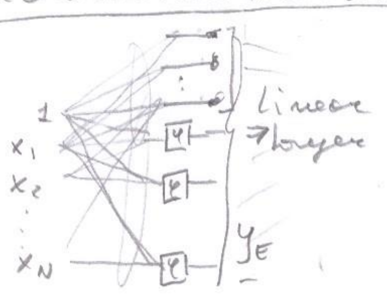
OTHER TWO-LAYER ARCHITECTURES

The examples seen in the previous section are only some of the possible architectures that can be built. The representation theorem suggests that there exist many other options to build the first layer. We will try to discuss a few of them in the following sections.

SKIP-CONNECTION ARCHITECTURE



In this architecture the embedding space is augmented with skip-connections, i.e. with a strictly linear part. The activation functions can be sigmoids, RELUs, or any other function.

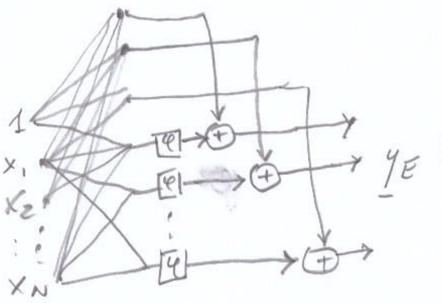
$$\underline{y}_E = \begin{bmatrix} \underline{C}_1^T \underline{x} + b_1 \\ \psi(C_2^T \underline{x} + b_2) \end{bmatrix} = \begin{bmatrix} \underline{y}_{EL} \\ \underline{y}_{E\psi} \end{bmatrix}$$

The idea here is to provide also a strictly linear part into the embedding space. Training of this architecture will ensure that classifiers and regressors are at least linear.

The nonlinear part would be used by the output layer if necessary. Note that the linear and the nonlinear parts can have different sizes.

RESIDUAL ARCHITECTURE

The residual architecture also builds on the idea of generalizing the linear layer. In this case the output of the linear part is <u>summed</u> to the output of the nonlinear part.



$$\underline{y}_E = \left(\underline{\underline{C}}_1 \underline{x} + \underline{b}_1\right) + \varphi\left(\underline{\underline{C}}_2^T \underline{x} + \underline{b}_2\right)$$

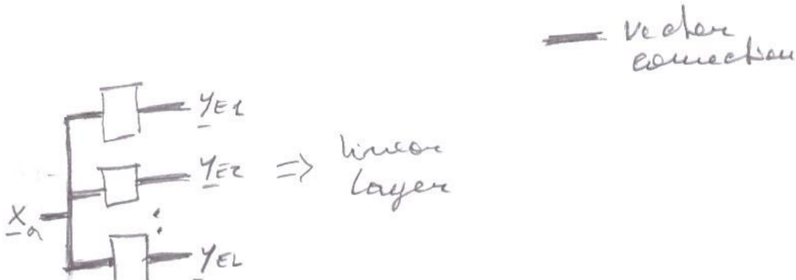
Clearly here the sizes of the linear and nonlinear parts have to be the same (N_E). The terminology "residual" comes from the fact that $\underline{y}_E$ may be considered the difference between a linear and a nonlinear mappings (differences or sums can be learned). Clearly this architecture is to be considered a bit less powerful than the skip connection parallel structure. Usually residual layers are used mostly on multi-layer architectures. A word of caution should be spent about using structures with more less constrained parameters: learning has to determine these

TW0.26

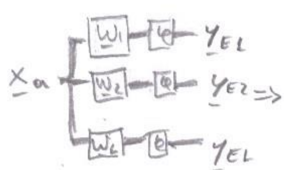
parameters and generalization should be checked.
Therefore any choice of neural network structure should be done with careful attention to overfitting.
The real challenge is therefore in finding the proper parametrization for the problem to be solved

PARALLEL ARCHITECTURES

The SKIP-CONNECTION architecture is a special case of parallel structures



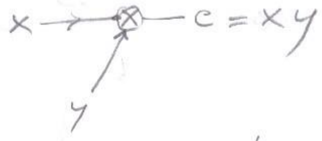
Each element of the parallel may be linear or nonlinear.
The embedding space $\underline{y}_E = \begin{bmatrix} y_{E1} \\ y_{E2} \\ \vdots \\ y_{EL} \end{bmatrix}$ is composed by the concatenation of the various vector-outputs that may specialize on different features.
When all the parallel structures are of the same type, for example linear + activation functions, the network is equivalent to a single network with constrained connectivity.



$\Rightarrow$ We will encounter architectures with constrained connectivity when we will talk about convolutional networks.

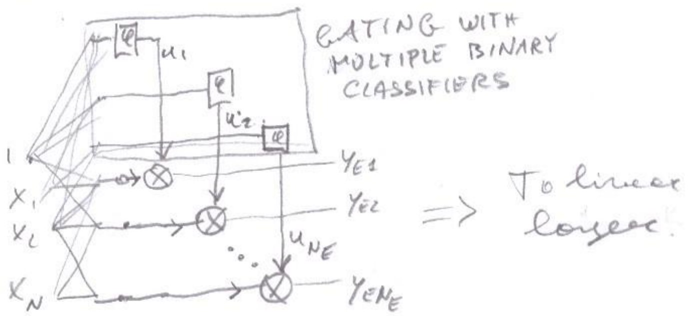
SUM-PRODUCT ARCHITECTURES

Until now we have considered only neural network architectures that use linear combinations and scalar nonlinearities. Larger functional capabilities can be obtained if we include also product units, i.e. nodes where the inputs are multiplied.



It may be argued that multiplication could be implemented summing the logs and then exponentiating. However there are some architectures where the product nodes may be very useful and act as "gates".
It should be pointed out that these structures are not the sum-product architectures used to model random variable distributions. They simply indicate the inclusion of product nodes.

GATED ARCHITECTURES



The embedding vector is obtained as

$$\underline{y}_E = \varphi(\underline{\underline{G}}^T \underline{x} + \underline{d}) \odot (\underline{\underline{C}}^T \underline{x} + \underline{b})$$

$$\underline{G} = [g_1 g_2 \cdots g_{N_E}]$$

$$\underline{C} = [c_1 c_2 , \ldots , c_{N_E}]$$

Here each binary classifier "activates" a linear output. If a classifier works in saturation $\simeq 1$ or $\simeq 0$, it turns on and off the connected linear filter at the bottom. It essentially builds one filter bank activated according to the gating mechanism. If instead the specific classifier works in its quasi-linear range, it implements a quadratic transformation

More specifically if the jth classifier gives (at a specific $\underline{x}$) $u_j \simeq 1$, $y_{E_j} \cong \underline{e}_j^T \underline{x} + b_j$, otherwise if $u_i \simeq 0$ $y_{E_j} \cong 0$.

Viceversa if the jth classifier works in its linear range, we have

$$u_j \simeq \underline{g}_j^T \underline{x} + d_j$$

and

$$y_{E_j} \cong (\underline{g}_j^T \underline{x} + d_j)(\underline{e}_j^T \underline{x} + b_j)$$

$$= \underline{g}_j^T \underline{x}\,\underline{x}^T \underline{e}_j + d_j \underline{e}_j^T \underline{x} + b_j \underline{g}_j^T \underline{x} + d_j b_j$$

which is a quadratic function of $\underline{x}$.

Once we have set N_E, the learning algorithm will provide the parameters $\underline{\underline{G}}, \underline{d}, \underline{\underline{E}}, \underline{b}$ that will characterize the embedding space needed for the final regressor or classifier.(*)

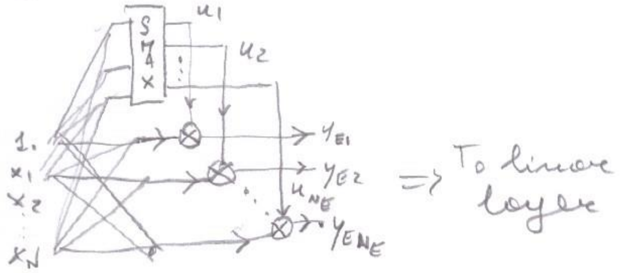
This is a very powerful architecture for its capabilities of providing linear and nonlinear embeddings.

This gating unit is utilized in the LSTM (LONG-SHORT TIME MEMORY) architectures which is a very powerful recurrent network. The LSTM will be discussed in one of the following chapters.

(*) Recall the gaussian classifier in the model-based chapter of this book that requires quadratic log-likelihoods. Therefore this architecture may be suitable to build a classifier with clusters that have different covariance matrices....(?)

Wh

Another gated architecture is the following one TWO.30



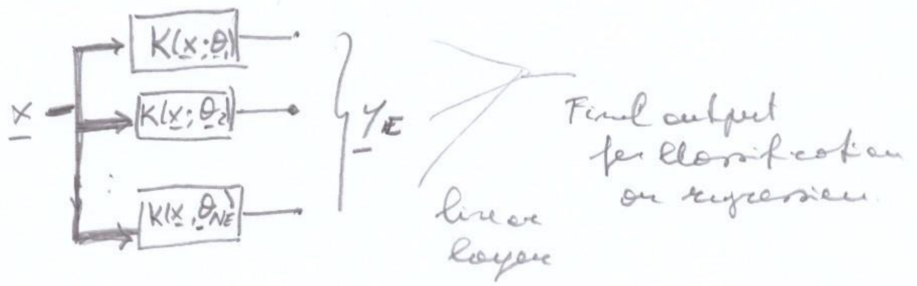
$$\underline{y}_E = S_{max}\left(\underline{\underline{C}}^T \underline{x} + \underline{d}\right) \odot \left(\underline{C}^T \underline{x} + \underline{b}\right)$$

Here the gating is done by a N_E-class linear classifier.
The difference with the previous architecture is that here the target is one class out of N_E (Recall that $\sum_{i=1}^{N_E} u_i = 1$)
Therefore if $\underline{x}$ is such that $u_j \simeq 1$ and the others are $\simeq 0$, $y_{Ej} \simeq c_j^T \underline{x} + b_j$. Essentially in every decision region of the linear classifier, one output of $\underline{y}_E$ is active and is a linear function of $\underline{x}$.

Note that both in this and the previous gated architectures, N_E can be made very large giving to the system the opportunity of building many useful basis functions $\{y_{Ei}(\underline{x}),\ i = 1 \ldots, N_E\}$.

KERNEL-BASED ARCHITECTURES

The representation theorem focuses on finding the proper embedding space with basis functions that can be linearly combined to produce the system output.
One of the early ideas about building such space, is to postulate a number of parametric "Kernels". The following figure shows such an architecture



Each embedding function is a Kernel function

$$Y_{Ei} = K(\underline{x};\underline{\theta}_i)$$

that depends on a number of parameters $\underline{\theta}_i$.
A scalar output (scalar regression) would be the superposition

$$y(\underline{x}) = \sum_{i=1}^{N_E} e_i \, K(\underline{x},\underline{\theta}_i) + b$$

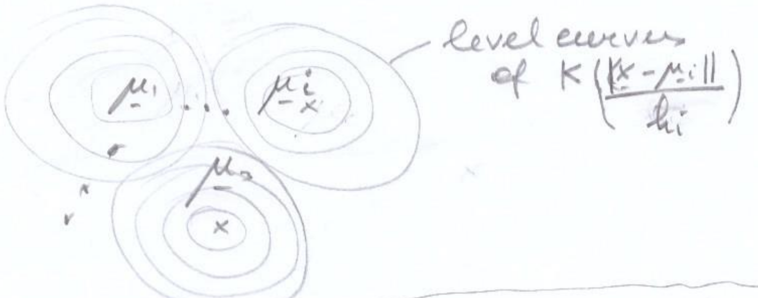
or similarly multiple outputs for classification or multiple regression.

RADIAL-BASIS FUNCTIONS

The choice of the Kernel functions $K(\underline{x},\underline{\theta}_i)$ is done essentially to "cover" the input space with functions that are usually assumed to have a radial symmetry

$$K(\underline{x};\underline{\theta}_i) = K\left(\frac{\|\underline{x}-\underline{\mu}_i\|}{h_i}\right)$$

where $\|\underline{x}-\underline{\mu}_i\|$ is the square of the euclidean distance between $\underline{x}$ and a "cluster point" $\underline{\mu}_i$, and h_i controls the spread.



NOTA:

The idea of using radial functions has come from the traditional problem of density estimation in statistics. (Parzen density estimator) More specifically if we are given a set of samples $\underline{x}_1, \underline{x}_2, \ldots \underline{x}_L$ we could build density function of $\underline{x}$ by placing a function around each point $\underline{x}_i$ and superimposing

$$\hat{p}(\underline{x}) = \sum_{i=1}^{L} \alpha_i K\left(\frac{\|\underline{x}-\underline{x}_i\|}{h_i}\right)$$

To be a valid pdf we have to impose

$$\int_{\mathcal{X}} \hat{p}(\underline{x})\, d\underline{x} = 1 \Rightarrow \sum_{i=1}^{L} \alpha_i \int_{\mathcal{X}} K\left(\frac{\|\underline{x}-\underline{x}_i\|}{h_i}\right) d\underline{x} = 1$$

Kernel functions can be normalized and the coefficients α_i regulate the height of each function.

TW0.33

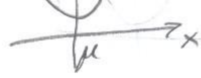
The most common choice for $K(\cdot)$ is the Gaussian function

$$K\left(\frac{\|\underline{x}-\underline{\mu}\|^2}{h^2}\right) = e^{-\frac{\|x-\mu\|^2}{2h^2}}$$

even if other choices are possible, such as the multiquadratic

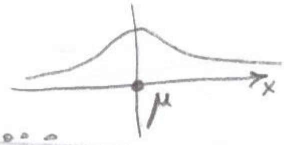
$$K\left(\|\underline{x}-\underline{\mu}\|^2, \beta\right) = \sqrt{\|\underline{x}-\underline{\mu}\|^2+\beta^2}$$

μ is the center and β controls the spread



or the inverse multiquadratic

$$K\left(\|x-\mu\|^2, \beta\right) = \frac{1}{\sqrt{\|x-\mu\|^2+\beta^2}}$$



There are other radial functions...

In any case more generally the $\underline{\mu}_i$ are cluster points and h_i controls the size of each the spherical region around $\underline{\mu}_i$.

The superposition of these spherical functions can build the necessary likelihoods or log-likelihoods for regression or classification.

Usually the determination of the cluster points μ_i and the size h_i is the result of an "unsupervised" phase. We have not talked about unsupervised learning yet as the topic will be more specifically addressed in one of the following chapters.

Essentially we can assume that the clusters will be determined only from $\{\underline{x}[n], n=1,\dots n_2\}$ in the training set, with an algorithm such as K-means or similar. (we will address this issue in the following)

The supervised learning phase determines then the best linear combination for classification or regression.

Usually the size of the embedding space N_E is kept much smaller than the number of examples and the size of the clusters h_i is chosen to make the overall function sufficiently smooth. Sometimes $h = \frac{d_{max}}{\sqrt{2N_E}}$ ← max distance between clusters

↑ # of clusters.

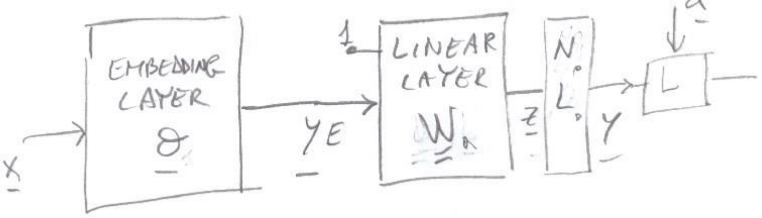
... more ...

... example ...

Note that the clusters can be built to have different ellipsoidal shape (covariances); obviously there would be many more parameters to be learned.

LEARNING IN TWO-LAYER ARCHITECTURES

The general two-layer architecture of Figure (*), repeated here for convenience, contains free parameters to be learned in both layers



The N.L. block is not present in regression and It is made up of a set of sigmoids in the multiple binary classifier and a softmax in the multi-class classifier.

The cost function to be minimized is

$$\mathcal{E}(\underline{\theta}, \underline{\underline{W}}_a) = \frac{1}{n_z} \sum_{n=1}^{n_z} L(\underline{y}[n], \underline{d}[n])$$

As explained in one of the previous chapter we need to compute the gradients

$$\nabla_{\underline{\underline{W}}_a} \mathcal{E}(\underline{\theta}, \underline{\underline{W}}_a) \text{ and } \nabla_{\underline{\theta}} \mathcal{E}(\underline{\theta}, \underline{\underline{W}}_a)$$

for making the updates

$$\underline{\underline{W}}_a[k] = \underline{\underline{W}}_a[k-1] - \mu \nabla_{\underline{\underline{W}}_a} \mathcal{E}(\underline{\theta}[k-1], \underline{\underline{W}}_a[k-1])$$

$$\underline{\theta}[k] = \underline{\theta}[k-1] - \mu \nabla_{\underline{\theta}} \mathcal{E}(\underline{\theta}[k-1], \underline{\underline{W}}_a[k-1])$$

To simplify notation let us drop the example index n and concentrate on forward propagation and gradients on a single example

THE OUTPUT LAYER

Recall the structure of the linear layer

$$\underline{z} = \underline{\underline{W}}_a^T \underline{y}_E^a = \begin{bmatrix} b_1 & \underline{w}_1^T \\ b_2 & \underline{w}_2^T \\ \vdots & \\ b_M & \underline{w}_M^T \end{bmatrix} \begin{bmatrix} 1 \\ \underline{y}_E \end{bmatrix} = \begin{bmatrix} \underline{b} & \begin{matrix} \underline{w}_1^T \\ \underline{w}_2^T \\ \underline{w}_M^T \end{matrix} \end{bmatrix} \begin{bmatrix} 1 \\ \underline{y}_E \end{bmatrix}$$

with the augmented matrix

$$\underline{\underline{W}}_a = \left(\underline{w}_1^a\ \underline{w}_2^a \dots \underline{w}_M^a \right) = \begin{bmatrix} b_1 & b_2 & & b_M \\ \underline{w}_1 & \underline{w}_2 & \dots & \underline{w}_M \end{bmatrix} = \begin{bmatrix} \underline{b}^T \\ \underline{\underline{W}} \end{bmatrix}$$

that contains in the first row the biases.

Alternatively to the augmented notation we could explicitly sum the biases to the output

$$\underline{z} = \underline{\underline{W}}^T \underline{y}_E + \underline{b}$$

This is totally equivalent to the augmented notation but it will allow easier analysis of the gradient propagation.

Now the loss $L(\underline{y}, \underline{d})$ has to be differentiated with respect to $\underline{W}$, $\underline{b}$, and $\underline{\theta}$.

For the second layer we need the gradients

$$\frac{\partial L}{\partial \underline{w}_j} = \frac{\partial L}{\partial z_j} \frac{\partial z_j}{\partial \underline{w}_j} \quad ; \quad \frac{\partial L}{\partial b_j} = \frac{\partial L}{\partial z_j} \frac{\partial z_j}{\partial b_j} \qquad j = 1, \dots, M$$

Since $\frac{\partial z_j}{\partial b_j} = 1$ for the bias $\frac{\partial L}{\partial \underline{b}} = \frac{\partial L}{\partial \underline{z}}$,

Similarly since $\frac{\partial z_j}{\partial \underline{w}_j} = \underline{y}_E$, $\frac{\partial L}{\partial \underline{w}_j} = \frac{\partial L}{\partial z_j} \underline{y}_E$.

Now we need the gradient of L w.r.t. $\underline{z}$. For a generic z_j, using total gradient, we have

$$\frac{\partial L}{\partial z_j} = \sum_{i=1}^{M} \frac{\partial L}{\partial y_i} \frac{\partial y_i}{\partial z_j} = \left(\frac{\partial L}{\partial y_1} \frac{\partial L}{\partial y_2} \dots \frac{\partial L}{\partial y_M} \right) \begin{pmatrix} \frac{\partial y_1}{\partial z_j} \\ \frac{\partial y_2}{\partial z_j} \\ \vdots \\ \frac{\partial y_M}{\partial z_j} \end{pmatrix} = \left(\frac{\partial L}{\partial \underline{y}} \right)^T \frac{\partial \underline{y}}{\partial z_j}$$

In a comprehensive matrix form (gradient matrix flow) for $\underline{\underline{W}}$ we have

$$\frac{\partial L}{\partial \underline{\underline{W}}} \stackrel{\Delta}{=} \left(\frac{\partial L}{\partial \underline{w}_1} \frac{\partial L}{\partial \underline{w}_2} \cdots \frac{\partial L}{\partial \underline{w}_M}\right) = \left[\left(\frac{\partial L}{\partial \underline{y}}\right)^T \frac{\partial \underline{y}}{\partial z_1} \underline{y}_E \left(\frac{\partial L}{\partial \underline{y}}\right)^T \frac{\partial \underline{y}}{\partial z_2} \underline{y}_E \cdots \left(\frac{\partial L}{\partial \underline{y}}\right)^T \frac{\partial \underline{y}}{\partial z_M} \underline{y}_E\right]$$

$$= \left(\frac{\partial L}{\partial \underline{y}}\right)^T \left(\frac{\partial \underline{y}}{\partial \underline{z}}\right) \otimes \underline{y}_E$$

where $\frac{\partial \underline{y}}{\partial \underline{z}}$ is the Jacobian matrix

$$\left(\frac{\partial \underline{y}}{\partial \underline{z}}\right) = \begin{bmatrix} \left(\frac{\partial y_1}{\partial \underline{z}}\right)^T \\ \left(\frac{\partial y_2}{\partial \underline{z}}\right)^T \\ \left(\frac{\partial y_M}{\partial \underline{z}}\right)^T \end{bmatrix} = \begin{pmatrix} \frac{\partial y_1}{\partial z_1} & \frac{\partial y_1}{\partial z_2} & \cdots & \frac{\partial y_1}{\partial z_M} \\ \frac{\partial y_2}{\partial z_1} & \frac{\partial y_2}{\partial z_2} & & \frac{\partial y_2}{\partial z_M} \\ \frac{\partial y_M}{\partial z_1} & \frac{\partial y_M}{\partial z_2} & \cdots & \frac{\partial y_M}{\partial z_M} \end{pmatrix}$$

For the biases

$$\frac{\partial L}{\partial \underline{b}} = \frac{\partial L}{\partial \underline{z}} = \begin{pmatrix} \frac{\partial L}{\partial z_1} \\ \frac{\partial L}{\partial z_2} \\ \frac{\partial L}{\partial z_M} \end{pmatrix} = \begin{pmatrix} \frac{\partial y^T}{\partial z_1} \frac{\partial L}{\partial \underline{y}} \\ \frac{\partial y^T}{\partial z_2} \frac{\partial L}{\partial \underline{y}} \\ \frac{\partial y^T}{\partial z_M} \frac{\partial L}{\partial \underline{y}} \end{pmatrix} = \begin{pmatrix} \frac{\partial y_1}{\partial z_1} & \frac{\partial y_2}{\partial z_1} & \cdots & \frac{\partial y_M}{\partial z_1} \\ \frac{\partial y_1}{\partial z_2} & \frac{\partial y_2}{\partial z_2} & & \frac{\partial y_M}{\partial z_2} \\ \frac{\partial y_1}{\partial z_M} & \frac{\partial y_2}{\partial z_M} & \cdots & \frac{\partial y_M}{\partial z_M} \end{pmatrix} \begin{pmatrix} \frac{\partial L}{\partial y_1} \\ \frac{\partial L}{\partial y_2} \\ \frac{\partial L}{\partial y_M} \end{pmatrix}$$

$$= \left(\frac{\partial \underline{y}}{\partial \underline{z}}\right)^T \left(\frac{\partial L}{\partial \underline{y}}\right)$$

For the first layer, we have computed all the N_θ free parameters in a vector $\underline{\theta}$ and we need the gradient vector TWO .38

$$\frac{\partial L}{\partial \underline{\theta}} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} \\ \frac{\partial L}{\partial \theta_2} \\ \vdots \\ \frac{\partial L}{\partial \theta_{N_\theta}} \end{bmatrix} = \begin{bmatrix} \left(\frac{\partial L}{\partial \underline{y}_E}\right)^T \frac{\partial \underline{y}_E}{\partial \theta_1} \\ \left(\frac{\partial L}{\partial \underline{y}_E}\right)^T \frac{\partial \underline{y}_E}{\partial \theta_2} \\ \vdots \\ \left(\frac{\partial L}{\partial \underline{y}_E}\right)^T \frac{\partial \underline{y}_E}{\partial \theta_{N_\theta}} \end{bmatrix} \qquad \text{I}$$

$$\left(\frac{\partial L}{\partial \underline{y}_E}\right)^T = \left(\frac{\partial L}{\partial y_{E1}} \; \frac{\partial L}{\partial y_{E2}} \cdots \frac{\partial L}{\partial y_{EN_E}}\right)$$

$$= \left[\sum_{i=1}^{M} \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial y_{E1}}, \sum_{i=1}^{M} \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial y_{E2}}, \ldots, \sum_{i=1}^{M} \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial y_{EN_E}}\right]$$

$$= \left(\frac{\partial L}{\partial z_1} \; \frac{\partial L}{\partial z_2} \cdots \frac{\partial L}{\partial z_M}\right) \begin{pmatrix} \frac{\partial z_1}{\partial y_{E1}} & \frac{\partial z_1}{\partial y_{E2}} & \cdots & \frac{\partial z_1}{\partial y_{EN_E}} \\ \frac{\partial z_2}{\partial y_{E1}} & \frac{\partial z_2}{\partial y_{E2}} & \cdots & \frac{\partial z_2}{\partial y_{EN_E}} \\ \vdots & & & \\ \frac{\partial z_M}{\partial y_{E1}} & \frac{\partial z_M}{\partial y_{E2}} & & \frac{\partial z_M}{\partial y_{EN_E}} \end{pmatrix}$$

$$= \left[\left(\frac{\partial L}{\partial \underline{y}}\right)^T \frac{\partial \underline{y}}{\partial z_1} \left(\frac{\partial L}{\partial \underline{y}}\right)^T \frac{\partial \underline{y}}{\partial z_2} \cdots \left(\frac{\partial L}{\partial \underline{y}}\right) \frac{\partial \underline{y}}{\partial z_M}\right] \begin{bmatrix} \frac{\partial z_1}{\partial y_{E1}} & \frac{\partial z_1}{\partial y_{E2}} & \cdots & \frac{\partial z_1}{\partial y_{EN_E}} \\ \frac{\partial z_2}{\partial y_{E1}} & \frac{\partial z_2}{\partial y_{E2}} & & \frac{\partial z_2}{\partial y_{EN_E}} \\ \frac{\partial z_M}{\partial y_{E1}} & \frac{\partial z_M}{\partial y_{E2}} & & \frac{\partial z_M}{\partial y_{EN_E}} \end{bmatrix} \qquad \text{II}$$

$$= \left[\left(\frac{\partial L}{\partial \underline{y}}\right)^T \begin{bmatrix} \frac{\partial y_1}{\partial z_1} \\ \frac{\partial y_2}{\partial z_1} \\ \vdots \\ \frac{\partial y_M}{\partial z_1} \end{bmatrix} \quad \left(\frac{\partial L}{\partial \underline{y}}\right)^T \begin{bmatrix} \frac{\partial y_1}{\partial z_2} \\ \frac{\partial y_2}{\partial z_2} \\ \vdots \\ \frac{\partial y_M}{\partial z_2} \end{bmatrix} \cdots \left(\frac{\partial L}{\partial \underline{y}}\right)^T \begin{bmatrix} \frac{\partial y_1}{\partial z_M} \\ \frac{\partial y_2}{\partial z_M} \\ \vdots \\ \frac{\partial y_M}{\partial z_M} \end{bmatrix} \right] \begin{bmatrix} \frac{\partial z_1}{\partial y_{E1}} & \frac{\partial z_1}{\partial y_{E2}} & \cdots & \frac{\partial z_1}{\partial y_{ENE}} \\ \frac{\partial z_2}{\partial y_{E1}} & \frac{\partial z_2}{\partial y_{E2}} & & \frac{\partial z_2}{\partial y_{ENE}} \\ \vdots & & & \\ \frac{\partial z_M}{\partial y_{E1}} & \frac{\partial z_M}{\partial y_{E2}} & & \frac{\partial z_M}{\partial y_{ENE}} \end{bmatrix}$$

$$= \left(\frac{\partial L}{\partial \underline{y}}\right)^T \left(\frac{\partial \underline{y}}{\partial \underline{z}}\right) \left(\frac{\partial \underline{z}}{\partial \underline{y}_E}\right) \quad \text{(row vector)}$$

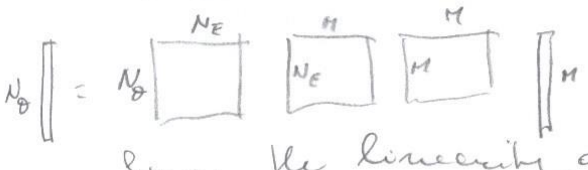
Defining the Jacobian

$$\frac{\partial \underline{y}_E}{\partial \underline{\theta}} \triangleq \begin{bmatrix} \frac{\partial y_{E1}}{\partial \theta_1} & \frac{\partial y_{E1}}{\partial \theta_2} & \cdots & \frac{\partial y_{E1}}{\partial \theta_{N_\theta}} \\ \frac{\partial y_{E2}}{\partial \theta_1} & \frac{\partial y_{E2}}{\partial \theta_2} & \cdots & \frac{\partial y_{E2}}{\partial \theta_{N_\theta}} \\ \vdots & & & \\ \frac{\partial y_{ENE}}{\partial \theta_1} & \frac{\partial y_{ENE}}{\partial \theta_2} & & \frac{\partial y_{ENE}}{\partial \theta_{N_\theta}} \end{bmatrix}$$

we have

$$\frac{\partial L}{\partial \underline{\theta}} = \left[\left(\frac{\partial L}{\partial \underline{y}}\right)^T \left(\frac{\partial \underline{y}}{\partial \underline{z}}\right) \left(\frac{\partial \underline{z}}{\partial \underline{y}_E}\right) \left(\frac{\partial \underline{y}_E}{\partial \underline{\theta}}\right) \right]^T$$

$$= \left(\frac{\partial \underline{y}_E}{\partial \underline{\theta}}\right)^T \left(\frac{\partial \underline{z}}{\partial \underline{y}_E}\right)^T \left(\frac{\partial \underline{y}}{\partial \underline{z}}\right)^T \frac{\partial L}{\partial \underline{y}}$$



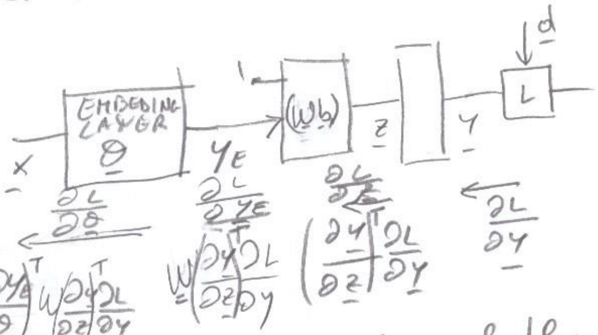
Since the linearity of the recent layer

$$\left(\frac{\partial \underline{z}}{\partial \underline{y}_E}\right)^T = \begin{pmatrix} \frac{\partial z_1}{\partial y_{E1}} & \frac{\partial z_1}{\partial y_{E2}} & \cdots & \frac{\partial z_1}{\partial y_{ENE}} \\ \frac{\partial z_2}{\partial y_{E1}} & \frac{\partial z_2}{\partial y_{E2}} & & \frac{\partial z_2}{\partial y_{ENE}} \\ \partial z & & & \end{pmatrix}^T = \begin{bmatrix} \underline{w}_1^T \\ \underline{w}_2^T \\ \vdots \\ \underline{w}_M^T \end{bmatrix}^T = \underline{\underline{W}}$$

therefore

$$\frac{\partial L}{\partial \underline{\theta}} = \left(\frac{\partial \underline{y}_E}{\partial \underline{\theta}}\right)^T \underline{\underline{W}} \left(\frac{\partial \underline{y}}{\partial \underline{z}}\right)^T \frac{\partial L}{\partial \underline{y}}$$

Note how the backward flow evolves over the network architecture



Note that going backward through the linear block the bias part carries no gradient.

This is the essence of the <u>back propagation algorithm</u>, that will be extended to multi-layer architectures, that essentially consists in propagating in the backward direction the gradient flow.

BACKWARD FLOW ON THE OUTPUT LAYER

Now we examine in more detail the 2-layer architecture for the 3 types of problems: regression, multiple binary classification and multi-class classification.

<u>Regression</u> In regression the loss function is

$$L(\underline{y},\underline{d}) = \sum_{i=1}^{M} \psi(d_i - y_i) = \sum_{i=1}^{M} \psi(\varepsilon_i)$$

and $\underline{y} = \underline{z}$. Therefore

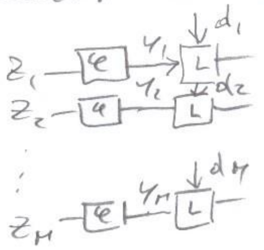
$$\frac{\partial L}{\partial \underline{y}} = \begin{bmatrix} \frac{\partial \psi(\varepsilon_1)}{\partial y_1} \\ \vdots \\ \frac{\partial \psi(\varepsilon_M)}{\partial y_M} \end{bmatrix} = -\begin{bmatrix} \psi'(\varepsilon_1) \\ \psi'(\varepsilon_2) \\ \psi'(\varepsilon_M) \end{bmatrix} = -\psi'(\underline{\varepsilon}) \quad \text{and} \quad \frac{\partial \underline{y}}{\partial \underline{z}} = \underline{\underline{I}}_M$$

The derivatives $\varphi'(\cdot)$ of the loss function have been discussed in previous chapters. For example for the quadratic loss $\varphi(\varepsilon_i) = \varepsilon_i^2$ we have simply

$$\frac{\partial L}{\partial \underline{\varepsilon}} = -2\,\underline{\varepsilon}$$

MULTIPLE BINARY CLASSIFIER

In the multiple binary classifier we have



$$y_i = \varphi(z_i), \quad i = 1, \dots M$$
$$0 \le d_i \le 1$$

The total loss is the sum of the losses on each output

$$L(\underline{y}, \underline{d}) = \sum_{i=1}^{M} L(y_i, d_i)$$

We could use standard loss functions $\varphi(\cdot)$ just like in regression

$$L(\underline{y}, \underline{d}) = \sum_{i=1}^{M} \varphi(d_i - y_i) = \sum_{i=1}^{M} \varphi(\varepsilon_i)$$

so that

$$\frac{\partial L}{\partial y} = -\varphi'(\underline{\varepsilon})$$

that for the quadratic loss is $\frac{\partial L}{\partial y} = -2\underline{\varepsilon}$.

More appropriately, using the binary crossentropy

$$L(\underline{y}, \underline{d}) = \sum_{i=1} H(d_i, y_i) = \sum_{i=1}^{M} \left(d_i \log \frac{1}{y_i} + (1-d_i) \log \frac{1}{1-y_i} \right)$$

already discussed in the chapter on linear binary classifiers. The gradient is easily computed (see Appendix)

$$\frac{\partial L}{\partial \underline{y}} = \begin{bmatrix} \dfrac{y_1 - d_1}{y_1(1-y_1)} \\ \dfrac{y_2 - d_2}{y_2(1-y_2)} \\ \vdots \\ \dfrac{y_n - d_n}{y_n(1-y_n)} \end{bmatrix}$$

The gradient ~~vector~~ matrix across the non linearities is diagonal and is

$$\frac{\partial \underline{y}}{\partial \underline{z}} = \operatorname{diag}\left(\frac{\partial y_1}{\partial z_1}, \frac{\partial y_2}{\partial z_2}, \ldots, \frac{\partial y_n}{\partial z_n}\right) = \operatorname{diag}\left(\varphi'(z_1), \varphi'(z_2), \ldots \varphi'(z_n)\right)$$

A table of sigmoidal functions is reported in Appendix X?. For the popular <u>logistic function</u>

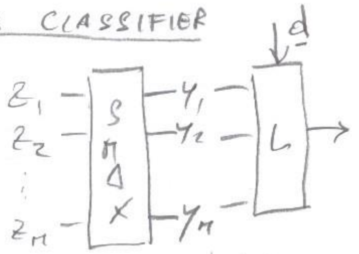
$$\varphi'(z_i) = \varphi(z_i)(1-\varphi(z_i)) \text{ , and}$$

$$\frac{\partial \underline{y}}{\partial \underline{z}} = \operatorname{diag}\left(y_1(1-y_1), y_2(1-y_2), \ldots, y_n(1-y_n)\right)$$

Note that using the cross entropy the denominators in $\frac{\partial L}{\partial \underline{y}}$ cancel out and we have simply

$$\frac{\partial L}{\partial \underline{z}} = \left(\frac{\partial \underline{y}}{\partial \underline{z}}\right)^T \frac{\partial L}{\partial \underline{y}} = \underline{y} - \underline{d}$$

MULTI-CLASS CLASSIFIER



(Recall that $\underline{d}$ may be the one-hot coded class or a smooth desired distribution.)

The total loss can be computed just as in regression, using the sum of individual losses, as

$$L(y,\underline{d}) = \sum_{i=1}^{M} \Psi(d_i - y_i) = \sum_{i=1}^{M} \Psi(\varepsilon_i)$$

that for the quadratic loss-function gives

$$L(y,d) = -2\underline{\varepsilon}$$

More appropriately using the cross entropy the loss is

$$L(\underline{y},\underline{d}) = H(\underline{d},\underline{y}) = \sum_{i=1}^{M} d_i \log \frac{d_i}{y_i}$$

and the gradient is (see Appendix)

$$\frac{\partial L}{\partial \underline{y}} = -\begin{bmatrix} \frac{d_1}{y_1} \\ \frac{d_2}{y_2} \\ \vdots \\ \frac{d_M}{y_M} \end{bmatrix} = -\frac{\underline{d}}{\underline{y}}$$

/ division element by element.

The Jacobian across the softmax function is (see Appendix)

$$\frac{\partial \underline{y}}{\partial \underline{z}} = \operatorname{diag}(\underline{y}) - \underline{y}\,\underline{y}^T .$$

Recall that both $\underline{d}$ and $\underline{y}$ are distributions, i.e. $\sum_{i=1}^{M} d_i = 1$, $\sum_{i=1}^{M} y_i = 1$. Therefore when we use the cross-entropy

TWO.44

$$\frac{\partial L}{\partial \underline{z}} = \left(\frac{\partial \underline{y}}{\partial \underline{z}}\right)^T \frac{\partial L}{\partial \underline{y}} = -\mathrm{diag}(\underline{y})\,\frac{\underline{d}}{\underline{y}} + \underline{y}\,\underline{y}^T\frac{\underline{d}}{\underline{y}}$$

$$= -\underline{d} + \begin{pmatrix} y_1^2 & y_1y_2 & \cdots & y_1y_M \\ y_2y_1 & y_2^2 & \cdots & y_2y_M \\ \\ y_My_1 & y_My_2 & \cdots & y_My_M \end{pmatrix} \begin{pmatrix} \frac{d_1}{y_1} \\ \frac{d_2}{y_2} \\ \\ \frac{d_M}{y_M} \end{pmatrix}$$

$$= -d + \begin{pmatrix} y_1d_1 + y_1 d_2 + \cdots + y_1 d_M \\ y_2d_1 + y_2 d_2 + \cdots + y_2 d_M \\ \vdots \\ y_Md_1 + y_M d_2 + \cdots + y_M d_M \end{pmatrix}$$

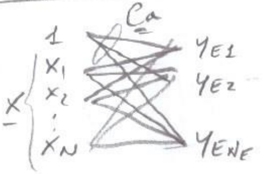
$$= -d + \underline{y} = \underline{y} - \underline{d}$$

It simplifies just as in the multiple binary classifier.

GRADIENTS FOR THE FIRST LAYER

Clearly the gradient $\frac{\partial y_E}{\partial \theta}$ depends on the type of embedding layer. Let us look at few options.

LINEAR LAYER



biases

$$y_E = \underline{\underline{C}}_a^T \underline{x}_a = \begin{bmatrix} c_{10} & \underline{c}_1^T \\ c_{20} & \underline{c}_2^T \\ \vdots & \vdots \\ c_{N_E0} & \underline{c}_{N_E}^T \end{bmatrix} \begin{bmatrix} 1 \\ \underline{x} \end{bmatrix}$$

$$\underline{\underline{C}}_a = \begin{bmatrix} c_{10} & c_{20} & \dots & c_{N_E0} \\ \underline{c}_1 & \underline{c}_2 & \dots & \underline{c}_{N_E} \end{bmatrix} = \left[\underline{c}_1^a \, \underline{c}_2^a \dots \underline{c}_{N_E}^a\right]$$

$$= \begin{bmatrix} \underline{c}_1^{aT} \\ \underline{c}_2^{aT} \\ \vdots \\ \underline{c}_{N_E}^{aT} \end{bmatrix} \begin{bmatrix} 1 \\ \underline{x} \end{bmatrix}$$

Since the following layer is linear

$$\underline{z} = \underline{W}_a^T \underline{y}_E^a = \underline{\underline{W}}_a^T \begin{bmatrix} 1 \\ \underline{\underline{C}}_a^T \underline{x}_a \end{bmatrix}$$

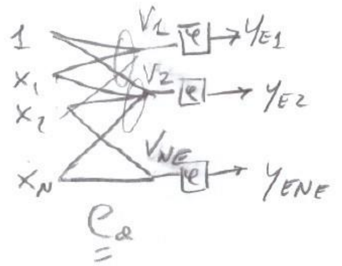
the entire mapping from $\underline{x}$ to $\underline{z}$ will be linear, and it may be useless to consider this option because $\underline{\underline{C}}_a$ could be absorbed into the second layer matrix $\underline{\underline{W}}_a$.

A possibility that could be interesting is to use the first layer as a compression layer, i.e. with $N_E << N$ as in principal component analysis. In any case looking at the connectivity

$$\frac{\partial L}{\partial \underline{c}_j^a} = \frac{\partial L}{\partial y_{Ej}} \frac{\partial y_{Ej}}{\partial \underline{c}_j^a} = \frac{\partial L}{\partial y_{Ej}} \underline{x}_a \qquad j = 1, .., N_E$$

$$\frac{\partial L}{\partial \underline{\underline{C}}^a} = \left[\frac{\partial L}{\partial \underline{c}_1^a} \, \frac{\partial L}{\partial \underline{c}_2^a} \dots \frac{\partial L}{\partial \underline{c}_{N_E}^a} \right] = \left[\frac{\partial L}{\partial y_{E1}} \underline{x}_a \, \frac{\partial L}{\partial y_{E2}} \underline{x}_a \dots \frac{\partial L}{\partial y_{EN_E}} \underline{x}_a \right] = \left(\frac{\partial L}{\partial \underline{y}_E} \right)^T \otimes \underline{x}_a$$

LINEAR LAYER WITH NONLINEARITIES



$\varphi(\cdot)$ could be RELU
Sigmoid
- - - -

$$\underline{\underline{C}}_a = \begin{bmatrix} c_{10} & c_{20} & \cdots & c_{N_E 0} \\ \underline{c}_1 & \underline{c}_2 & & \underline{c}_{N_E} \end{bmatrix} = \begin{bmatrix} \underline{c}_1^a & \underline{c}_2^a & \cdots & \underline{c}_{N_E}^a \end{bmatrix}$$

$$\underline{y}_E = \begin{bmatrix} \varphi(v_1) \\ \varphi(v_2) \\ \vdots \\ \varphi(v_{N_E}) \end{bmatrix} = \varphi(\underline{v}) = \varphi\left(\underline{\underline{C}}_a^T \underline{x}_a\right)$$

$$\frac{\partial L}{\partial \underline{c}_j^a} = \frac{\partial L}{\partial y_{Ej}} \frac{\partial y_{Ej}}{\partial \underline{c}_j^a} = \frac{\partial L}{\partial y_{Ej}} \frac{\partial y_{Ej}}{\partial v_j} \frac{\partial v_j}{\partial \underline{c}_j^a}$$

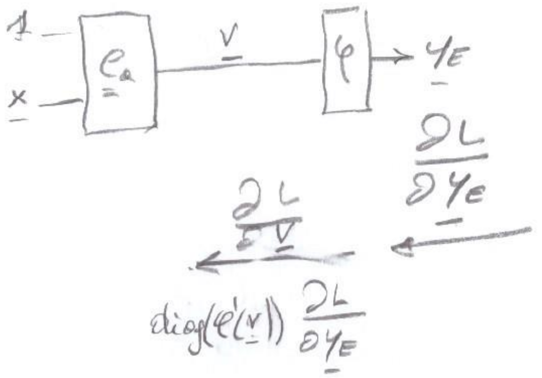
$$= \frac{\partial L}{\partial y_{Ej}} \varphi'(v_j)\, \underline{x}_a$$

$$\frac{\partial L}{\partial \underline{\underline{C}}^a} = \begin{bmatrix} \frac{\partial L}{\partial \underline{c}_1^a} & \frac{\partial L}{\partial \underline{c}_2^a} & \cdots & \frac{\partial L}{\partial \underline{c}_{N_E}^a} \end{bmatrix} = \left[\frac{\partial L}{\partial y_{E1}} \frac{\partial y_{E1}}{\partial v_1} \underline{x}_a, \frac{\partial L}{\partial y_{E2}} \frac{\partial y_{E2}}{\partial v_2} \underline{x}_a, \ldots, \frac{\partial L}{\partial y_{EN_E}} \frac{\partial y_{EN_E}}{\partial v_{N_E}} \underline{x}_a \right]$$

$$= \left[\frac{\partial L}{\partial y_{E1}} \varphi'(v_1) \underline{x}_a, \frac{\partial L}{\partial y_{E2}} \varphi'(v_2) \underline{x}_a, \ldots, \frac{\partial L}{\partial y_{EN_E}} \varphi(v_{N_E}) \underline{x}_a \right]$$

$$= \left[\frac{\partial L}{\partial v_1} \underline{x}_a, \frac{\partial L}{\partial v_2} \underline{x}_a, \ldots, \frac{\partial L}{\partial v_{N_E}} \underline{x}_a \right]$$

$$= \left(\frac{\partial L}{\partial \underline{v}}\right)^T \otimes \underline{x}_a = \left(\left(\frac{\partial L}{\partial \underline{y}_E}\right)^T \mathrm{diag}(\varphi'(\underline{v})) \right) \otimes \underline{x}_a$$



The rule is then: Once reached the linear layer output $\underline{v}$ with $\frac{\partial L}{\partial \underline{v}}$,

$$\frac{\partial L}{\partial \underline{\underline{C}}_a} = \left(\frac{\partial L}{\partial \underline{v}}\right)^T \otimes \underline{x}_a$$

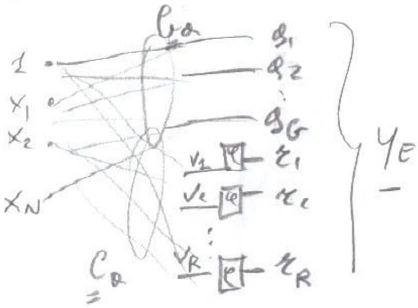
In going through a block multiply by the Jacobian transposed.

Recall that for RELUS $\varphi'(v_i) = u(v_i)$ that sets to zero some components of $\frac{\partial L}{\partial \underline{y}_E}$

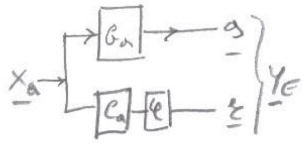
For <u>logistic</u> sigmoids $\varphi'(v_i) = \varphi(v_i)(1-\varphi(v_i))$
$= y_{Ei}(1-y_{Ei})$

For all other nonlinearities Appendix X gives expressions and derivatives.

SKIP-CONNECTION LAYER



$$\begin{cases} \underline{g} = \underline{\underline{B}}_a^T \underline{x}_a \\ \underline{r} = \varphi\left(\underline{\underline{C}}_a^T \underline{x}_a\right) \end{cases}$$



$$\underline{y}_E = \begin{pmatrix} \underline{g} \\ \underline{r} \end{pmatrix}$$

$$\frac{\partial L}{\partial \underline{y}_E} = \begin{pmatrix} \dfrac{\partial L}{\partial \underline{g}} \\[2ex] \dfrac{\partial L}{\partial \underline{r}} \end{pmatrix}$$

The backword gradient on $\underline{y}_E$ is split in two parts.

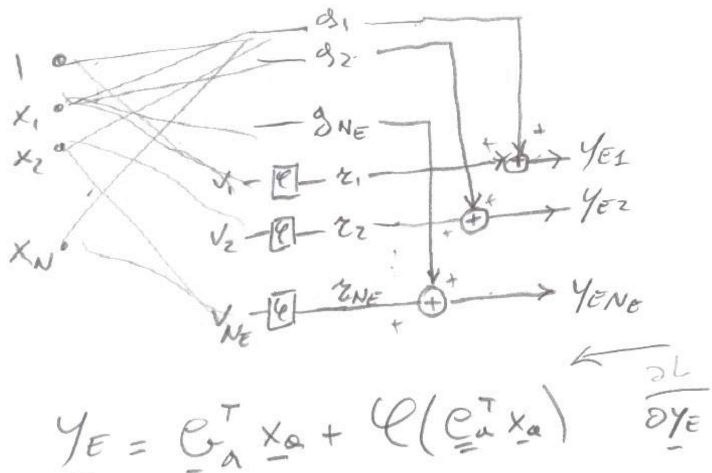
The gradient for the update of $\underline{\underline{B}}_a$, similarly to the above cases becomes

$$\frac{\partial L}{\partial \underline{\underline{B}}_a} = \left(\frac{\partial L}{\partial \underline{g}}\right)^T \otimes \underline{x}_a$$

The gradient for the update of matrix $\underline{\underline{C}}_a$ is

$$\frac{\partial L}{\partial \underline{\underline{C}}_a} = \left(\left(\frac{\partial L}{\partial \underline{r}}\right)^T \mathrm{diag}\left(\varphi'(\underline{v})\right)\right) \otimes x_a$$

RESIDUAL LAYER



$$\underline{y}_E = \underline{\underline{G}}_a^T \underline{x}_a + \varphi(\underline{\underline{C}}_a^T \underline{x}_a)$$

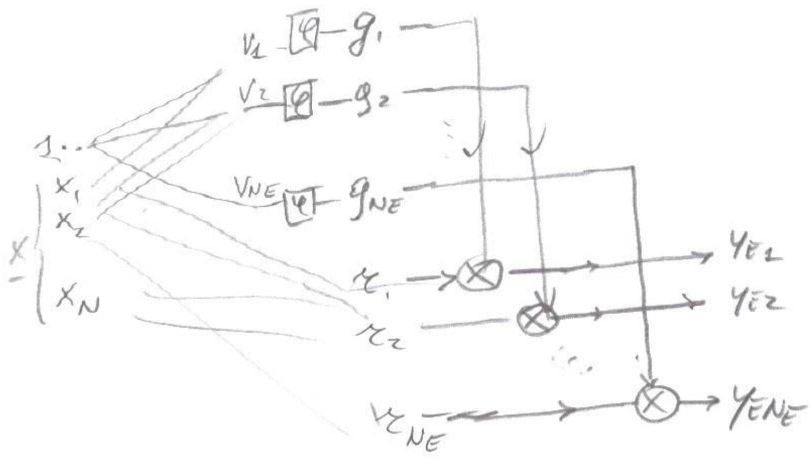
The gradient for the update of $\underline{\underline{G}}_a$ is

$$\frac{\partial L}{\partial \underline{\underline{G}}_a} = \left(\frac{\partial L}{\partial \underline{y}_E}\right)^T \otimes \underline{x}_a$$

and the one for the update of $\underline{\underline{C}}_a$

$$\frac{\partial L}{\partial \underline{\underline{C}}_a} = \left[\left(\frac{\partial L}{\partial \underline{y}_E}\right)^T \text{diag}(\varphi'(\underline{v}))\right] \otimes \underline{x}_a$$

GATED LAYER WITH SIGMOIDS



$$\underline{y}_E = \underline{g} \odot \underline{r} = \varphi\left(\underline{\underline{G}}_a^T \underline{x}_a\right) \odot \left(\underline{\underline{C}}_a^T \underline{x}_a\right)$$

$$\| N_E \qquad \| N_E$$

To go through the multiplier we can consider

$\frac{\partial L}{\partial \underline{r}}$ and $\frac{\partial L}{\partial \underline{g}}$ that give

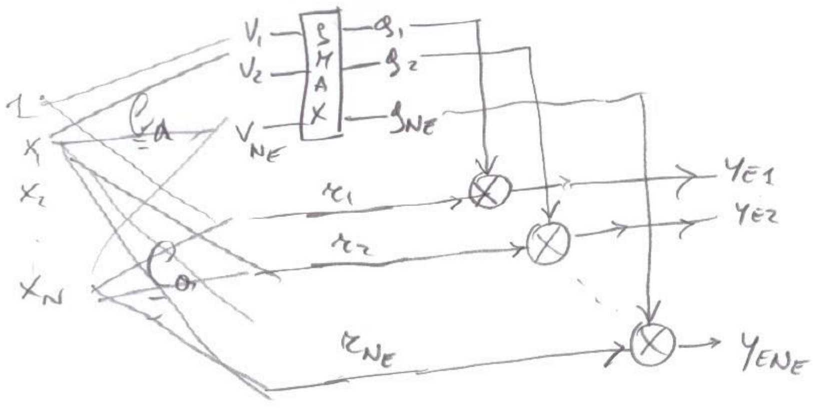
$$\frac{\partial L}{\partial \underline{r}} = \frac{\partial L}{\partial \underline{y}_E} \odot \underline{g}$$

$$\frac{\partial L}{\partial \underline{g}} = \frac{\partial L}{\partial \underline{y}_E} \odot \underline{r}$$

$$\begin{cases} \dfrac{\partial L}{\partial \underline{\underline{C}}_a} = \left(\dfrac{\partial L}{\partial \underline{r}}\right)^T \otimes \underline{x}_a = \left(g^T \odot \left(\dfrac{\partial L}{\partial \underline{y}_E}\right)^T\right) \otimes \underline{x}_a \\ \dfrac{\partial L}{\partial \underline{\underline{G}}_a} = \left(\left(\dfrac{\partial L}{\partial \underline{g}}\right)^T \mathrm{diag}\left(\varphi'(\underline{v})\right)\right) \otimes x_a = \left[\left(r^T \odot \left(\dfrac{\partial L}{\partial \underline{y}_E}\right)^T\right) \mathrm{diag}\left(\varphi'(v)\right)\right] \otimes \underline{x}_a \end{cases}$$

GATED LAYER WITH SOFTMAX

TWO.51



$$\underline{y}_E = \underline{g} \odot \underline{r} = \underline{S}_{max}\left(\underline{\underline{C}}_a^T \underline{x}_a\right) \odot \left(\underline{\underline{C}}_a^T \underline{x}_a\right)$$

For the lower branch, we have just as for the gated layer with sigmoids

$$\frac{\partial L}{\partial \underline{\underline{C}}_a} = \left(\underline{g}^T \odot \left(\frac{\partial L}{\partial \underline{y}_E}\right)^T\right) \otimes \underline{x}_a$$

For the upper branch we need the Jacobian across the softmax that we recall is

$$\frac{\partial \underline{g}}{\partial \underline{v}} = \mathrm{diag}(\underline{g}) - \underline{g}\underline{g}^T$$

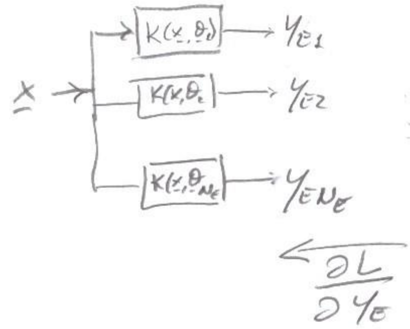
$$\frac{\partial L}{\partial \underline{\underline{C}}_a} = \left(\frac{\partial L}{\partial \underline{v}}\right)^T \otimes \underline{x}_a = \left(\left(\frac{\partial \underline{g}}{\partial \underline{v}}\right)^T \frac{\partial L}{\partial \underline{g}}\right)^T \otimes \underline{x}_a$$

$$= \left[\left(\frac{\partial L}{\partial \underline{g}}\right)^T \left(\mathrm{diag}(\underline{g}) - \underline{g}\underline{g}^T\right)\right] \otimes \underline{x}_a$$

$$= \left[\left(\left(\frac{\partial L}{\partial \underline{y}_E}\right)^T \odot \underline{r}^T\right)\left(\mathrm{diag}(\underline{g}) - \underline{g}\underline{g}^T\right)\right] \otimes \underline{x}_a$$

GRADIENT FOR SOME KERNEL-BASED ARCHITECTURES

Kernel-based architectures, such as the radial-basis functions, are usually trained in an unsupervised way, i.e. only on the input sequence $\{\underline{x}[n]\}$ of the training set.
However, their parameters, at least in principle can be adjusted according to a backward gradient flow coming from the output.



For each kernel function $K(\underline{x},\underline{\theta}_i)$ we need the gradient for $\underline{\theta}_i$

$$\frac{\partial L}{\partial \underline{\theta}_i} = \frac{\partial L}{\partial y_{Ei}} \frac{\partial y_{Ei}}{\partial \underline{\theta}_i}$$

$$\frac{\partial y_{Ei}}{\partial \underline{\theta}_i} = \frac{\partial K(\underline{x},\underline{\theta}_i)}{\partial \underline{\theta}_i}$$

For radial functions

$$K(\underline{x},\underline{\theta}_i) = K\left(\frac{\|\underline{x}-\underline{\mu}_i\|^2}{h_i}\right), \quad \underline{\theta}_i = \begin{pmatrix}\underline{\mu}_i \\ h_i\end{pmatrix}$$

where $\underline{\mu}_i$ is the cluster centers and h_i controls the size of the sphere around $\underline{\mu}_i$.

TWO.53

For gaussian radial functions

$$K(\underline{x}, \underline{\theta}_i) = e^{-\frac{(\underline{x}-\underline{\mu}_i)^T(\underline{x}-\underline{\mu}_i)}{2h_i^2}}$$

$$\frac{\partial K}{\partial \underline{\mu}_i} = -e^{-\frac{(\underline{x}-\underline{\mu}_i)^T(\underline{x}-\underline{\mu}_i)}{2h_i^2}} \frac{\partial}{\underline{\mu}_i}\left[\frac{(\underline{x}-\underline{\mu}_i)^T(\underline{x}-\underline{\mu}_i)}{2h_i^2}\right]$$

$$= -e^{-\frac{(\underline{x}-\underline{\mu}_i)^T(\underline{x}-\underline{\mu}_i)}{2h_i^2}} \frac{\partial}{\partial \underline{\mu}_i}\left[\frac{\underline{x}^T\underline{x} - 2\mu_i^T\underline{x} + \mu_i^T\mu_i}{2h_i^2}\right]$$

$$= e^{-\frac{(\underline{x}-\underline{\mu}_i)^T(\underline{x}-\underline{\mu}_i)}{2h_i^2}} \frac{\cancel{2}(\underline{x}-\underline{\mu}_i)}{\cancel{2}h_i^2} = y_{Ei}\left(\frac{\underline{x}-\underline{\mu}_i}{h_i^2}\right)$$

$$\frac{\partial K}{\partial h_i} = -e^{-\frac{(\underline{x}-\underline{\mu}_i)^T(\underline{x}-\underline{\mu}_i)}{2h_i^2}} \frac{\partial}{\partial h_i}\left(\frac{(\underline{x}-\underline{\mu}_i)^T(\underline{x}-\underline{\mu}_i)}{2h_i^2}\right)$$

$$= -e^{-\frac{(\underline{x}-\underline{\mu}_i)^T(\underline{x}-\underline{\mu}_i)}{2h_i^2}} \frac{1}{\cancel{2}}(\underline{x}-\underline{\mu}_i)^T(\underline{x}-\underline{\mu}_i)\left(-\frac{\cancel{2}}{h_i^3}\right)$$

$$= y_{Ei}\frac{\|\underline{x}-\underline{\mu}_i\|^2}{h_i^3}$$

$$\begin{cases} \dfrac{\partial K}{\partial \underline{\mu}_i} = y_{Ei}\dfrac{\underline{x}-\underline{\mu}_i}{h_i^2} \\[2ex] \dfrac{\partial K}{\partial h_i} = y_{Ei}\dfrac{\|\underline{x}-\underline{\mu}_i\|^2}{h_i^3} \end{cases}$$

FOR Multiquadratic functions

$$K(\underline{x}, \underline{\theta}_i) = \sqrt{\|\underline{x}-\underline{\mu}_i\|^2+\beta_i^2}$$

$$\frac{\partial K}{\partial \underline{\mu}_i} = \frac{-(\underline{x}-\underline{\mu}_i)}{\sqrt{\|\underline{x}-\underline{\mu}_i\|^2+\beta_i^2}} = -\gamma_{Ei}(\underline{x}-\underline{\mu}_i)$$

$$\frac{\partial K}{\partial \beta_i} = \frac{\beta_i}{\sqrt{\|\underline{x}-\underline{\mu}_i\|^2+\beta_i^2}} = \gamma_{Ei}\beta_i$$

$$\frac{\partial \|\underline{x}-\underline{\mu}_i\|^2}{\partial \underline{\mu}_i} = \frac{\partial}{\partial \underline{\mu}_i}\left(\underline{x}^T\underline{x} - 2\underline{\mu}_i^T\underline{x} + \underline{\mu}_i^T\underline{\mu}_i\right) = -2\underline{x} + 2\underline{\mu}_i = 2(\underline{\mu}_i - \underline{x})$$

FOR the INVERSE MULTIQUADRATIC

$$K(\underline{x}, \underline{\theta}_i) = \frac{1}{\sqrt{\|\underline{x}-\underline{\mu}_i\|^2+\beta_i^2}}$$

$$\frac{\partial K}{\partial \underline{\mu}_i} = +\frac{1}{\cancel{2}}\frac{1}{\left(\|\underline{x}-\underline{\mu}_i\|^2+\beta_i^2\right)^{3/2}}\cancel{2}(\underline{x}-\underline{\mu}_i)$$

$$= \gamma_{Ei}^3(\underline{x}-\underline{\mu}_i)$$

$$\frac{\partial K}{\partial \beta_i} = -\frac{1}{\cancel{2}}\frac{\cancel{2}\beta_i}{\left(\|\underline{x}-\underline{\mu}_i\|^2+\beta_i^2\right)^{3/2}} = -\gamma_{Ei}^3\beta_i$$