TWO-LAYER ARCHITECTURES

Prof. FRANCESCO A. N. PALMIERI

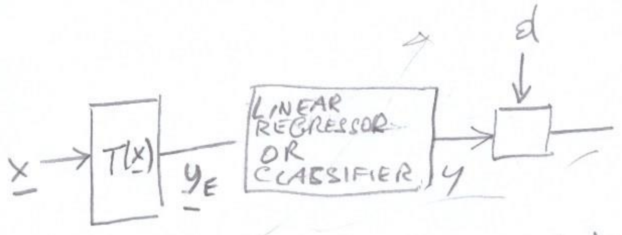
UNIVERSITA` DEGLI STUDI DELLA CAMPANIA
"LUIGI VANVITELLI"

[CORSO DI SIGNAL PROCESSING
AND DATA FUSION
AA 2024-25]

We have seen in the previous chapters how linear ^{ML1} regressors or classifiers can provide a first solution to our supervised learning problem.
More specifically, linear regressors that have been around for decades in the statistical and signal processing literature, may already provide satisfactory solutions in many applications (adaptive filtering, equalization, regression of time series, etc.)
Similarly, linear classifiers with their partitions of the input space in regions delimited by hyperplanes do provide solutions in many cases. Think of receivers for digital modulation on Gaussian channels, some limited image classification problems, etc.
However when linear solutions show their limitations, we are pushed to consider more flexible families of non linear parametric functions.
In any case, (it is good practice in any problem to test first the linear (relatively) simpler solution before moving to more complicated architectures. Performances of the linear networks could always be used as baseline for comparison.
In this chapter we will talk about two-layer architecture showing that a first layer followed by a linear regressor or classifier, provides already a quite powerful scheme for implementing complex mappings.
Networks with more than two layers will be studied in following chapters.

LINEAR NETWORKS WITH A FIXED EMBEDDING SPACE

A first idea to build a parametric non linear function is to transform the input space into a predefined embedding space on which the linear regressor or classifier can be applied.



The initial transformation $\underline{y}_E = \underline{T}(\underline{x})$ is a fixed vector function, it's non linear, and it may be determined by our intuition on how to transform the input space into a new representation that may be more suitable for a linear operator. The size of $\underline{y}_E$ may be larger or smaller than the size of $\underline{x}$.
The embedding space $\underline{y}_E \in \mathcal{Y}_E$ may provide more options to the linear stage that would (linearly) combine non linear transformations of the components of $\underline{x}$.

Example
Assume $\underline{x}$ to be 3-dimensional $\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ and $\underline{T}(\underline{x})$ to be a function that computes squares and cross products to be added to $\underline{x}$. More specifically

$$\underline{y}_E = \underline{T}(\underline{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_1^2 \\ x_2^2 \\ x_3^2 \\ x_1x_2 \\ x_1x_3 \\ x_2x_3 \end{bmatrix}$$

. Now the linear estimator can work on linear combinations of these elements

$$y_i = \underbrace{w_{0i} + w_{1i}x_1 + w_{2i}x_2 + w_{3i}x_3}_{\text{linear part.}} + \underbrace{w_{4i}x_1^2 + w_{5i}x_2^2 + w_{6i}x_3^2 + w_{7i}x_1x_2 + w_{8i}x_1x_3 + w_{9i}x_2x_3}_{\text{non linear part}}$$

Regressors and classifiers have now the opportunity to adaptively include the non linear parts, if necessary.

[This example is a truncated Volterra expansion widely used in numerical methods.]

Other examples may be considered where $\underline{T}(\underline{x})$ essentially extracts "relevant features" from $\underline{x}$. For example if $\underline{x}$ is an image (vectorised), $\underline{T}(\underline{x})$ may consist into a number of characteristics computed from the image and made available to the linear classifier.

Nothing changes in the adaptation of the linear part with respect to the discussion presented in the previous chapters, except that here the input is a transformed version of $\underline{x}[n]$. The training set is then

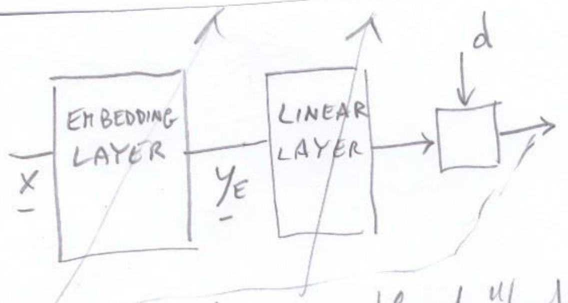
$$\mathcal{T} = \{ (\underline{d}[n], \underline{T}(\underline{x}[n])), n = 1, \ldots, n_T \}$$

The quest for the most appropriate embedding space has characterized decades of research, specially in image processing problems. For example all ~~kinds of~~ parameters can be extracted from an image to be fed to a linear classifier (image histograms, edge orientations,

Key points, etc). However the challenge to find the NL4
appropriate $\underline{T}(\underline{x})$ has achieved limited success
$\overset{\text{only}}{\text{in}}$ specific application areas. This is the reason
why the scientific community has been pushed to
consider architectures where also the embedding
space, or even multiple embedding spaces, can be
learned from data.

ADAPTIVE FIRST LAYER

The first modification to the two-layer architecture discussed in the previous section, is to make also the first layer "learnable".



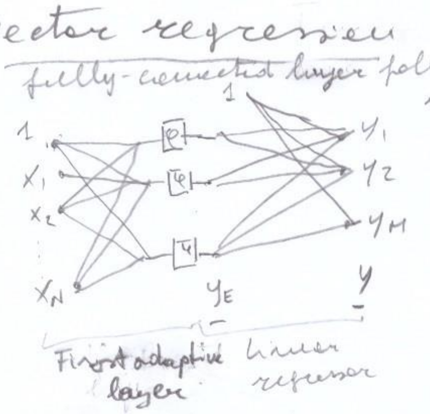
(*)

The peculiarity here is that "both" stages are adaptive and optimized on the training set.
While the choice of the second layer (linear) is imposed by the type of required outputs (regression, multiple binary classification, M-ary classification), the choice of the architecture of the first layer may be quite challenging. We need to choose the size N_E of y_E and the parametric functions for the embedding layer.

The following three neural networks are first examples of such a generalization.
Note that there are many other choices for the first layer. Some of them will be considered in the following sections. However the three architectures considered here are the most typical ones and lots of research attention has been dedicated to them for their peculiarities

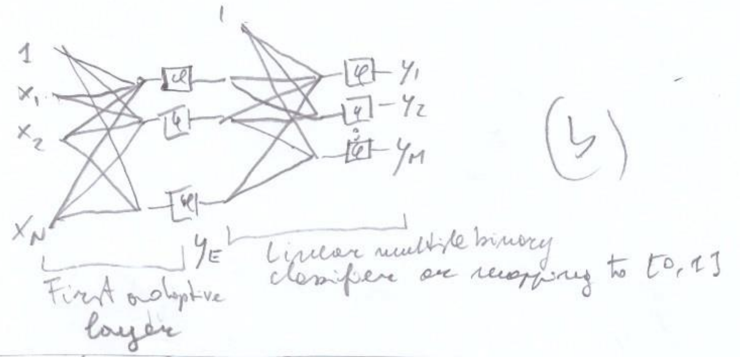
For vector regression the following architecture has a fully-connected layer followed by the standard linear regressor.

TWD.2



(a)

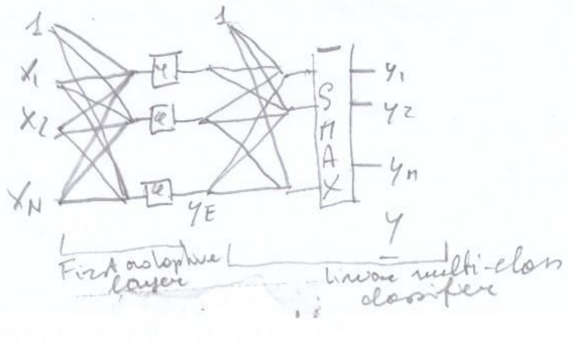
For multiple binary classification, or regression on the hypercube $[0,1]^n$, we could consider the following architecture with activation functions at the output.



(b)

For a multi-class classification

TWD.2



(c)

In all cases of Figures (a)(b) and (C) we have introduced a layer with N_E linear combiners and activation functions at the outputs.
Clearly we need to choose what type of activation functions we are going to use.
Many experiments have been conducted in the last decades on the most diverse applications and datasets. A rich catalog of activation function is available in Appendix x. We can use sigmoids, RELUS, Swish, etc and we will focus on the peculiarities of some of them in the following sections.
In any case, the results reported in the scientific literature of the last 40 years, has pointed to the crucial role of the double-layer architecture.
The size of the embedding space is a crucial choice because it can be very large in comparison to the input space $N_E \gg N$ and also $N_E \ll N$.
The reason for this lays in the fact that the first layer has to "unfold" the information contained in the input space in a form to be suitable for linear regression or classification
Attempts with a fixed first layer have found limited success.(*) Generally the embedding space should be learned and it may
. data distribution
be very different for each application.
From a theoretical point of view the double-layer approach has been supported by

* It should be mentioned that some limited success may be obtained with a random first layer with N_e very large.

a number of so-called Representation Theorems that aim at demonstrating mathematically that two-layer networks may be sufficient to represent with arbitrary accuracy any nonlinear function.

The most famous result due to Cybenko [1989][(x)] can be exemplified in the following statement (adapted)

THEOREM Let $\varphi(z)$ be a sigmoidal function (such that $\varphi(z) \xrightarrow{z\to\infty} 1$ $\varphi(z) \xrightarrow{z\to-\infty} 0$) then the finite sums of the form

$$G(\underline{x}) = \sum_{j=1}^{N_E} \alpha_j \, \varphi(\underline{c}_j^T \underline{x} + b_j)$$

are dense in $C(I_N)$. (I_N denotes the N-dimensional cube $I_N = [0,1]^N$ and $C(I_N)$ is the set of continuous functions on it). In other words given any function $f(\underline{x}) \in C(I_n)$ and an $\varepsilon > 0$, there exist a sum $G(\underline{x})$ of the above form for which

$$|G(\underline{x}) - f(\underline{x})| < \varepsilon \qquad \forall \ \underline{x} \in I_N$$

(x) G. Cybenko, "Approximation by Superposition of a Sigmoidal function", Mathematics of Control Signals and Systems. Vol 2, pp 303-314, 1989.

The proof of the theorem can be found in the paper,
but for our introductory purpos we would
like just emphasize the importance of
this result. The function $G(\underline{x})$ is exactly the
architecture of Figure 1(a) with $M=1$ ~~and~~ this result
confers to the two-layer structure a
property of universality as far as N_E is
chosen appropriately large. Note that the classifiers
in Figure 1(b) and (c) can benefit from the
same result since they can use the two
layers to build any sufficient statistics for
their decision. A few comments are appropriate:

(1) Even if the theorem is stated for
functions defined on the hypercube I_N, in
practice any function used in the applications
is defined on a limited domain that can
be easily scaled and shifted (trivial affine transformation)
to be mapped into I_N.

(2) The theorem is an approximation theorem,
meaning that since the parameters
$\underline{c}, \underline{b}, \underline{\alpha}$ can be found with any optimization
algorithm, the crucial role is played by
the size N_E, that may need to be quite large to
obtain the desider accuracy.

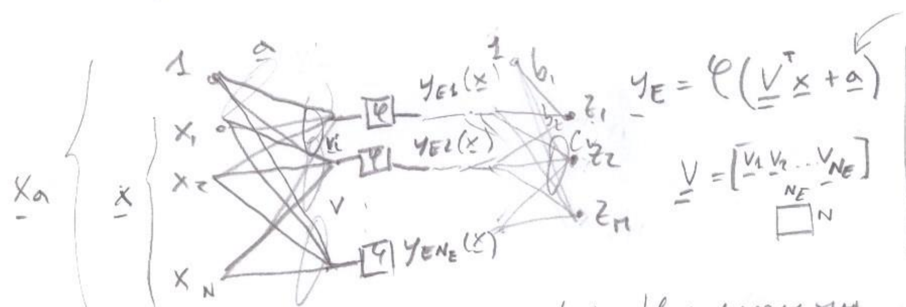
the literature is full of contributions on
representation capabilities of neural networks
that explore the use of different nonlinearities |
$\varphi(\cdot)$, multiple layers, sparse matrices, etc in
an attempt to suggest to the designer the most
/ appropriate architecture. For extension of Cybenko's theorem
we refer the interested student to the relevant literature.
Unfortunately the situation at the current state
of the art is not so simple, because many
aspects of the representation capabilities of
these kind of systems are not yet fully
understood. Therefore the users of these
systems must leverage between totally
heuristics-based trial-and-error approaches, to
partial comprehension of the functioning
of some of the network building blocks.

The reader should be warned that the extensive
functional capabilities of the neural network architecture,
are only a part of the whole story. We have pointed
out in previous chapter how the networks that
result from the training set may be useless, unless
they are able to "generalize" on data not seen during
learning. This issue is not addressed by the
representation theorems. The "regularity" of the
solution" is ~~very~~ usually difficult to assess from a rediretto
theoretical point of view. Common practice is to contain
the number of free parameters to avoid overfitting,
but this is only approximately related to generalization.

In the following, we concentrate $\overset{on}{\wedge}$ a few typical ~~[illegible]~~ architectures with $\overset{two}{\wedge}$ ~~different~~ embedding activation ~~(or~~ functions to gain an approximate understanding of their functional capabilities. The examples are displayed in two dimensions for visualization. TWO.7

EMBEDDINGS WITH SIGMOIDAL FUNCTIONS

Consider first the architectures ~~of~~ Figure 1 with sigmoidal functions in the first layer.



$$y_E = \varphi(\underline{V}^T \underline{x} + \underline{a})$$

$$\underline{V} = [\underline{v}_1 \, \underline{v}_2 \dots \underline{v}_{N_E}]$$

$$\underset{N_E}{\square}\,N$$

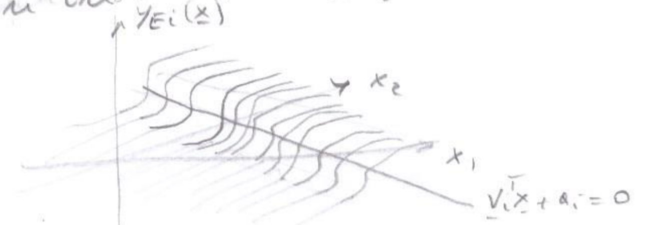
biases

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

Each embedding component is the mapping

$$y_{Ei}(\underline{x}) = \varphi(\underline{v}_i^T \underline{x} + a_i) \qquad i = 1, .., N_E$$

with $\varphi(\cdot)$ being a sigmoidal function. $y_{Ei}(\underline{x})$ is a "ridge" function on the hyperplane $\underline{v}_i^T\underline{x} + a_i = 0$ as shown in the following figure for $N=2$.



The transition is sharp or smooth according to the steepness of the sigmoidal function. These embedding functions $y_{E1}(x), .., y_{EN_E}(x)$ are now the basis functions for the second layer that combines them linearly.

(a) Note that the steepness of a sigmoid functions is easily controlled $\overset{by}{\wedge}$ scaling its input.

The transition $\varphi(\xi) \rightarrow \varphi(\alpha\xi) \quad \alpha > 0$ is steeper if $\alpha > 1$ and smoother if $0 < \alpha < 1$.

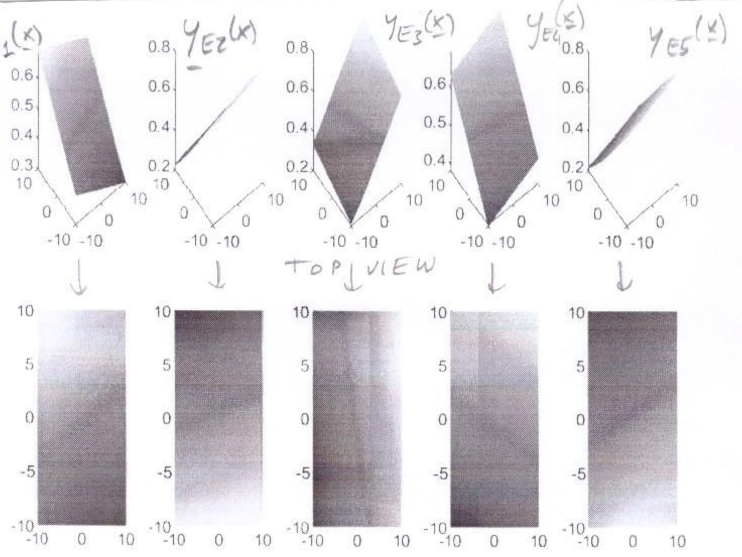
Obviously scaling the linear transformation $\alpha(\underline{V}^T\underline{x} + \underline{a})$ does not change the hyperplane position.

The following figures shows 3 sets of embeddings for randomly chosen $\underline{V}$ and $\underline{a}$ (Gaussian samples with zero mean and unit variance), with $N_E = 5$, $M = 3$. TWO.8

The 3 sets correspond to 3 different choices of scaling. The range for $\underline{x}$ is $[-10, 10]^2$ and the pictures correspond to the 3 scalings $\alpha = 0.1, 1, 10$.

Note that for $\alpha = 0.1$ the embedding functions are almost linear; for $\alpha = 1$ they have smooth ridges and for $\alpha = 10$ they are almost step functions.

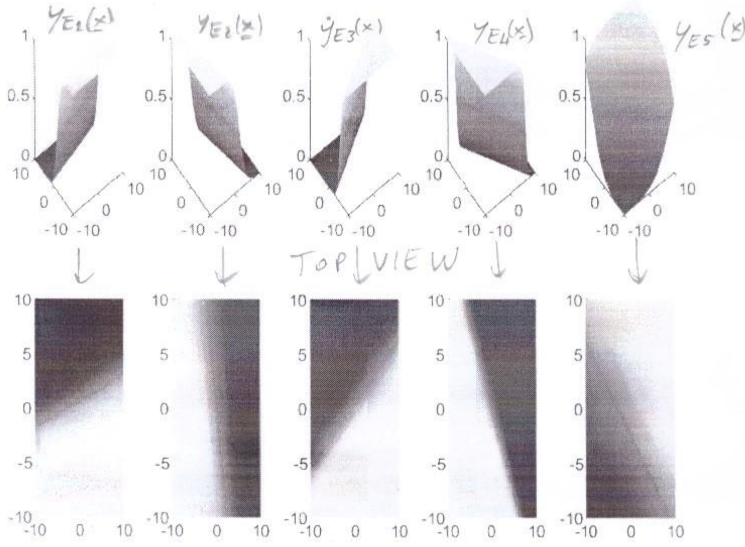
TWO.9



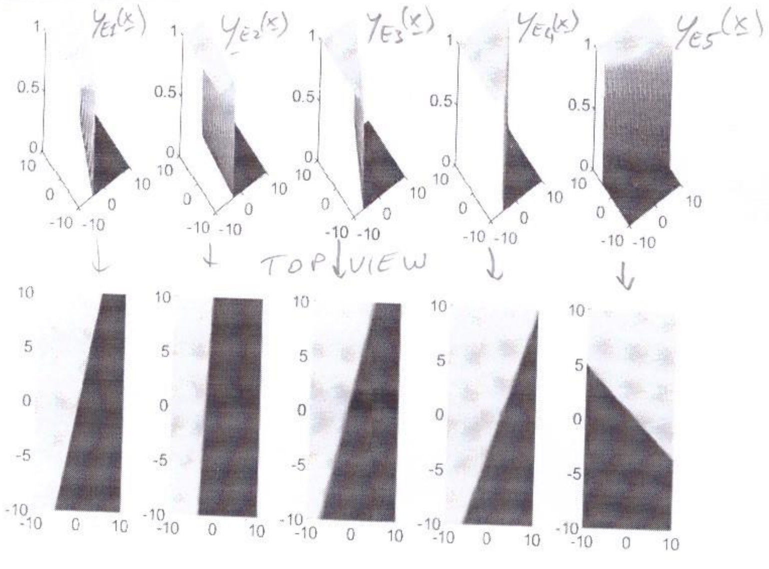
$\alpha = 0.1$

EMBEDDINGS

(*)



$\alpha = 1$



$\alpha = 10$

The N_E basis functions are linearly combined M times

$$\begin{cases} z_1(\underline{x}) = \underline{c}_1^T \underline{y}_E(\underline{x}) + b_1 \\ z_2(\underline{x}) = \underline{c}_2^T \underline{y}_E(\underline{x}) + b_2 \\ \\ z_M(\underline{x}) = \underline{c}_M^T \underline{y}_E(\underline{x}) + b_M \end{cases}$$

to be used as regressors or as statistics for classification.

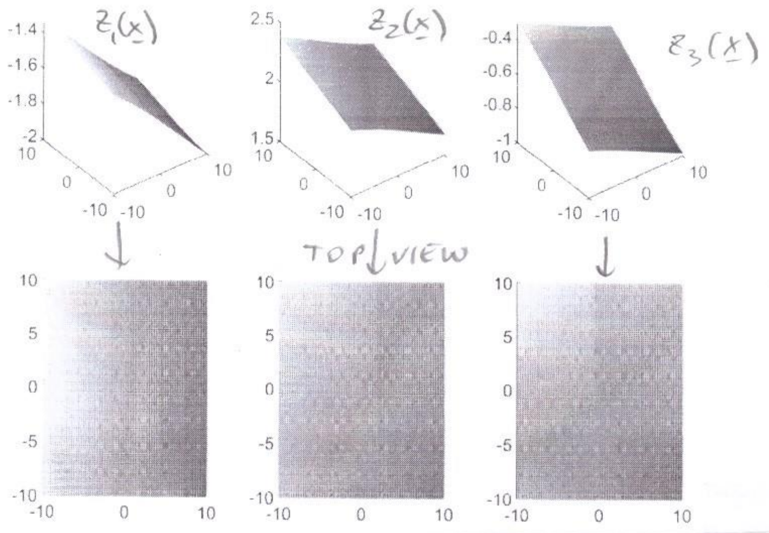
The following 3 sets of figures show 3 sets of $M=3$ linear combinations for $\alpha=0.1, 1, 10$ with random weights in $\underline{c}$ and $\underline{b}$ (gaussian samples with zero mean and unit variance) using the same methodology of Figure (A)

Note that using $\alpha=0.1$ in the sigmoids of the first layer we can only produce almost-linear functions.

For $\alpha=1$ the smooth "ridges" in the basis functions are combined to produce smooth non linear functions. Note the plateaux arising from the saturation regions of the sigmoids.

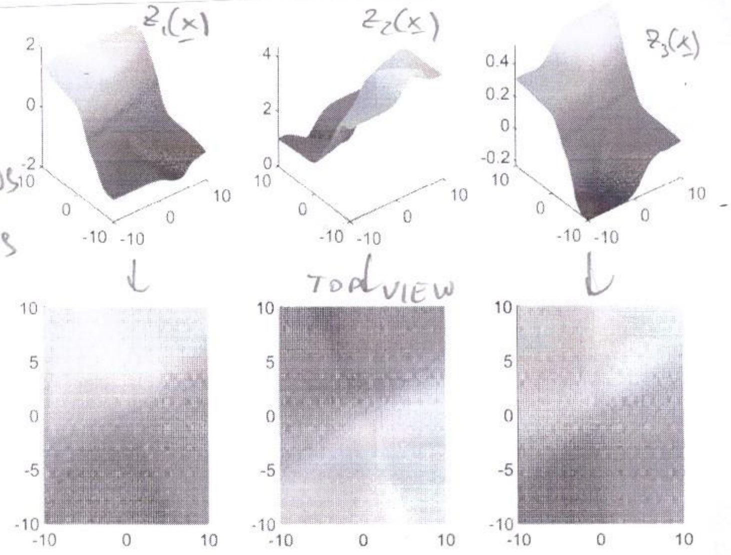
When $\alpha=10$, since the basis functions are sharp ridges, their linear combinations can only produce "flat-top" functions.

TWO.11

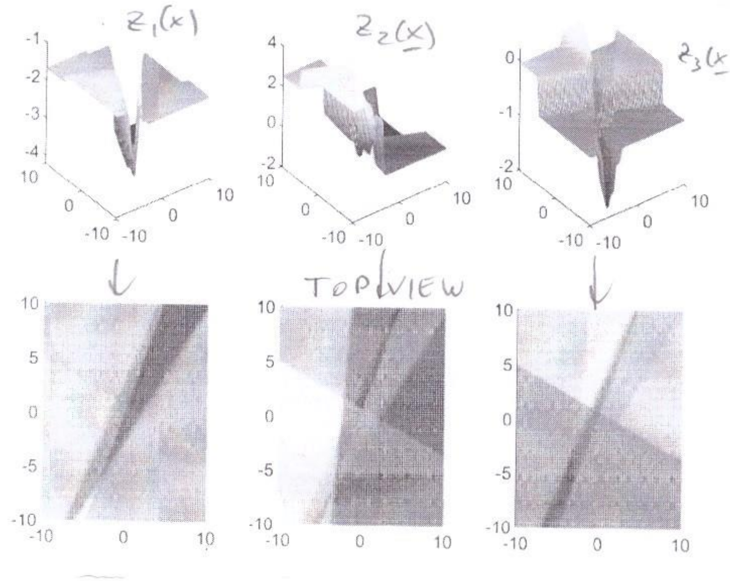


$\alpha = 0.1$

LINEAR COMBINATIONS OF EMBEDDINGS (**)



$\alpha = 1$



$\alpha = 10$

MULTIPLE BINARY CLASSIFIERS

In the architecture of Figure 1(b), the final layer gives at the output the smooth binary classifiers

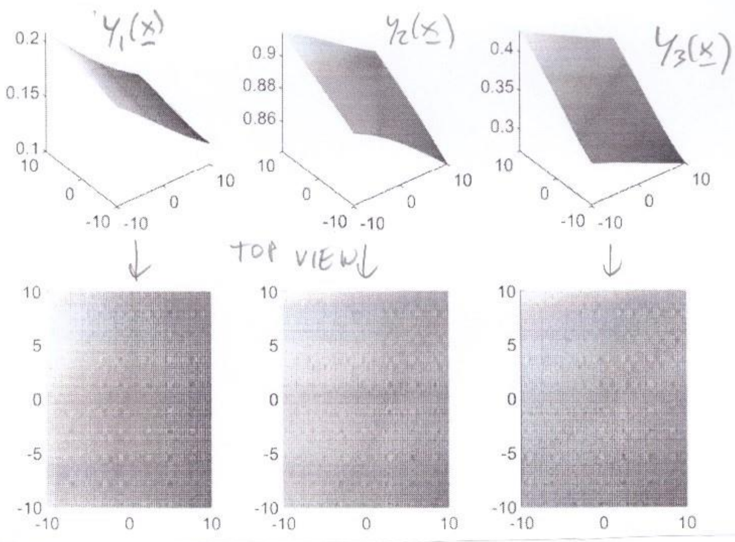
$$\begin{cases} y_1(\underline{x}) = \varphi\left(\underline{c}_1^T \underline{y}_E(\underline{x}) + b_1\right) = \varphi(z_1(\underline{x})) \\ y_2(\underline{x}) = \varphi\left(\underline{c}_2^T \underline{y}_E(\underline{x}) + b_2\right) = \varphi(z_2(\underline{x})) \\ \vdots \qquad\qquad\qquad\qquad\qquad \vdots \\ y_M(x) = \varphi\left(c_M^T \underline{y}_E(\underline{x}) + b_M\right) = \varphi(z_M(\underline{x})) \end{cases}$$

These are just the thresholded functions $z_i(\underline{x})$

The following set of Figures show the outputs of the sigmoids (in $[0,1]$) for the embeddings of the previous page. Note that these transformation do not change very much the functions $z_1(\underline{x})$, $z_2(\underline{x})$ and $z_3(\underline{x})$. Again when $\alpha = 0.1$ everything is almost linear. When $\alpha = 1$ the smooth flat regions remain with partitions of the input space made just a bit sharper. For $\alpha = 10$, the embedding $z_1(\underline{x})$, $z_2(\underline{x})$ $z_3(\underline{x})$ are already quite sharp and they are essentially replicated in $y_1(\underline{x})$ $y_2(\underline{x})$ and $y_3(\underline{x})$.

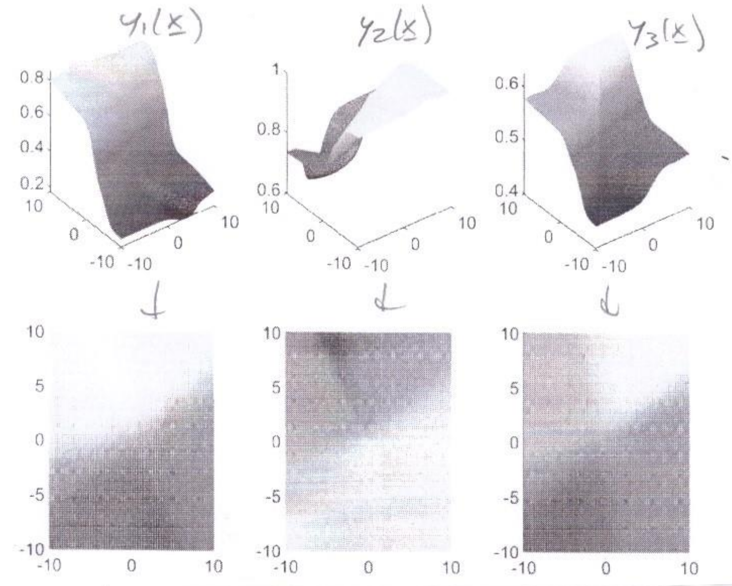
We have not changed here the sharpness of the final sigmoids ($y_i = \varphi(z_i) = \frac{1}{1+e^{-z_i}}$). Had we done so the final functions would be just 0-1 functions.

TWO.13



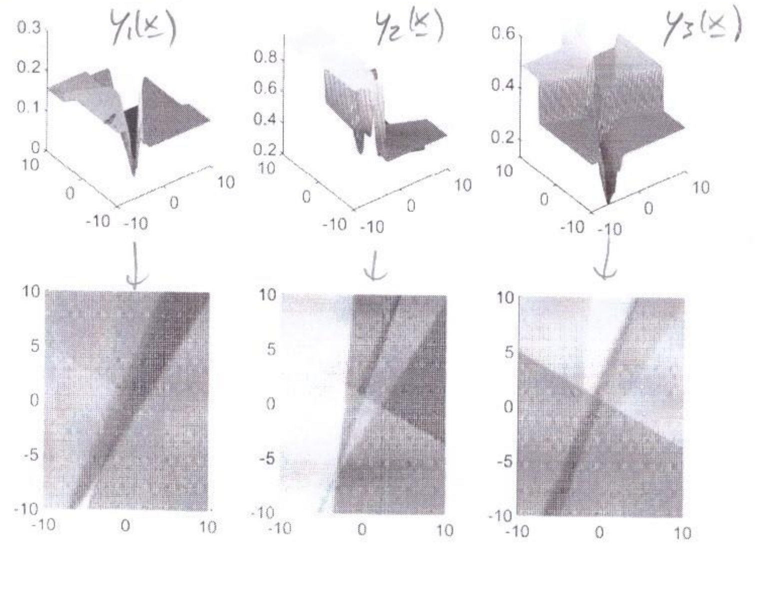
$\alpha = 0.1$

MULTIPLE BINARY CLASSIFIERS OUTPUTS (~~XXX~~)



$\alpha = 1$

Multiple binary



$\alpha = 10$

MULTI-CLASS CLASSIFICATION

In the architecture of Figure 1(c), the last stage is a softmax that gives

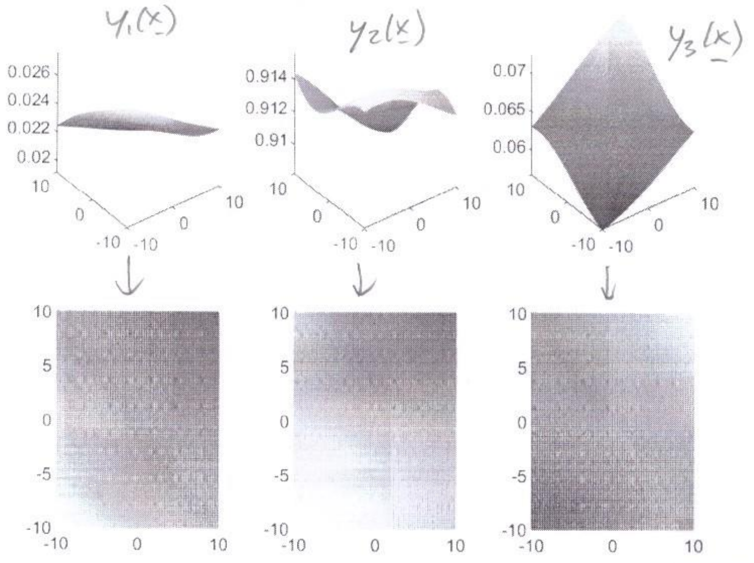
$$\underline{y}(\underline{x}) = S_{max}\begin{pmatrix} z_1(\underline{x}) \\ z_2(\underline{x}) \\ \vdots \\ z_M(\underline{x}) \end{pmatrix} = S_{max}\left(\underline{\underline{C}}^T \underline{y}_E(\underline{x}) + \underline{b}\right)$$

Recall that $0 < y_i(\underline{x}) < 1$ and $\sum_{i=1}^{M} y_i(\underline{x}) = 1$. The outputs are then interpretable as the posterior probabilities for the M classes.

Note that the functions $z_1(\underline{x}), z_2(\underline{x}), \ldots z_M(\underline{x})$, from the model-based chapters, are interpretable as the log-likelihoods of the M classes in an ML classifier. If we had priors, they would be the log-likelihoods + log-priors in a MAP classifier. Essentially the first layer of Figure 1(c) "prepares" the basis for the log-likelihoods of the linear classifier.

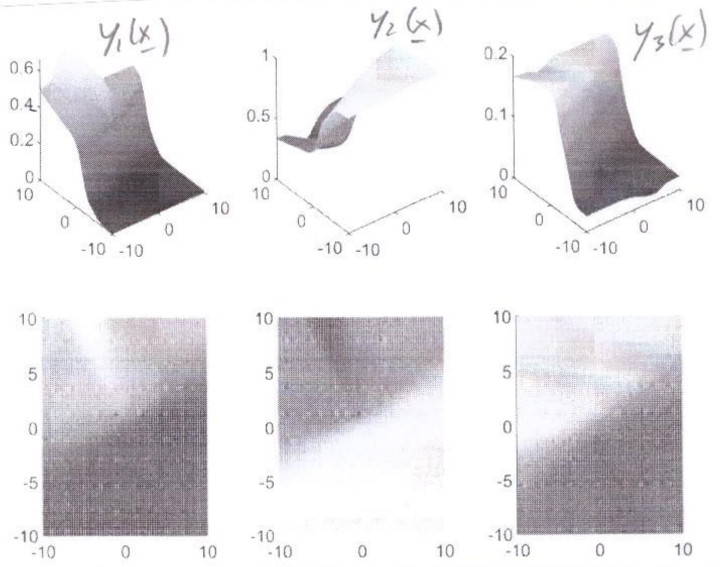
The following figure $^{(\times\times\times\times)}$ shows the output of the softmax for the linear combinations of embeddings in Figure (××). As expected, for $\alpha = 0.1$ the functions are almost linear. For $\alpha = 1$, $z_1(\underline{x}), z_2(\underline{x})$ and $z_3(\underline{x})$ "compete" to give the three posteriors $y_1(\underline{x})$ $y_2(\underline{x})$ and $y_3(\underline{x})$ with a smooth partition of the input space. For $\alpha = 10$, being z_1, z_2 and z_3 already sharp, they provide a sharper partition of the space of $\underline{x}$

TW 0.15

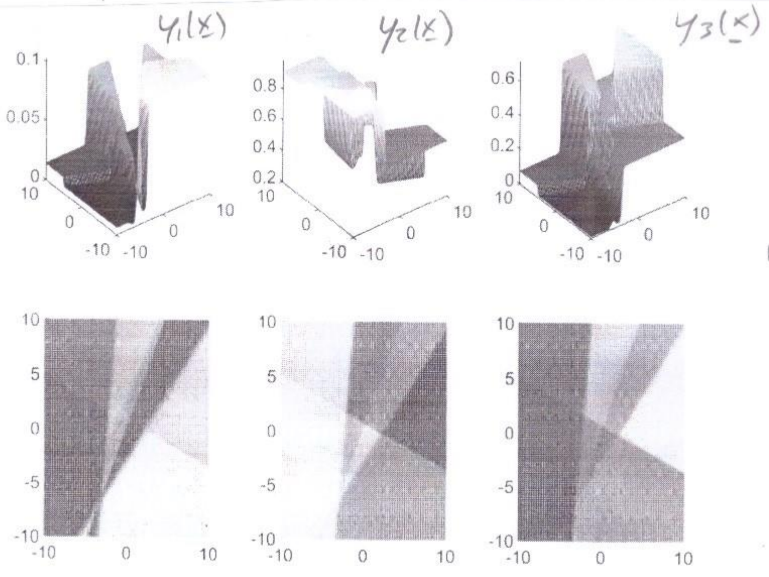


$\alpha = 0.1$

MULTI-CLASS CLASSIFIER (SOFTMAX OUTPUTS) (★★★★)



$\alpha = 1$



$\alpha = 10$

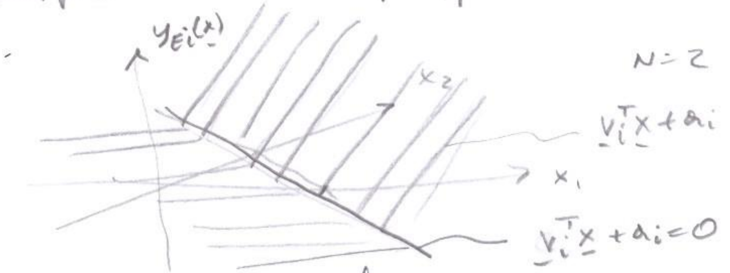
EMBEDDINGS WITH RELU FUNCTIONS

It is useful to consider also another common choice for the activation functions, where ~~that~~ the output of the first layer in Fig 1, is the RELU

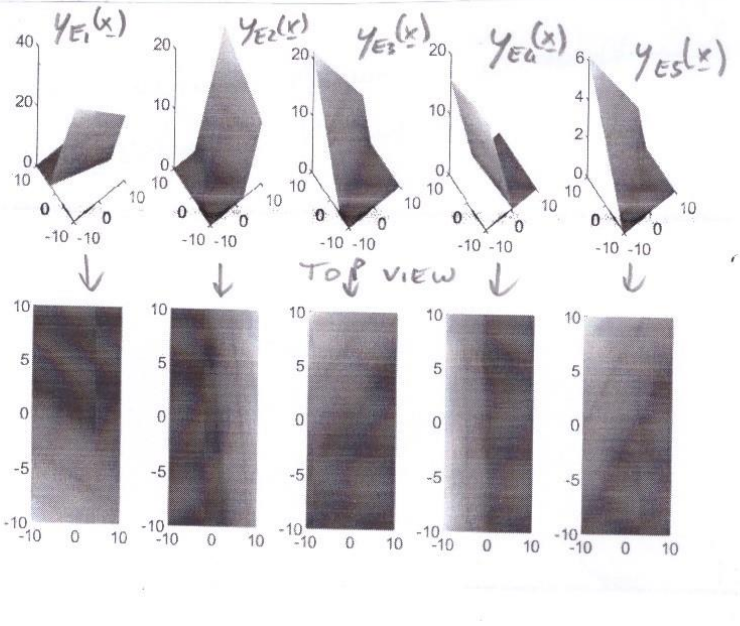
$$y_{Ei}(\underline{x}) = r(\underline{v}_i^T\underline{x} + a_i) \qquad i = 1, .., N_E$$

where $r(\xi) = \begin{cases} \xi & \xi \geq 0 \\ 0 & \text{else} \end{cases}$ ($r(\xi)$ is also called "ramp" or "rectifier")

Each output is a "half-hyperplane" function


$y_{Ei}(\underline{x})$
x_2
$N=2$
$\underline{v}_i^T\underline{x}+a_i$
x_1
$\underline{v}_i^T\underline{x}+a_i=0$

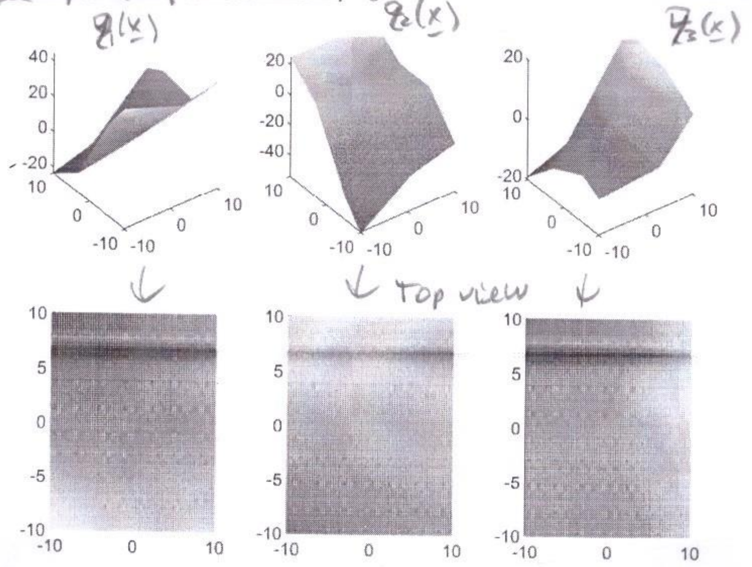
depicted in the figure for $N=2$.
The following figure shows 5 embedding functions for randomly chosen $\underline{V}$ and $\underline{a}$.



The following M linear combinations

$$z_1(\underline{x}) = \underline{c}_1^T \underline{y}_E(\underline{x}) + b_1$$
$$z_2(\underline{x}) = \underline{c}_2^T \underline{y}_E(\underline{x}) + b_2$$
$$\vdots$$
$$z_M(\underline{x}) = \underline{c}_M^T \underline{y}_E(\underline{x}) + b_M$$

are then piece-wise linear and look like "rooftops" with flat tiles. Quite arbitrarily-shaped functions can be built. The following figure shows three random superpositions of the five embeddings shown in the previous page.



Note the flat faces.

$z_1(\underline{x}), \dots, z_M(\underline{x})$ can be used as regressors, and as just said their shapes can be made quite arbitrary increasing the number of units N_E in the first layer.

Architectures with RELUs are usually preferred to those that have sigmoids in the embedding stage because the functions do not saturate.

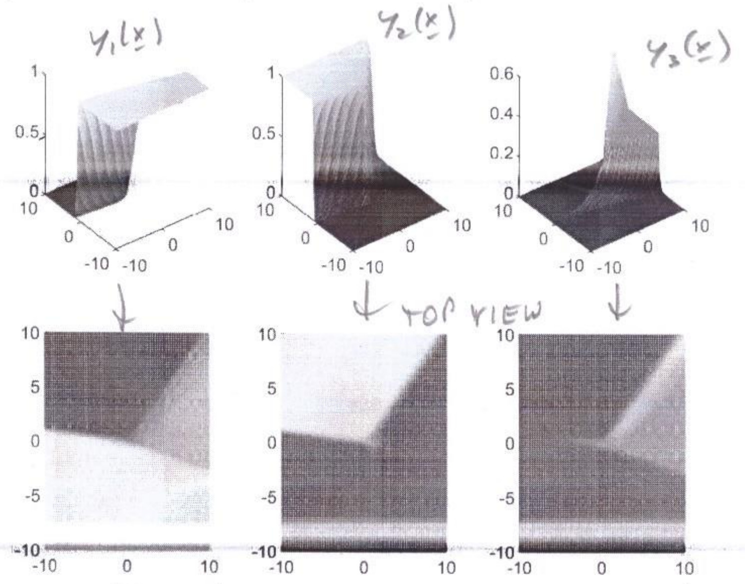
We will see in the following chapters how the RELU's can provide with multiple layers the greatest flexibility in implementing nonlinear functions.

MULTIPLE BINARY CLASSIFIERS

In the architecture of Figure 1(b), the functions $z_1(\underline{x}), z_2(\underline{x}), \dots z_M(\underline{x})$ are thresholded with sigmoidal function.

$$\begin{cases} y_1(\underline{x}) = \varphi(z_1(\underline{x})) \\ \vdots \\ y_M(x) = \varphi(z_M(x)) \end{cases}$$

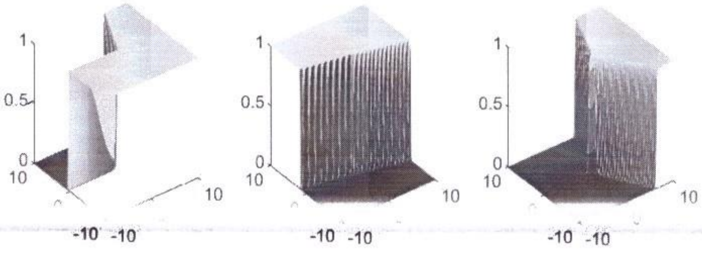
and we have almost-binary functions as shown in the following figure for the three examples of the previous page.

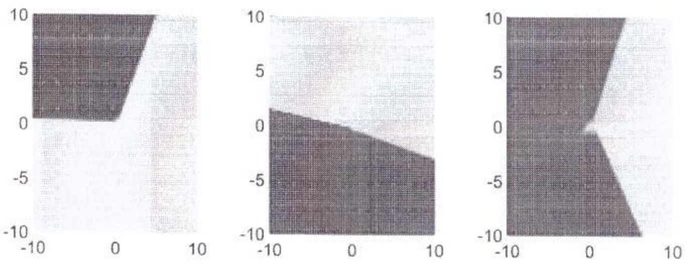


By changing the hyperplane positions in the first layer and the linear combinations in the second layer, we can obtain quite complex partitions of the input space in two regions.

M-ARY CLASSIFICATION

In the architecture of Fig 1(c) the linear combinations of the embedding functions are fed to a softmax for multi-class classification. The difference with the architecture that uses sigmoids on the first layer, is that the log-likelihood functions $z_i(\underline{x}) = \underline{c}_i^T \underline{y}_E(\underline{x}) + b_i$, $i=1,\dots,M$ are rooftop functions. The following picture shows a 3-class classifier for random weights.





Note the partition of the input space in 3 regions.
Rooftop functions are a very flexible class of multidimensional functions to implement quite arbitrarily-shaped log-likelihoods.

[my note: All figures are drawn in MATLAB with two_layer.m]

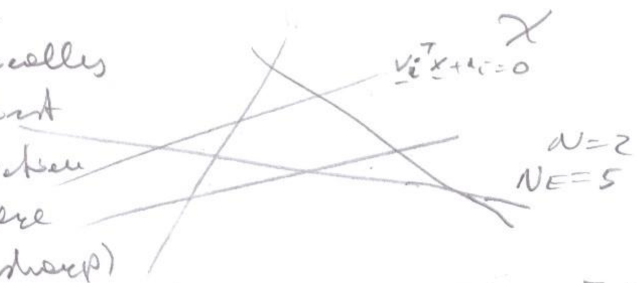
The pictures we have drawn in 2D are quite simple to understand. However, we should be warned that they may not necessary help to see what happens in high dimensions ($N \gg 2$) because N may be in the order of hundreds or even millions. Think of $\underline{x}$ being a vectorized version of an image. In such cases our intuition gained from $N=2$ may be partially elusive

NUMBER OF REGIONS CREATED BY THE FIRST LAYER

We have seen in the previous sections that the basis functions provided by the first layer are either "ridges", when we use sigmoids, or "ramps" when we use RELU functions.
The edges of these regions in the N-dimensional space $\mathcal{X}$ are determined by the N_E hyperplanes

$$\underline{V}_i^T \underline{x} + a_i = 0 \quad i = 1, \dots, N_E$$

Therefore all the linear combination of the embeddings $y_{Ei}(\underline{x})$ will lay on top of the partitions of $\mathcal{X}$ determined by those hyperplanes.
More specifically when the first layer activation functions are sigmoids (sharp)



any linear (affine) function $z_j(\underline{x}) = \underline{C}_j^T y_E(\underline{x}) + b_j$ will be the sum of flat functions in each region (see pictures)
Similarly, when the activation functions are RELUs, any affine function $z_j(\underline{x})$ will be a linear surface (see pictures) in each region.
Note that there is continuity at all the region boundaries.

Therefore the complexity of the linear functions that can be built on top of the first layer, depends on the number of regions that are created by the N_E hyperplanes.
It is possible to count the maximum number of regions that are created by N_E hyperplane in an N-dimensional space.

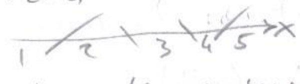
THEOREM

The maximum number of regions formed by N_E hyperplanes in general positions (linearly independent coefficients) in $\mathbb{R}^N$ is

$$G(N,N_E) = \sum_{K=0}^{N} \binom{N_E}{K} . \qquad (20)$$

It is interesting to look at the counting for low numbers.
In onedimension $(N=1)$ $G(1,N_E)=\binom{N_E}{0}+\binom{N_E}{1}=1+N_E$

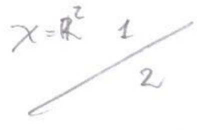
$N_E=4$

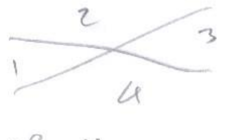


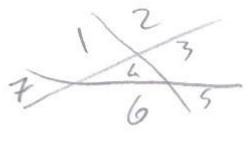
In two dimensions $(N=2)$

$$G(N,N_E)=\binom{N_E}{0}+\binom{N_E}{1}+\binom{N_E}{2}=1+N_E+\frac{N_E(N_E-1)}{2}=\frac{2+N_E+N_E^2}{2}$$

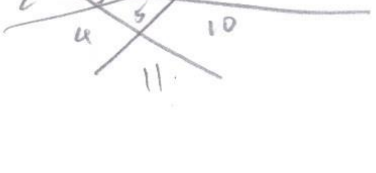
$N_E=1; G=2$ $\quad$ $N_E=2; G=4$ $\quad$ $N_E=3; G=7$







$N_E=4; G=11$



By looking at the expansion

$$G(N,N_E) = \binom{N_E}{0} + \binom{N_E}{1} + \binom{N_E}{2} + \binom{N_E}{3} + \binom{N_E}{4} + \dots + \binom{N_E}{N}$$

$$= 1 + N_E + \frac{N_E(N_E-1)}{2} + \frac{N_E(N_E-1)(N_E-2)}{\underset{\uparrow\ \textcircled{3!}}{6}} + \frac{N_E(N_E-1)(N_E-2)(N_E-3)}{\underset{\nwarrow 4!}{24}}$$

$$+ \dots + \frac{N_E(N_E-1)\dots(N_E-N+1)}{N!}$$

For N_E sufficiently large the dominant term is the last one with the largest exponent. Therefore a rough estimate is

$$G(N,N_E) \simeq \frac{N_E^N}{N!}$$

The growth in N_E is not exponential, but polynomial with exponent N. In any case the complexity of the functions built by the network grows with the number of embedding dimensions justifying the universal approximation property of this architecture predicted by the representation theorems.

THE PROOF OF THE THEOREM

The formula for $G(N,N_E)$ can be easily proven considering the recursion

$$G(N,N_E) = G(N,N_E-1) + G(N-1,N_E-1) \qquad (r1)$$

with initial conditions

$$G(N,0) = 1 \qquad N=1,2,3,\dots \qquad (r2)$$

The recursion is based on the single consideration that adding one hyperplane N_E in N dimensions, produces additional regions $G(N-1, N_E-1)$, which are the maximum intersected regions by this last hyperplane.

Now it is sufficient to show that (r0) satisfies (r1) and (r2). Substituting (r0) in (r1), we get

$$\sum_{K=0}^{N} \binom{N_E}{K} = \sum_{K=0}^{N} \binom{N_E-1}{K} + \sum_{K=1}^{N} \binom{N_E-1}{K-1}$$

From combinatoric analysis we know that we can write

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

therefore

$$\binom{N_E-1}{0} + \sum_{K=1}^{N} \left[\binom{N_E-1}{K} + \binom{N_E-1}{K-1} \right]$$

$$= \quad 1 \quad + \sum_{K=1}^{N} \binom{N_E}{K} = \sum_{K=0}^{N} \binom{N_E}{K}$$