LECTURE 11

Let us consider the following problem:

$\bar{X}_t \in \mathcal{R}^p$ - vector of $p \geq 1$ observations at time $t \geq 0$

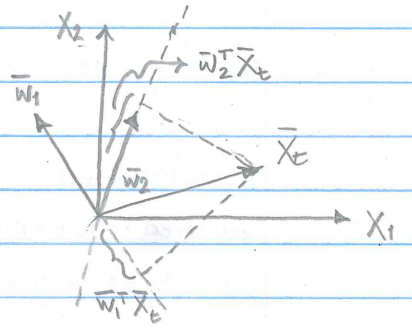
$\mathcal{C}_1 \subseteq \mathcal{R}^p$
$\mathcal{C}_2 \subseteq \mathcal{R}^p$
$\vdots$
$\mathcal{C}_J \subseteq \mathcal{R}^p$

$J \geq 2$ classes of vectors in $\mathcal{R}^p$ with $\bigcap_{j=1}^{J} \mathcal{C}_j = \emptyset$

We must decide a rule that assigns $\bar{X}_t$ to any of the $J \geq 2$ classes $\forall t$ and is optimal in some sense (CLASSIFICATION problem)

Note that, for a generic vector $\bar{w} \in \mathcal{R}^p$, the following facts hold:

- $v_t \triangleq \bar{w}^T \bar{X}_t$ (i.e., the projection of $\bar{X}_t$ along the direction of $\bar{w}$) may "enhance" a subset of components or features of $\bar{X}_t$. Ex.: $\bar{w} = (0,0,1)^T$ in $\mathcal{R}^3$



- A set of projections $\{v_{i,t}\}_i$, $v_{i,t} \triangleq \bar{w}_i^T \bar{X}_t$ and $i = 1, 2, 3, \ldots$ can separate and emphasize different features in $\bar{X}_t$

- The probability of $\bar{X}_t$ belonging to a specific class $\mathcal{C}_j$ may depend on these features and different combinations of features may result in different values of the probability

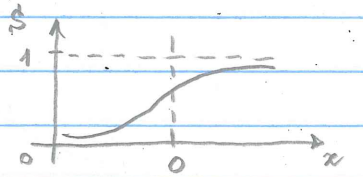
$\Downarrow$

A solution to the classification problem can be formalized in this way:

a) $z_{m,t} = S(\alpha_{0m} + \bar{\alpha}_m^T \bar{X}_t)$ $m = 1, 2, \ldots, M$

$\bar{\alpha}_m \triangleq [\alpha_{1m}\ \alpha_{2m} \ldots \alpha_{pm}] \in \mathcal{R}^p$

$S(x) \triangleq \dfrac{1}{1+e^{-x}}$ - sigmoidal function



b) $T_{kt} = \beta_{0k} + \bar{\beta}_k^T \bar{Z}_t \quad k = 1, 2, \ldots, J$

$\bar{Z}_t \triangleq [Z_{1t}\ Z_{2t} \ldots Z_{Mt}]^T$ from a)

$\bar{\beta}_k \triangleq [\beta_{1k}\ \beta_{2k} \cdots \beta_{Mk}]^T \in \mathcal{R}^M$

c) $f_k(\bar{X}_t) = g_k(\bar{T}_t) \quad k = 1, 2, \ldots, J$

$\bar{T}_t \triangleq [T_{1t}\ T_{2t} \ldots T_{Jt}]^T$ from b)

$g_k(\bar{T}_t) \triangleq \dfrac{e^{T_k}}{\sum_{\ell=1}^{J} e^{T_\ell}}$ - softmax function

Solution a)-c) is called "Artificial Neural Network" (ANN)

In the ANN a)-c), $f_k(\bar{X}_t) \in [0, 1]$ by construction and represents the probability of $\bar{X}_t$ belonging to $\mathcal{C}_k \quad k = 1, 2, \ldots, J$

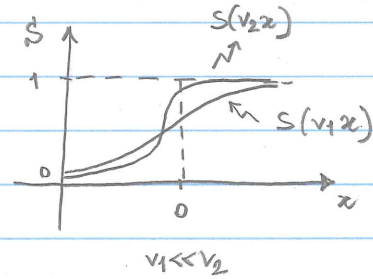
Note the ideas behind the ANN: Step a) ⇒ It is meant to separate and amplify different features

Step b) ⇒ We compute several weighted combinations of features (as many as the number of classes) and we assume that each class is linked to a preferred combination

Step c) ⇒ The probability of $\bar{X}_t$ belonging to one class depends nonlinearly on the combinations of features

These ideas are still satisfied if the sigmoidal and softmax functions are replaced ⇐

Note: $S(x)$ - sigmoidal ⇒ The size of the linearity region depends on $\|\bar{\alpha}_m\|$, i.e., the larger the norm, the smaller the region ⇒ The ANN can achieve better classification performance

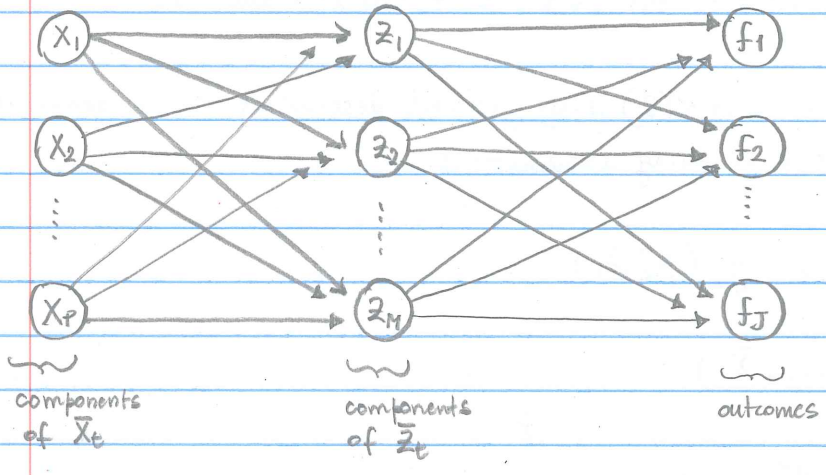


$v_1 \ll v_2$

Moreover, the ANN has a graphical interpretation that justifies, in part, the nomenclature:

$$z_{mt} = \sigma\left(\alpha_{0m} + \bar{\alpha}_m^T \bar{X}_t\right)$$

$$f_k = g_k\left(\beta_{01} + \bar{\beta}_1^T \bar{Z}_t, \dots, \beta_{0J} + \bar{\beta}_J^T \bar{Z}_t\right)$$



The solution is obtained by following a feed-forward network, in which one layer of variables (i.e., $\bar{Z}_t$) is not directly observed, i.e., it is HIDDEN

Note that the ANN a)-c) can be further generalized by adding more hidden layers between $\bar{X}_t$ and $\{f_k\}_{k=1}^J$ ⇒ We can consider an hierarchy of features (i.e., features computed out of combinations of other features), thus reaching increasing levels of abstraction

⇓

The solution a)-c), though, requires the estimation of the parameters:

$$(\alpha_{0m}\ \alpha_{1m} \dots \alpha_{pm}) \in \mathbb{R}^{p+1} \qquad m = 1, 2, \dots, M$$

$$(\beta_{0k}\ \beta_{1k} \dots \beta_{Mk}) \in \mathbb{R}^{M+1} \qquad k = 1, 2, \dots, J$$

Given a set of $N \geq 1$ observations $\{\bar{X}_t\}_{t=1}^N$ with correspondent (known) labels $\{\bar{Y}_t\}_{t=1}^N$, where $\bar{Y}_t \triangleq [Y_{1t}\ Y_{2t} \dots Y_{Jt}]^T$ and $Y_{kt} = 1$ if $\bar{X}_t \in \mathcal{C}_k$, $Y_{kt} = 0$ otherwise, we can find the solution:

$$\bar{\theta}^* = \arg\min_{\bar{\theta}} R(\bar{\theta})$$

$$R(\bar{\theta}) = -\sum_{t=1}^{N}\left(\sum_{k=1}^{J} Y_{kt} \log f_k(\bar{X}_t)\right)$$

$$\bar{\theta} \triangleq [\alpha_{01}\ \alpha_{11} \dots \alpha_{pM}\ \beta_{01}\ \beta_{11} \dots \beta_{MJ}]^T \quad \text{- } (p+1)\times M + (M+1)\times J \text{ parameters}$$

Note that: $R_t \triangleq -\sum_{k=1}^{J} y_{kt} \log f_k(\bar{X}_t)$ is a measure of the error obtained by assigning $\bar{X}_t$ based on the probabilities $f_k(\cdot)$ $k=1,2,\dots,J \Rightarrow R_t \geq 0$ $\forall t$ and is $R_t = 0$ when $\exists!\, k \in \{1,2,\dots,J\}$ s.t. $f_k(\bar{X}_t)=1$ and $y_{kt}=1$

An approach to minimizing $R(\bar{\theta})$ is by gradient descendent. To illustrate the method, let us make some simplifying assumptions:

$$g_k(\bar{T}_t) = g_k(T_k) = g_k(\bar{\beta}_k^T \bar{Z}_t)$$

$$Z_{m,t} = S(\eta_t) = S(\bar{\alpha}_m^T \bar{X}_t)$$

$$\Downarrow$$

$$\frac{\partial R_t}{\partial \beta_{mk}} = -\frac{y_{kt}}{f_k(\bar{X}_t)} \cdot \frac{dg_k}{dT_k}(\bar{\beta}_k^T \bar{Z}_t) \cdot z_{m,t}$$

$$\frac{\partial R_t}{\partial \alpha_{em}} = -\sum_{k=1}^{J} \frac{y_{kt}}{f_k(\bar{X}_t)} \cdot \frac{dg_k}{dT_k}(\bar{\beta}_k^T \bar{Z}_t)\beta_{mk} \cdot \frac{dS}{d\eta_t}(\bar{\alpha}_m^T \bar{X}_t) \cdot x_{et}$$

where: $\bar{\beta}_k \triangleq [\beta_{1k}\ \beta_{2k} \dots \beta_{Mk}]^T$ $\bar{\alpha}_m \triangleq [\alpha_{1m}\ \alpha_{2m} \dots \alpha_{pm}]^T$ $\bar{Z}_t \triangleq [z_{1t}\ z_{2t} \dots z_{M,t}]^T$

and $\bar{X}_t \triangleq [x_{1t}\ x_{2t} \dots x_{pt}]^T$ - Let us use the notation $\alpha_{em}^{(r)}$, $\beta_{mk}^{(r)}$ to denote the estimation of parameters α_{em}, β_{mk}, respectively, at the r-th iteration. The gradient descendent method is then given by the formula:

$$\beta_{mk}^{(r+1)} = \beta_{mk}^{(r)} - \gamma_r \sum_{t=1}^{N} \frac{\partial R_t}{\partial \beta_{mk}^{(r)}} \qquad r=1,2,\dots \qquad (1)$$

$$\alpha_{em}^{(r+1)} = \alpha_{em}^{(r)} - \gamma_r \sum_{t=1}^{N} \frac{\partial R_t}{\partial \alpha_{em}^{(r)}}$$

where initial conditions $\{\alpha_{em}^{(0)}\}_{e,m}$ $\{\beta_{mk}^{(0)}\}_{m,k}$ must be chosen and γ_r is the "learning rate" of the algorithm and must be set by the user

Note the following: $$\frac{\partial R_t}{\partial \beta_{mk}^{(r)}} = \underbrace{\left(\frac{\gamma_{kt}}{f_k^{(r)}(\bar{X}_t)} \cdot \frac{\partial g_k}{\partial \tau_k} \left(\bar{\beta}_k^{(r)T} \bar{Z}_t^{(r)} \right) \right)}_{\triangleq \delta_{k,t}^{(r)}} \cdot Z_{m,t}^{(r)} \qquad (*)$$

$$\frac{\partial R_t}{\partial \alpha_{em}^{(r)}} = \underbrace{\left(\sum_{k=1}^{J} \delta_{k,t}^{(r)} \beta_{mk}^{(r)} \frac{dS}{d\eta_t} \left(\bar{\alpha}_m^{(r)T} \bar{X}_t \right) \right)}_{\triangleq s_{mt}^{(r)}} \cdot X_{et}$$

where $Z_{m,t}^{(r)}$ is the value of the hidden variable $Z_{m,t}$ estimated by using the parameters at the r-th iteration. Because of the relationship:

$$s_{mt}^{(r)} = \frac{dS}{d\eta_t} \left(\bar{\alpha}_m^{(r)T} \bar{X}_t \right) \sum_{k=1}^{J} \delta_{k,t}^{(r)} \beta_{mk}^{(r)} \qquad (**)$$

updates (1) can be implemented with a two-pass algorithm:

S1) Estimate $f_k^{(r)}(\bar{X}_t)$ by using $\alpha_{em}^{(r)}$ and $\beta_{mk}^{(r)}$ $\forall m, k, e$:

$$f_k^{(r)}(\bar{X}_t) = g_k \left(\bar{\beta}_k^{(r)T} \bar{Z}_t^{(r)} \right) \qquad k = 1, 2, \ldots, J \text{ and } t = 1, 2, \ldots, N$$

$$\text{with } Z_{m,t}^{(r)} = S\left(\bar{\alpha}_m^{(r)T} \bar{X}_t \right) \qquad m = 1, 2, \ldots, M \text{ and } t = 1, 2, \ldots, N$$

S2) Estimate $\delta_{k,t}^{(r)}$ by using $(*)$ and then $s_{mt}^{(r)}$ by using $(**)$ $k = 1, 2, \ldots, J$, $m = 1, 2, \ldots, M$, and $t = 1, 2, \ldots, N$. $\Rightarrow$ Then the updates are given by:

$$\beta_{mk}^{(r+1)} = \beta_{mk}^{(r)} - \gamma_r \sum_{t=1}^{N} \delta_{k,t}^{(r)} Z_{m,t}^{(r)} \qquad (1')$$

$$\alpha_{em}^{(r+1)} = \alpha_{em}^{(r)} - \gamma_r \sum_{t=1}^{N} s_{mt}^{(r)} X_{et}$$

Note: Pass S1 fixes the parameters and moves forward along the network to estimate the probabilities $f_k^{(r)}(\cdot) \Rightarrow$ It is the FORWARD PASS

Pass S2 uses the current probabilities to compute $\delta_{k,t}^{(r)}$ first (which is related

to the first layer of computation from the right side of the graph) and then $s_{mt}^{(r)}$ (which is related to the last layer from the right) $\Rightarrow$ It is the BACKWARD PASS

Also note that, in (1'), we first update β_{mk} and then α_{em}, i.e., we move backward from the outer layer of the network toward the inner one $\Rightarrow$ The's implementation of the gradient descendent method is called BACK-PROPAGATION LEARNING ALGORITHM

A few considerations:

- Initial conditions must be chosen $\Rightarrow$ Typically, the values $\alpha_{em}^{(0)}$ and $\beta_{mk}^{(0)}$ are set randomly and close to 0 $\Rightarrow$ The ANN operates in linearity region at first, and then evolves toward a nonlinear behavior as the value of the parameters increases

- Overfitting is often obtained with the back-propagation algorithm $\Rightarrow$ Early stopping rules can be considered. Alternatively, one can minimize a modified error function:

$$R'(\bar{\theta}) \triangleq R(\bar{\theta}) + \lambda J(\bar{\theta}) \quad \text{with } \lambda \geq 0 \text{ (to be chosen)}$$

$$J(\bar{\theta}) \triangleq \sum_{k,m} \beta_{mk}^2 + \sum_{e,m} \alpha_{em}^2$$

$\Rightarrow$ The weights α_{em} and β_{mk} are let decay by penalizing their square value

The value of λ as well as eventual early stopping rules are typically chosen via cross-validation methods

- The error function $R(\bar{\theta})$ is typically nonconvex $\Rightarrow$ Because several minima are possible, a number of initial conditions should be tried out, then the ANNs obtained from these initial conditions should be used and the classification predictions should be averaged across these ANNS

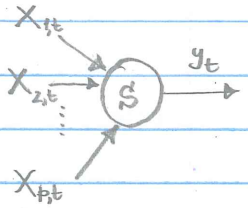
- Consider the more simplistic case: $M=1 \Rightarrow Z_t = S(\bar{\alpha}^T \bar{X}_t)$

$$J=2 \Rightarrow f_2(\bar{X}_t) = 1 - f_1(\bar{X}_t)$$

$$f_1(\bar{X}_t) = Z_t = \frac{1}{1 + e^{-\bar{\alpha}^T \bar{X}_t}}$$

From a qualitative standpoint, the single processing unit:

$$y_t = \frac{1}{1 + e^{-\bar{\alpha}^T \bar{X}_t}}$$



behaves as a neuron that processes pre-synaptic stimuli and eventually responds with an action potential
⇒ Hence, the ANN terminology

Moreover, in this case, $R_t = -Y_{1,t} \log y_t - (1 - Y_{1,t}) \log (1 - y_t)$ ⇒ The derivative with respect to one of the parameters α_i in $\bar{\alpha} \triangleq [\alpha_1 \; \alpha_2 \; \dots \; \alpha_p]^T$ becomes:

$$\frac{\partial R_t}{\partial \alpha_i} = -(Y_{1,t} - y_t) X_{i,t}$$ ⇒ The back propagation algorithm updates the parameter estimation based on the prediction error

* Capacity of a Single Neuron in an ANN

Let us consider the single neuron from above and let us implement the rule:

$$y_t = f(\bar{\alpha}^T \bar{X}_t)$$

$$f(a) = \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{if } a \leq 0 \end{cases}$$

Let us also assume: $\{X_t\}_{t=1}^N$ - given
$Y_t \in \{0,1\}$ - known
⇒ We want to choose $\bar{\alpha}$ such that the max number of true labels is matched

A theoretical analysis can be done to determine the "capacity" of the network, i.e, the amount of information that can be stored by training an ANN:

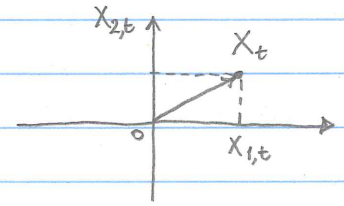
- $p=1$ ⇒ By choosing $\alpha > 0$ all $X_t > 0$ result in $y_t = 1$
By choosing $\alpha < 0$ all $X_t > 0$ result in $y_t = 0$
⇒ By choosing α, we can define 2 classification rules

Let us define: $T(N,p) \triangleq$ number of possible classification rules that can be enforced by training the ANN on N observations $\{X_t\}_{t=1}^{N}$, with $X_t \in \mathbb{R}^p$

$t = 1, 2, \ldots, N$

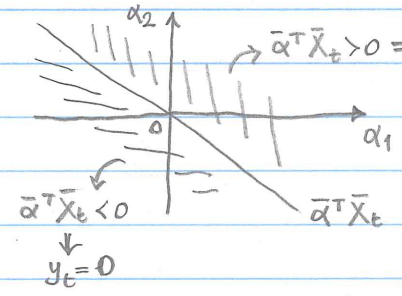
Hence, $p=1 \Rightarrow T(N,1) = 2 \quad \forall N > 0$

- $p=2 \Rightarrow$ If $N=1$, then the condition $\bar{\alpha}^T \bar{X}_t = 0$ identifies a threshold in the space of parameters $\bar{\alpha} = [\alpha_1 \; \alpha_2]^T$:



$$\bar{\alpha}^T \bar{X}_t = 0 \Leftrightarrow \alpha_1 X_{1,t} + \alpha_2 X_{2,t} = 0$$

$$\Leftrightarrow \alpha_2 = -X_{1,t} / X_{2,t} \, \alpha_1$$

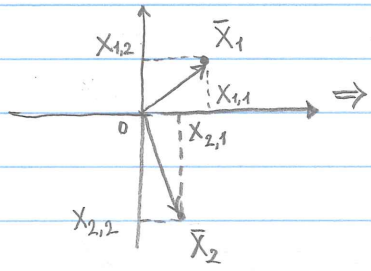


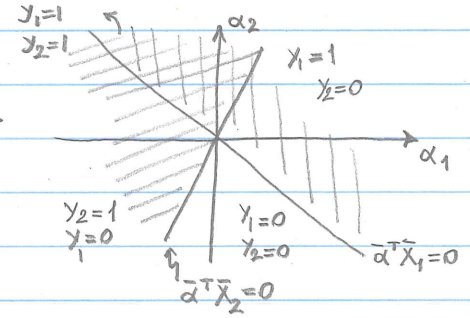
Hence, by choosing $\bar{\alpha}$, we can determine two classification rules, i.e., $T(1,2) = 2$

- The approach can be generalized to $N > 1$ observations, i.e., each condition $\bar{\alpha}^T \bar{X}_t = 0$ $t = 1, 2, \ldots, N$ defines a threshold in the space of parameters $\bar{\alpha}$ and the sequence of classification decisions depends on the intersection of these thresholds. Ex.:

$t=1 \Rightarrow \alpha_2 = -X_{1,1} / X_{2,1} \, \alpha_1$

$t=2 \Rightarrow \alpha_2 = -X_{1,2} / X_{2,2} \, \alpha_1$



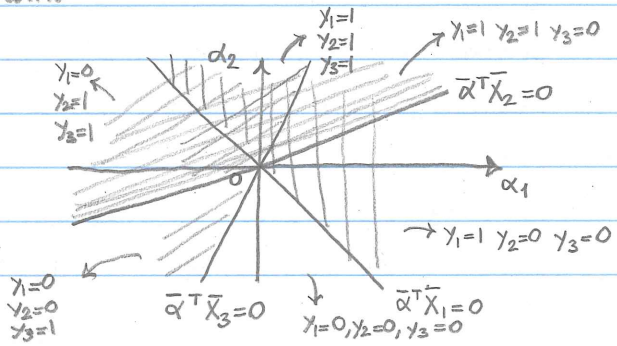


In this case, we have: $T(2,2) = 4$, i.e., the ANN can be trained to correctly classify 1 out of 4 possible pairs of observations $\Rightarrow$ The ANN can have "memory" of that specific pair of observations it was trained with

Ex.: $t=1 \Rightarrow \alpha_2 = -X_{1,1} / X_{2,1} \, \alpha_1$

$t=2 \Rightarrow \alpha_2 = -X_{1,2} / X_{2,2} \, \alpha_1$

$t=3 \Rightarrow \alpha_2 = -X_{1,3} / X_{2,3} \, \alpha_1$



In this case, $T(3,2) = 6 < 2^3 = 8$, i.e., there are two sequences (i.e., $(0,1,0)$ and $(1,0,1)$) that will never be correctly matched regardless of how we vary $\bar{\alpha} \Rightarrow$ The "capacity" of the ANN is only 6

In general, it can be proved that: $T(N,2) = 2N \quad \forall N > 0$
The same approach can be followed when $p > 2 \Rightarrow$ It can be proved that the number $T(N,p)$, which can be used as a measure of the capacity (i.e., memory) of the ANN, grows according to the recursive formula:

$$T(N,p) = T(N-1,p) + T(N-1,p-1)$$

or, equivalently:

$$T(N,p) = 2 \sum_{k=0}^{p-1} \binom{N-1}{k} \qquad (2)$$

Note that, in (2), $T(N,p) = 2^N$ if $p \geq N \Rightarrow$ We can measure the probability that a generic sequence of labels $\{\gamma_t\}_{t=1}^N$ is actually memorized by the ANN as $T(N,p)/2^N$
$\Rightarrow$ The condition $N = p$ defines the max number of observations on which any arbitrary labelling can be realized $\Rightarrow$ p is called "Vapnik-Chervonenkis dimension"

From this, one may conclude:
$N \leq p \Rightarrow T(N,p)/2^N = 1$
$p < N \leq 2p \Rightarrow 1/2 \ll T(N,p)/2^N < 1$
$N > 2p \Rightarrow T(N,p)/2^N \simeq 0$
$\Rightarrow$ N labels can be "almost certainly" memorized for $N \leq 2p$

* Statistical Aspects of a Single Neuron in an ANN

The error function: $R = -\sum_{t=1}^{N} \left[\gamma_{1,t} \log y_t + (1-\gamma_{1,t}) \log (1-y_t) \right]$ can be envisioned as the log-likelihood function (multiplied by -1) of a sequence of Bernoulli trials:

$$\left.\begin{aligned} P(\gamma_{1,t} = 1 \mid \bar{\alpha}, \bar{X}_t) &= f(\bar{\alpha}^T \bar{X}_t) = y_t \\ P(\gamma_{1,t} = 0 \mid \bar{\alpha}, \bar{X}_t) &= 1 - y_t \end{aligned}\right\} \Leftrightarrow P(\gamma_{1,t} \mid \bar{\alpha}, \bar{X}_t) = y_t^{\gamma_{1,t}} \cdot (1-y_t)^{1-\gamma_{1,t}}$$

Therefore, $\exp(-R) = P(Y_{1,1}, Y_{1,2}, \ldots, Y_{1,N} \mid \bar{\alpha}, \bar{X}_1, \bar{X}_2, \ldots, \bar{X}_N)$

In general, we are interested in assess the estimated parameters $\bar{\alpha}$, i.e., we want to calculate: $P(\bar{\alpha} \mid \underbrace{\{Y_{1,t}\}_{t=1}^{N}, \{\bar{X}_t\}_{t=1}^{N}}_{\mathcal{H}_{N+1}}) \Rightarrow$ By using the Bayes' rule:

$$P(\bar{\alpha} \mid \mathcal{H}_{N+1}) = \frac{P(\mathcal{H}_{N+1} \mid \bar{\alpha})\, P(\bar{\alpha})}{P(\mathcal{H}_{N+1})}$$

where $P(\bar{\alpha})$ is the prior probability function of $\bar{\alpha}$. The choice of $P(\bar{\alpha})$ is, at this point, arbitrary. However, a typical choice is:

$$P(\bar{\alpha}) = \frac{1}{Z_\alpha(\lambda)} \exp\left(-\lambda E_\alpha\right)$$

where λ is a parameter to be chosen and $Z_\alpha(\lambda) \triangleq \int \exp\left(-\lambda E_\alpha\right) d\bar{\alpha}$, with E_α being a function of $\bar{\alpha}$ to be chosen $\Rightarrow$ Typical choice is:

$E_\alpha \triangleq \frac{1}{2} \sum_{k=1}^{p} \alpha_k^2 \Rightarrow P(\bar{\alpha}) = P(\bar{\alpha} \mid \lambda)$ is Gaussian with S.D. $\sigma = \frac{1}{\lambda}$ and

$$Z_\alpha(\lambda) = \sqrt{\left(\frac{2\pi}{\lambda}\right)^p}$$

Note that: $\lambda E_\alpha = \lambda \frac{1}{2} \sum_{k=1}^{p} \alpha_k^2 \Rightarrow \lambda E_\alpha \propto \lambda J$ - quadratic term added to R in $R' = R + \lambda J$ to implement the weight decay

Hence, we can derive:

$$P(\mathcal{H}_{N+1} \mid \bar{\alpha}) = P\left(\{Y_{1,t}\}_{t=1}^{N} \mid \bar{\alpha}, \{\bar{X}_t\}_{t=1}^{N}\right) \underbrace{P\left(\{\bar{X}_t\}_{t=1}^{N} \mid \bar{\alpha}\right)}_{=1 \text{ since the observations are given}} = e^{-R}$$

$$P(\bar{\alpha} \mid \lambda) = \frac{1}{Z_\alpha(\lambda)} e^{-\lambda E_\alpha}$$

$$P(\bar{\alpha}\,|\,\mathcal{H}_{N+1}) = \frac{e^{-R}\cdot e^{-\lambda E_\alpha}}{Z_\alpha(\lambda)\,P(\mathcal{H}_{N+1})}$$

Note: $P(\mathcal{H}_{N+1}) = \int e^{-R}\, e^{-\lambda E_\alpha}\, \frac{d\bar{\alpha}}{Z_\alpha(\lambda)}$ - Therefore, by calling:

$$\left.\begin{array}{l} G \triangleq R + \lambda E_\alpha \\ Z_G \triangleq \int e^{-G}\, d\bar{\alpha} \end{array}\right\} \Rightarrow \text{We can write: } P(\bar{\alpha}\,|\,\mathcal{H}_{N+1}) = \frac{e^{-G}}{Z_G} \qquad (***)$$

The calculation of the probability function (***) can be used to assess the outcomes of the classification on new data $\Rightarrow$ Let us assume that a new observation $\bar{X}_{N+1}$ is given and weights $\bar{\alpha} = [\alpha_1\ \alpha_2 \dots \alpha_p]^T$ are used to provide the classification outcome y_{N+1} but $\bar{\alpha}$ were NOT trained on $\bar{X}_{N+1}$:

$$P(Y_{1,N+1}\,|\,\bar{X}_{N+1}, \mathcal{H}_{N+1}) = \int P(Y_{1,N+1}\,|\,\bar{\alpha}, \bar{X}_{N+1}, \mathcal{H}_{N+1})\, P(\bar{\alpha}\,|\,\mathcal{H}_{N+1})\, d\bar{\alpha}$$

$$\left.\begin{array}{l} \text{where: } P(Y_{1,N+1}\,|\,\bar{\alpha}, \bar{X}_{N+1}, \mathcal{H}_{N+1}) = y_{N+1}^{Y_{1,N+1}} \cdot (1-y_{N+1})^{1-Y_{1,N+1}} \\ \qquad \uparrow \text{It does not depend on } \mathcal{H}_{N+1} \\ P(\bar{\alpha}\,|\,\mathcal{H}_{N+1}) = e^{-G}/Z_G \end{array}\right\} \Rightarrow \text{Given that } G = G(\bar{\alpha}), \text{ we have:}$$

$$P(Y_{1,N+1}=1\,|\,\bar{X}_{N+1}, \mathcal{H}_{N+1}) = \int y_{N+1}(\bar{\alpha})\, \frac{e^{-G(\bar{\alpha})}}{Z_G(\bar{\alpha})}\, d\bar{\alpha} \Rightarrow$$

The probability of classifying $\bar{X}_{N+1}$ as belonging to class 1 is the average of the output y_{N+1} of the neuron under the posterior distribution of $\bar{\alpha}$

The integral in the formula above can be computed approximately as follows:

$\bar{\alpha}^*$ is the arg of the (local) minimum of $G(\bar{\alpha}) = R(\bar{\alpha}) + \lambda E_\alpha(\bar{\alpha})$ $\Rightarrow$ (Taylor's expansion) $G(\bar{\alpha}) \cong G(\bar{\alpha}^*) + \frac{1}{2}(\bar{\alpha}-\bar{\alpha}^*)^T A\,(\bar{\alpha}-\bar{\alpha}^*)$ (We are neglecting higher order terms)

where A is the Hessian of $G(\bar{\alpha})$, i.e., it's a $p \times p$ matrix whose (i,j)-th

element is: $A_{ij} = \left.\frac{\partial^2 G}{\partial\alpha_i\,\partial\alpha_j}\right|_{\bar{\alpha}=\bar{\alpha}^*}$ $i,j = 1,2,\ldots,p$ $\Rightarrow$ Hence the posterior probability can be (locally) approximated as:

$$\frac{e^{-G(\Delta\bar{\alpha})}}{Z_G(\Delta\bar{\alpha})} \cong \sqrt{\frac{\det(A)}{(2\pi)^p}}\exp\left(-\frac{1}{2}\Delta\bar{\alpha}^T A\,\Delta\bar{\alpha}\right) \quad \text{where } \Delta\bar{\alpha} \triangleq \bar{\alpha} - \bar{\alpha}^*$$

$$\Downarrow$$

$$P(y_{i,N+1}=1 \mid \bar{X}_{N+1}, \mathcal{H}_{N+1}) \cong \sqrt{\frac{\det(A)}{(2\pi)^p}} \int y_{N+1}(\bar{\alpha})\exp\left(-\frac{1}{2}\Delta\bar{\alpha}^T A\,\Delta\bar{\alpha}\right) d\bar{\alpha}$$

Denoted with $a_{N+1} \triangleq \bar{\alpha}^T X_{N+1}$, if $\bar{\alpha}$ is a Gaussian RV (which happens because $\Delta\bar{\alpha}$ is locally Gaussian), then a_{N+1} is Gaussian $\Rightarrow$ We can write:

$$p_a(a_{N+1} \mid \bar{X}_{N+1}, \bar{\alpha}, \mathcal{H}_{N+1}) = \frac{1}{\sqrt{2\pi}\,\sigma_\alpha}\exp\left(-\frac{(a_{N+1}-a_{N+1}^*)^2}{2\sigma_\alpha^2}\right)$$

where: $p_a(\cdot)$ is the conditional pdf of a_{N+1}, $\sigma_\alpha^2 \triangleq \bar{X}_{N+1}^T A^{-1} \bar{X}_{N+1}$, and $a_{N+1}^* \triangleq \bar{\alpha}^* \bar{X}_{N+1}$

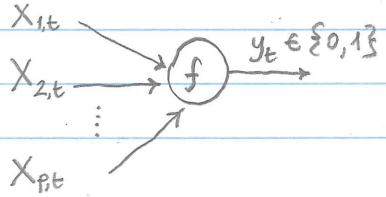
Therefore, given that $y_{N+1} = f(a_{N+1})$, we can approximate the integral with the formula:

$$P(y_{i,N+1}=1 \mid \bar{X}_{N+1}, \mathcal{H}_{N+1}) \cong \int f(\bar{\alpha}^T \bar{X}_{N+1}) \frac{1}{\sqrt{2\pi}\,\sigma_\alpha}\exp\left(-\frac{(\Delta\bar{\alpha}^T \bar{X}_{N+1})^2}{2\sigma_\alpha^2}\right) d\bar{\alpha}$$

□

* Hopfield Artificial Neural Networks

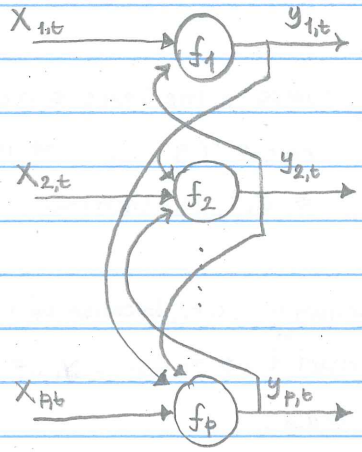
In the networks considered thus far, each neuron combines all the input variables and projects the outcomes of the processing to the next layer. For instance:



$$y_t = f(\bar{\alpha}^T \bar{X}_t) \qquad f(a) = \begin{cases} 1 & a > 0 \\ 0 & a \le 0 \end{cases}$$

or, alternatively: $f(a) = \frac{1}{1+e^{-a}}$ - sigmoidal function

Let us now consider the following example:



Each neuron f_i, $i = 1, 2, \dots, P$, receives one observation ($X_{i,t}$) and the output of every other neuron f_j $j \neq i$ in the same layer

⇓

P1) We have a feedback ANN

P2) The network is fully connected

P3) There is no auto-feedback

Let us also make these two assumptions:

P4) $f_i(a) = f(a) \triangleq \begin{cases} 1 & a \geq 0 \\ -1 & a < 0 \end{cases}$

P5) The network has symmetric weights, i.e., $\forall i \neq j$ we have:

$$\left.\begin{aligned} y_{i,t} &= f(a_i) \quad a_i \triangleq \alpha_{ii} X_{i,t} + \sum_{\ell \neq i} \alpha_{i\ell}\, y_{\ell,t} \\ y_{j,t} &= f(a_j) \quad a_j \triangleq \alpha_{jj} X_{j,t} + \sum_{\ell \neq j} \alpha_{j\ell}\, y_{\ell,t} \end{aligned}\right\} \text{ with } \alpha_{ij} = \alpha_{ji}$$

An ANN that satisfies P1-P5 is called a "Hopfield Network". ⇒ Note that, from a practical standpoint, the execution of the ANN may follow one of two paradigms:

- SYNCHRONOUS UPDATE: All summations a_i are computed at the same time by using old values of $y_{j,t}$ $j \neq i$

- ASYNCHRONOUS UPDATE: Summations a_i are computed sequentially (order to be decided) with the new value $y_{i,t}$ used in any following summation a_j with $j \neq i$

In this network, the "state" is given by the vector $\bar{y}_t \triangleq [y_{1,t}\ y_{2,t} \dots y_{P,t}]^T \in \{1, -1\}^{P \times 1}$,

i.e., $\bar{y}_t$ is a binary pattern, and the weights α_{ij} are typically estimated on a training set of N input vectors $\bar{X}_t$:

Hebb Rule

$$\alpha_{ii} = \eta \sum_{t=1}^{N} X_{i,t}^2 \qquad \alpha_{ij} = \eta \sum_{t=1}^{N} y_{i,t}\, y_{j,t} \qquad \text{with } \eta \text{ - scaling factor to be chosen}$$

Note: high corr $(X_{i,*}, X_{j,*}) \Rightarrow y_{i,*}$ and $y_{j,*}$ evolve in the same direction and exert mutual reinforcement (e.g., $y_{i,t}=1 \Rightarrow$ it contributes to increase $y_{j,t}$) $\Rightarrow \alpha_{ij}, a_{ji}$ increase

The weights operate as an "associative memory", i.e., because of this training, next time that $X_{i,t}$ (or $X_{j,t}$) alone presents a tract that triggers $y_{i,t}=1$ (or $y_{j,t}=1$), the probability of having $y_{j,t}=1$ (or $y_{i,t}=1$) increases

Note: The implementation of "associative memory" implies that, for a given $\bar{X}_t$, the response $\bar{y}_t$ will vary with the number of updates that are allowed for $y_{i,t}$ $i=1,2,\ldots,p$ and α_{ij} before moving to the next observations $\bar{X}_{t+1}$

⇓

It can be proved that, asymptotically, any pattern $\bar{Y}_t = [\,Y_{1t}\ Y_{2t} \cdots Y_{pt}\,]^T \in \{-1,1\}^{p\times 1}$ can be reached by $\bar{y}_t$, provided that the network is updated asynchronously □

References:

Mackay, D.J.C. (2004) "Information Theory, Inference, and Learning Algorithms," Cambridge University Press, Cambridge, UK

↑

I referred to ch. 38, 39, 40, 41, 42 (up to section 42.5) - A pre-print copy of the book is on Husky CT (NOT TO BE PRINTED!) - A copy of the book is in the library