{y}_j\end{aligned}$$



Title Page



Page 10 of 26

Go Back

Full Screen

Close

Quit

where the sum over l is extended to all the units in the (first) hidden layer.

From Eqs. (1) and (2) we have:

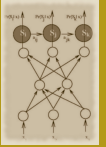
$$\frac{\partial C}{\partial w_{ij}} = -(y_i - \hat{y}_i) f'_i(a_i) \hat{y}_j \quad (3)$$

We define:

$$\delta_i = (y_i - \hat{y}_i) f'_i(a_i) \quad (4)$$

We substitute it into Eq. (3), and we can (finally) write:

$$\Delta w_{ij} = \eta \delta_i \hat{y}_j \quad (5)$$



Title Page



Page 11 of 26

Go Back

Full Screen

Close

Quit

BP Case 2: unit j in the (topmost) hidden layer

Let w_{jk} be the weight between k -th unit in the previous layer (either hidden, or input layer) and j -th unit in the topmost hidden layer:

$$\Delta w_{jk} = -\eta \frac{\partial C}{\partial w_{jk}} \quad (6)$$

Again:

$$\begin{aligned} \frac{\partial C}{\partial w_{jk}} &= \frac{1}{2} \sum_{n=1}^{d_o} \frac{\partial (y_n - \hat{y}_n)^2}{\partial w_{jk}} \quad (7) \\ &= - \sum_{n=1}^{d_o} (y_n - \hat{y}_n) \frac{\partial \hat{y}_n}{\partial w_{jk}} \end{aligned}$$

where:



Title Page



Page 12 of 26

Go Back

Full Screen

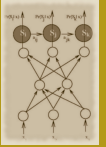
Close

Quit

$$\begin{aligned}
\frac{\partial \hat{y}_n}{\partial w_{jk}} &= \frac{\partial f_n(a_n)}{\partial w_{jk}} & (8) \\
&= \frac{\partial f_n(a_n)}{\partial a_n} \frac{\partial a_n}{\partial w_{jk}} \\
&= f'_n(a_n) \frac{\partial a_n}{\partial w_{jk}}
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial a_n}{\partial w_{jk}} &= \frac{\partial \sum_l w_{nl} \hat{y}_l}{\partial w_{jk}} & (9) \\
&= \sum_l w_{nl} \frac{\partial \hat{y}_l}{\partial w_{jk}} \\
&= w_{nj} \frac{\partial \hat{y}_j}{\partial w_{jk}}
\end{aligned}$$



Title Page



Page 13 of 26

Go Back

Full Screen

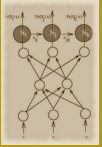
Close

Quit

(where, again, the sum over l is extended to all the units in the topmost hidden layer). In turn:

$$\begin{aligned}\frac{\partial \hat{y}_j}{\partial w_{jk}} &= \frac{\partial f_j(a_j)}{\partial w_{jk}} & (10) \\ &= \frac{\partial f_j(a_j)}{\partial a_j} \frac{\partial a_j}{\partial w_{jk}} \\ &= f'_j(a_j) \frac{\partial \sum_m w_{jm} x_m}{\partial w_{jk}} \\ &= f'_j(a_j) x_k.\end{aligned}$$

(of course the sum over m extends over all the units in the previous layer w.r.t. j).



Title Page



Page 14 of 26

Go Back

Full Screen

Close

Quit

Substituting eqs. (7), (8), (9) and (10) into equation (6) we obtain:

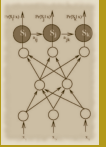
$$\begin{aligned}
 \Delta w_{jk} &= \eta \sum_n [(y_n - \hat{y}_n) f'_n(a_n) w_{nj}] f'_j(a_j) x_k \quad (11) \\
 &= \eta \left\{ \sum_n [w_{nj} (y_n - \hat{y}_n) f'_n(a_n)] \right\} f'_j(a_j) x_k \\
 &= \eta \left(\sum_n w_{nj} \delta_n \right) f'_j(a_j) x_k
 \end{aligned}$$

where δ_n is defined by Eq. (4). We define:

$$\delta_j = \left(\sum_n w_{nj} \delta_n \right) f'_j(a_j) \quad (12)$$

and Eq. (11) becomes

$$\Delta w_{jk} = \eta \delta_j x_k \quad (13)$$



Title Page



Page 15 of 26

Go Back

Full Screen

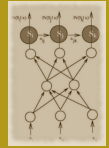
Close

Quit

which is known as the BP *delta rule*, i.e. a compact expression of the BP algorithm itself which captures the idea of top-down *backpropagation of deltas* throughout the MLP.

The delta rule holds also for the other layers in the ANN (proof is easy, by induction on the number of layers).

- The rule is applied one example at a time, over the whole training set. A complete cycle is known as an *Epoch*. Many epochs are required in order to accomplish the ANN training.
- Popular choices for the activation functions: linear ($f(a) = a$) and *sigmoid* ($f(a) = \frac{1}{1+e^{-a}}$)
- The technique suffers from *local minima*



Title Page



Page 16 of 26

Go Back

Full Screen

Close

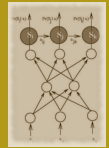
Quit

Universal property of MLPs

Theorems (independently proved by Lippmann, Cybenko and others) state that for any given continuous and limited function $\phi : \mathcal{R}^{d_i} \rightarrow \mathcal{R}^{d_o}$, a MLP with a single hidden layer with sigmoid units exists which approximates $\phi(\cdot)$ arbitrarily well.

These are *existence* theorems, that is to say they stress the flexibility of MLPs but:

1. they do not tell us which architecture is the “right” one for a given $\phi(\cdot)$ (i.e., for any given task)
2. even if the right topology were known, they do not tell us anything about the practical convergence of the BP algorithm to the “right” weight values



Title Page



Page 17 of 26

Go Back

Full Screen

Close

Quit

MLP for Pattern Classification

What is the relation (if any) between MLPs and Bayesian pattern classification (e.g., speech recognition, OCR)?

The answer comes from theorems independently proved by Bourlard, Cybenko and others:

- Let us consider a classification problem involving c classes $\omega_1, \dots, \omega_c$, and a supervised training sample $\mathcal{T} = \{(\mathbf{x}_i, \omega(\mathbf{x}_i)) \mid i = 1, \dots, N\}$ (where $\omega(\mathbf{x}_i)$ denotes the class which pattern \mathbf{x}_i belongs to)



[Title Page](#)



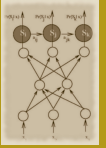
[Page 18 of 26](#)

[Go Back](#)

[Full Screen](#)

[Close](#)

[Quit](#)



- Let us create a MLP-oriented training set \mathcal{T}' from \mathcal{T} as follows: $\mathcal{T}' = \{(\mathbf{x}_i, \mathbf{y}_i) \mid i = 1, \dots, N\}$ where $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,c}) \in \mathcal{R}^c$ and

$$y_{i,j} = \begin{cases} 1.0 & \text{if } \omega_j = \omega(\mathbf{x}_i) \\ 0.0 & \text{otherwise} \end{cases} \quad (14)$$

(i.e., \mathbf{y}_i has null components, except for the one which corresponds to the *correct class*)

- Then (theorem), training a MLP over \mathcal{T}' is equivalent to training it over the training set $\{(\mathbf{x}_i, (P(\omega_1 \mid \mathbf{x}_i), P(\omega_2 \mid \mathbf{x}_i), \dots, P(\omega_c \mid \mathbf{x}_i))) \mid i = 1, \dots, N\}$ although, in general, we do not know $P(\omega_1 \mid \mathbf{x}_i), P(\omega_2 \mid \mathbf{x}_i), \dots, P(\omega_c \mid \mathbf{x}_i)$ in advance.

Title Page



Page 19 of 26

Go Back

Full Screen

Close

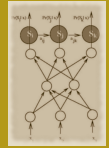
Quit

In so doing, we can train a MLP to estimate Bayesian posterior probabilities without even knowing them on the training sample. Due to the universal property, the nonparametric estimate that we obtain may be “optimal”.

Practical issues:

On real-world data, the following problems usually **prevent the MLP from reaching the optimal solution:**

1. Choice of the architecture (i.e., number of hidden units)
2. Choice of η and of the number of training epochs
3. Random initialization of connection weight
4. BP gets stuck into local minima



[Title Page](#)



Page 20 of 26

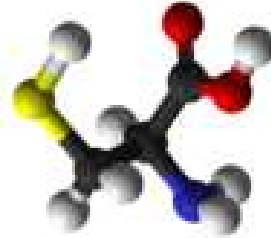
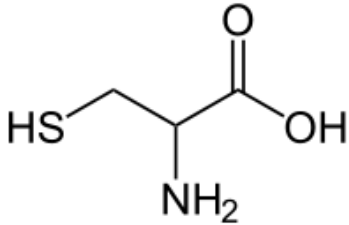
[Go Back](#)

[Full Screen](#)

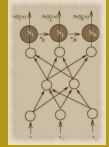
[Close](#)

[Quit](#)

Example: prediction of disulfide binding state



- **Cysteines** (*C* or *Cys*) are α -amino acids
- (Standard) α -amino acids are molecules which differ in their “residue”: via condensation, chains of residues form *proteins*
- The linear sequence of residues is known as the *primary structure* of the protein
- Cysteines play a major role in *structural* and *functional* properties of proteins, due to the high reactivity of their side-chain



[Title Page](#)



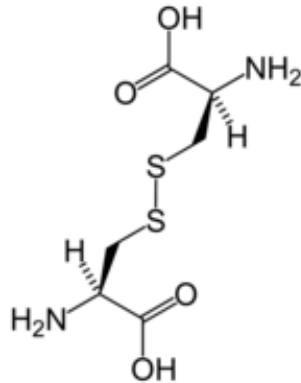
Page 21 of 26

[Go Back](#)

[Full Screen](#)

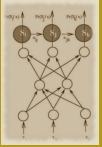
[Close](#)

[Quit](#)



- Oxidation of a pair of cysteines form a new molecule called *Cystine* via a (-S-S-) disulfide bond
- The disulfide bond has an impact on protein folding: (a) it holds two portions of the protein together; (b) it stabilizes the secondary structure

Prediction of the binding state of *Cys* within the primary structure of a protein would provide information on the secondary and tertiary structures.



Title Page



Page 22 of 26

Go Back

Full Screen

Close

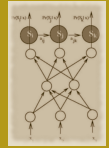
Quit

Classification task: predict the binding state ($\omega_1 =$ *bond*, $\omega_2 =$ *no bond*) of any given cysteine within the protein primary structure.

We use a dataset of sequences, e.g. the Protein Data-bank (PDB) which consists of more than 1,000 sequences, and we apply a supervised approach:

QNFITSKHNIDKIMT**C**NIRLNE**C**HDNIFEI**C**GS GK...
GHFTLELV**C**QRNFVTAIEIDHKLKT TENKLVDH**C**DN...
LNKDILQFKFPNSYKIFGN**C**IPYNIS**C**TDIRVFDS...

Part of the dataset is used for training, another (non-overlapping) part is used for validation (i.e., tuning of the model parameters) and test (i.e., evaluation of the generalization performance in terms of estimated probability of error).



Title Page



Page 23 of 26

Go Back

Full Screen

Close

Quit

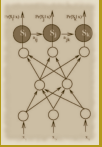
We are faced with 2 problems:

1. We cannot classify on the basis of an individual cysteine only, since $P(\omega_i | C)$ is just the prior $P(\omega_i)$. Information from the *sequence* is needed, but the sequence is long and may have variable length, while statistical models and MLP require a fixed-dimensionality feature space.

- Solution: we take fixed-size windows (i.e., subsequences) centered in the cysteine at hand:

QNFHNIDKIMTCNIRSKLNECHDNIFEICGSGK...

The window might contain from 11 to 31 amino-acids. An overlap between adjacent windows is allowed, i.e. a cysteine may become part of the window of another cysteine.



Title Page



Page 24 of 26

Go Back

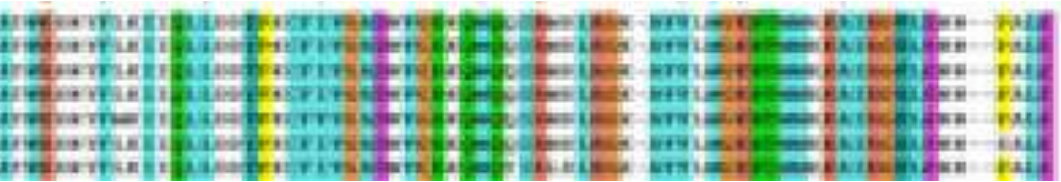
Full Screen

Close

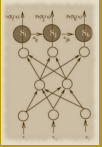
Quit

2. We cannot feed the MLP with symbols (namely, the literals of the amino-acids): a *coding* procedure is required.

- Solution: profiles of multiple alignment among *homologous* (i.e., similar) proteins



In so doing, a sequence of 20-dim real vectors $\mathbf{x}_1, \dots, \mathbf{x}_T$ is obtained, where $x_{t,i}$ is the probability (relative frequency) of observing i -th amino-acid in t -th position within the sequence.



Title Page



Page 25 of 26

Go Back

Full Screen

Close

Quit

MLP solution to the classification problem:

- Let us assume that the amino-acid in t -th position is a cysteine. The window centered in this *Cys* is now defined as $\mathbf{W} = (\mathbf{x}_{t-k}, \dots, \mathbf{x}_t, \dots, \mathbf{x}_{t+k})$ for a certain k , that is a $20 \times (2k+1)$ -dimensional real-valued vector
- A training set $\mathcal{T} = \{(\mathbf{W}, \omega(\mathbf{W}))\}$ is created, where $\omega(\mathbf{W})$ is either 0 or 1 according to the binding state (i.e., no bond, bond) of the corresponding cysteine
- A 1-output MLP (6 hidden units) is trained on \mathcal{T} via BP (6 epochs) to estimate $P(\omega_i | \mathbf{W})$
- **Results:** 19.36% *error rate* (which is an estimate of the probability of error)



Title Page



Page 26 of 26

Go Back

Full Screen

Close

Quit