Neural networks

Blanca Cano Camarero

Universidad Autónoma de Madrid

December 16, 2022

Overview

The origins

The term *neural network* has its origins in attempts to find mathematical representations of information processing in biological system.
(McCulloch and Pitts, 1943; Widrow and Hoff, 1960; Rosenblatt, 1962; Rumelhart et al., 1986)

Feed-forward Network Functions

Lineal models:

$$y(x, w) = f\left(\sum_{j=1}^{M} w_j \phi_j(x)\right) \tag{1}$$

where f is a nonlinear activation function in classification and the identity in regression. Our goal is to extend the model by making ϕ depend on parameters and then to allow these parameters to be adjusted along with the coefficients $\{w_i\}$.

Activations

First we construct M linear combinations of the input variables $x_1, \ldots, x_D$ in the form

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2}$$

where $j \in \{1, \ldots, M\}$, and the superscript indicates the layer of the network the parameters are in.

We shall refer to

- The parameters $w_{ji}^{(1)}$ as *weights*.
- The parameters $w_{j0}^{(1)}$ as the *biases*.
- The quantities a_j as *activations*.

Hidden unit and activation functions

Each of the *activations* parameters are transformed using a **differentiable**, nonlinear *activation function* h

$$z_j = h(a_j). \tag{3}$$

We shall refer to

- The quantities z_j as hidden units.

1

[1] Show examples of activations functions.

Discussion: What is the real importance of activation function

- The hypothesis: should't be polynomial.
- ReLu is the most used.
- Kernel method, the kernel not as relevant.
- Balance between precision and computational cost?

Output unit activations

The values (3) are again lineal combined to give *output unit activations*

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{4}$$

where $k \in \{1, \ldots, K\}$, and K is the total number of outputs. This transformation corresponds to the second layer of the network.

Network output

Finally, the output unit activations (4) are transformed using an appropriate activations function to give a set of network output.

- For standard regression problems the activation function is the identity so that $y_k = a_k$.
- For binary classification problems, each output unit activations is transformed using a logistic sigmoid function so that

$$y_k = \sigma(a_k) \tag{5}$$

where

$$\sigma(a) = \frac{1}{1 + exp(-a)}. \tag{6}$$

- For multiclass problems a softmax activation function (4.62)

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)} = \frac{exp(a_k)}{\sum exp(a_j)}. \tag{7}$$

The resulting model

$$y_k(x, w) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{j=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{8}$$

2 3

[2] Comment differences with lineal regression

[3] Write as a matrix and print a graph.

Example of a neural network having a general feed-forwed topology

I have not found any article that create a math formulation.

- Add skip layer connection explicitly.
- Same treat.

Other example recurrent neural
Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network Sherstinsky (2018) (difference equations and PST).

Weight space symmetries

Change the weights and obtain the same results.
How to obtain symmetries?

- Changing the sigh of all nodes and using that *tanh* is odd.
- Permuting hidden nodes.

Bisop: Role comparing with the Bayesian model. For us: There are going to be different solutions.
Some observation:

- Homotopy of the image for me the same function.
- There are any approach that use equivalence class?
- Meng et al. (2018)

The error function

$$t = y(x, w) + \epsilon \tag{9}$$

where $\epsilon \sim \mathcal{N}(0, \beta^{-1})$ and $\beta \in \mathbb{R}$ is the precision ($\beta^{-1} = \sigma^2$).
Thus we can write

$$p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1}). \tag{10}$$

For a single real-valued variable x, the Gaussian distribution is defined by

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}. \tag{11}$$

We have to minimize the following error function

$$-\log p(t|X,w,\beta) = \frac{n\beta}{2}\sum_{i=1}^{n}\{y(x_i,w)-t_i\}^2 - \frac{n}{2}\log\beta - \frac{n}{2}\log(2\pi) \tag{12}$$

which can be used to learn the parameters w and β.

The error function

Let take the error function as the

$$E(w) = \frac{1}{2}\sum_{i=1}^{n}\{y(x_i, w) - t_i\}^2 \tag{13}$$

where we have discarded additive and multiplicative constant.
Some considerations:

- The value of w found by minimizing $E(w)$ will be denoted as w_{ML} (maximum likelihood solution).
- The nonlinearity of the network function $y(x_n, w)$ causes the error $E(w)$ to be nonconvex.
- So in practice W_{ML} would be a local minimum.

About the precision

Having found w_{ML} the value of β can be found by minimizing the negative log likelihood to give

$$\frac{1}{\beta_{ML}} = \frac{1}{N}\frac{1}{2}\sum_{i=1}^{N}\{y(x_i, w_{ML}) - t_i\}^2 \tag{14}$$

Multiple target variable

$$p(t|x, w) = \mathcal{N}(t|y(x, w), \beta^{-1} I). \tag{15}$$

The noise precision is the given by

$$\frac{1}{\beta_{ML}} = \frac{1}{NK} \sum_{i=1}^{N} \|y(x_i, w_{ML}) - t_i\|^2 \tag{16}$$

Minimizing regression case

$$\frac{\partial E}{\partial a_k} = y_k - t_k \tag{17}$$

Binary classification problem

We have a single target t such that $t = 1$ dentotes class C_1 and $t = 0$ denotes class C_2. As we see last week we consider a single output whose activation function is a logistic sigmoid:

$$y = \frac{1}{1 + exp(-a)} \tag{18}$$

so that

$$0 \leq y(x, w) \leq 1. \tag{19}$$

We can interpret $y(x, w)$ as the condicional probability $p(C_1|x)$.

The cross entropy error function

The conditional distribution of targets given inputs is the Bernoulli distribution of the form

$$p(t|x, w) = y(x, w)^t \{1 - y(x, w)\}^{1-t}. \tag{20}$$

If we consider a training set of independent observation, then the error function which is given by the negative log likelihood, is then a cross entropy error function of the form

$$E(w) = -\sum_{n=1}^{N} \{t_n \log y_n + (1 - t_n) \log(1 - y_n)\} \tag{21}$$

Using cross entropy error function instead of the sum of squares for classification problem leads to faster training as well as improved generalization.

K binary classification

$$p(t|x,w) = \prod_{k=1}^{K} y(x,w)^t \left\{1 - y(x,w)\right\}^{1-t}. \tag{22}$$

$$E(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K} \left\{t_{nk} \log y_{nk} + (1 - t_{nk}) \log(1 - y_{nk})\right\} \tag{23}$$

1 of K coding scheme

The network outputs are interpreted as

$$y_k(x, w) = p(t_k = 1|x), \tag{24}$$

leading to the following error function

$$E(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \log y_k(x_n, w). \tag{25}$$

Universal approximator

This paper rigorously establishes that standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators. Hornik et al. (1989)

Backpropagation

$$h_{t+1} = h_t - \eta \nabla E(h_t). \tag{26}$$

Properties

- Local optimization.
- Fix the number of neurons first.
- η is a hyperparameter.

Our model

$$h_k(x) = \sum_{j=1}^{n} \beta_{jk} \sigma \left(\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right) \tag{27}$$

Derivative with respect to β_{vw} st $v \in \{1, \ldots, n\}$ and $w \in \{1, \ldots, s\}$

$$\frac{\partial E(h)}{\partial \beta_{vw}} = \frac{\partial}{\partial \beta_{vw}} \left[\frac{1}{2} \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y_k)^2 \right] \tag{28}$$

$$= \frac{1}{2} \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} 2 \left(h_k(x) - y_k\right) \frac{\partial h_k(x)}{\partial \beta_{vw}} \tag{29}$$

$$= \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y_k) \frac{\partial}{\partial \beta_{vw}} \left[\sum_{j=1}^{n} \beta_{jk} \sigma \left(\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right) \right] \tag{30}$$

$$= \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y_k) \left(\sum_{j=1}^{n} \frac{\partial}{\partial \beta_{vw}} \left[\beta_{jk} \sigma \left(\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right) \right] \right) \tag{31}$$

$$= \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y_k) \left(\sum_{j=1}^{n} \chi_{[j=v \wedge k=w]} \sigma \left(\alpha_{0j} + \sum_{j=1}^{d} \alpha_{ij} x_i \right) \right) \tag{32}$$

$$= \sum_{(x,y) \in \mathcal{D}} (h_w(x) - y_w) \left(\sigma \left(\alpha_{0v} + \sum_{i=1}^{d} \alpha_{iv} x_i \right) \right). \tag{33}$$

Derivative with respect to α_{0v} st $v \in \{1, \ldots, n\}$

$$\frac{\partial E(h)}{\partial \alpha_{0v}} = \frac{\partial}{\partial \alpha_{0v}} \left[\frac{1}{2} \sum_{(x,y)\in\mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y)^2 \right] \tag{34}$$

$$= \frac{1}{2} \sum_{(x,y)\in\mathcal{D}} \sum_{k=1}^{s} 2\,(h_k(x) - y_k)\, \frac{\partial h_k(x)}{\partial \alpha_{0v}} \tag{35}$$

$$= \sum_{(x,y)\in\mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y_k)\, \frac{\partial}{\partial \alpha_{0v}} \left[\sum_{j=1}^{n} \beta_{jk}\, \sigma\left(\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right) \right] \tag{36}$$

$$= \sum_{(x,y)\in\mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y_k) \left(\sum_{j=1}^{n} \beta_{jk}\, \frac{\partial}{\partial \alpha_{0v}} \left[\sigma\left(\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right) \right] \right) \tag{37}$$

$$= \sum_{(x,y)\in\mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y_k) \left(\sum_{j=1}^{n} \beta_{jk}\, \sigma'\left(\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right) \frac{\partial}{\partial \alpha_{0v}} \left[\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right] \right) \tag{38}$$

$$= \sum_{(x,y)\in\mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y_k) \left(\sum_{j=1}^{n} \beta_{jk}\, \sigma'\left(\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right) \chi_{[j=v]} \right) \tag{39}$$

$$= \sum_{(x,y)\in\mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y_k) \left(\beta_{vk}\, \sigma'\left(\alpha_{0v} + \sum_{i=1}^{d} \alpha_{iv} x_i \right) \right). \tag{40}$$

Derivative with respect α_{uv} st $u \in \{1, \ldots, d\}$ and $v \in \{1, \ldots, n\}$

$$\frac{\partial E(h)}{\partial \alpha_{uv}} = \frac{\partial}{\partial \alpha_{uv}} \left[\frac{1}{2} \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} \left(h_k(x) - y_k \right)^2 \right] \tag{41}$$

$$= \frac{1}{2} \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} 2 \left(h_k(x) - y_k \right) \frac{\partial h_k(x)}{\partial \alpha_{uv}} \tag{42}$$

$$= \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} \left(h_k(x) - y_k \right) \frac{\partial}{\partial \alpha_{uv}} \left[\sum_{j=1}^{n} \beta_{jk} \sigma \left(\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right) \right] \tag{43}$$

$$= \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} \left(h_k(x) - y_k \right) \left(\sum_{j=1}^{n} \beta_{jk} \frac{\partial}{\partial \alpha_{uv}} \left[\sigma \left(\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right) \right] \right) \tag{44}$$

$$= \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} \left(h_k(x) - y_k \right) \left(\sum_{j=1}^{n} \beta_{jk} \sigma' \left(\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right) \frac{\partial}{\partial \alpha_{uv}} \left[\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right] \right) \tag{45}$$

$$= \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} \left(h_k(x) - y_k \right) \left(\sum_{j=1}^{n} \beta_{jk} \sigma' \left(\alpha_{0j} + \sum_{i=1}^{d} \alpha_{ij} x_i \right) \chi_{[i=u \wedge j=v]} x_i \right) \tag{46}$$

$$= \sum_{(x,y) \in \mathcal{D}} \sum_{k=1}^{s} \left(h_k(x) - y_k \right) \left(\beta_{vk} \sigma' \left(\alpha_{0v} + \sum_{i=1}^{d} \alpha_{iv} x_i \right) x_u \right). \tag{47}$$

Summary

$$\frac{\partial E(h)}{\partial \beta_{vw}} = \sum_{(x,y)\in\mathcal{D}} (h_w(x) - y_w) \left(\sigma \left(\alpha_{0v} + \sum_{i=1}^{d} \alpha_{iv} x_i \right) \right). \tag{48}$$

$$\frac{\partial E(h)}{\partial \alpha_{0v}} = \sum_{(x,y)\in\mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y_k) \left(\beta_{vk} \sigma' \left(\alpha_{0v} + \sum_{i=1}^{d} \alpha_{iv} x_i \right) \right). \tag{49}$$

$$\frac{\partial E(h)}{\partial \alpha_{uv}} = \sum_{(x,y)\in\mathcal{D}} \sum_{k=1}^{s} (h_k(x) - y_k) \left(\beta_{vk} \sigma' \left(\alpha_{0v} + \sum_{i=1}^{d} \alpha_{iv} x_i \right) x_u \right). \tag{50}$$

Overview

Often **neural networks have to many weights** and will overfit the data at the global minimum.[4]

- Early stopping.
- Weight decay.
- Weight elimination.

Bubeck and Sellke (2021) (over-parameterized regime where the parameters of the model exceed the size of the training dataset.)

[4] Hastie et al. (2001) page 398

Weight decay

Analogous to ridge regression used for linear models.
We add a penalty to the error function $R(\theta) + \lambda J(\theta)$, where

$$J(\theta) = \sum_{km} \beta_{km}^2 + \sum_{ml} \alpha_{ml}^2 \tag{51}$$

and $\lambda \geq 0$ is a tuning parameter.
Larger values of λ will tend to shrink the weights toward zero.
The effet to the penalty is to simply add terms $2\beta_{km}$ and $2\alpha_{ml}$ to the respective gradient expressions.

weight eliminaition penalty

(Similar to lasso)

$$J(\theta) = \sum_{km} \frac{\beta_{km}^2}{1+\beta_{km}^2} + \sum_{ml} \frac{\alpha_{ml}^2}{1+\alpha_{ml}^2}, \tag{52}$$

this has te effect of shrinking smaller weights more.

Near zero

- If the weights are near zero, then the operative part of the sigmoid is roughly linear, and hence the neural network collapse into an approximately linear model.
- Become nonlinear as the weights increase.
- The exact zero weights leads to zero derivatives and perfect symmetry and the algorithm never moves.
- Starting instead with large weights often lead to poor solutions.

Transfer learning

Definition

Given some/an observations corresponding to $m^s \in \mathbb{N}^+$ source domains and task and some observations about $m^T \in \mathbb{N}^T$. Transfer learning utilizes the knowledge implied in the source domains to improve the performance of the learned decision function on the target domains.

Zhuang et al. (2019)
Aji et al. (2020)

My idea

Main idea:

- Fix the network architecture.
- Select a dataframe subset randomly.
- Build the network from this subset to work perfectly.

Some notation

$$h : \mathbb{R}^d \longrightarrow \mathbb{R}^s, \tag{53}$$

$$h_k(x) = \sum_{i=1}^{n} \left(\beta_{ik} \gamma \left(\sum_{j=1}^{d} (\alpha_{ij} x_j) + \alpha_{0j} \right) \right). \tag{54}$$

determined by its params:

$$(A, B, S) \in R^{n \times s} \times R^{d \times n} \times R^d. \tag{55}$$

$A = (\alpha_{ij})$ con $i \in \{1, \dots d\}$, $j \in \{1, \dots n\}$.
$S = (\alpha_{0j})$ con $j \in \{1, \dots, n\}$.
$B = (\beta_{jk})$ con $j \in \{1, \dots n\}$, $k \in \{1, \dots s\}$.

Algorithm description: 1. Values initialization

Let be $h \in \mathcal{H}(\mathbb{R}^d, \mathbb{R}^s)$ with n hidden units. and let $M \in R^+$ chosen

1. Take randomly $p \in \mathbb{R}^{d+1} \setminus \{0\}$.
2. Let Λ be an empty set.
3. While $|\Lambda| < n$ repeat:
   1. Pick randomly $(x, y) \in \mathcal{D}$.
   2. If x satisfies that for every $(z, w) \in \Lambda$

$$p \cdot (x - z) \neq 0, \tag{56}$$

   then let $\Lambda \leftarrow \Lambda \cup \{(x, y)\}$.

Order de set

Without loss of generality the elements of Λ

$$\Lambda = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$$

are ordered by the following statement

$$p \cdot x_1 < p \cdot x_2 < \ldots p \cdot x_n. \tag{57}$$

Solve equation

Pick $(x_1, y_1) \in \Lambda$

$$
\begin{aligned}
S_1 &= Mp_0, \\
A_{1*} &= Mp_{[1,d]}, \\
B_{*1} &= y_1.
\end{aligned}
$$

For $k \in \{1, \ldots, n\}$

$$
\begin{aligned}
S_k &= M - \frac{2M}{p \cdot (x_k - x_{k-1})}(p \cdot x_k), \\
A_{ki} &= \frac{2M}{p \cdot (x_k - x_{k-1})} p_i \quad i \in \{1, \ldots d\}, \\
B_{*k} &= y_k - y_{k-1}.
\end{aligned}
$$

where $(x_k, y_k) \in \Lambda$.
(A, S, B) are the matrix we searched for.

Some problems

- If the architecture *is too big* them overfitting.
- More??

Activation function selection. Theorem

See neural network as a functional spaces

Sea $\phi \in \mathcal{A}(\mathbb{R}^2)$ una función afín cuya forma matricial asociada es de la forma:

$$\phi((x,y)) = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x & t_y \end{bmatrix} \tag{58}$$

con $a, b \in \mathbb{R}^*$ y $t_x, t_y \in \mathbb{R}$.

Sean dos funciones de activación σ, γ tales que

$$\phi(Grafo(\sigma)) = Grafo(\gamma),$$

entonces el espacio de redes neuronales de n neuronas creado a partir de la función de activación σ es igual al espacio de redes neuronales creado a partir la función de activación γ.

Proof part 1

Sea $\mathcal{H}^{+}_{\sigma,n}(\mathbb{R}^d, \mathbb{R}^s)$ el espacio de redes neuronales con n neuronas con sesgo.
Está claro que $\mathcal{H}^{+}_{\gamma,n}(\mathbb{R}^d, \mathbb{R}^s)$ y $\mathcal{H}^{+}_{\sigma,n}(\mathbb{R}^d, \mathbb{R}^s)$ son biyectivos.
Ya que basta con tomar una red neuronal de una y cambiarle la función de activación por la de la otra. Veamos que se da la igualdad viendo que una está contenida en la otra.
Para cualquier $h \in \mathcal{H}^{+}_{\sigma,n}(\mathbb{R}^d, \mathbb{R}^s)$ la proyección i-ésima de h será de la forma

$$h_i(x) = \sum_{j=1}^{n} (\beta_j \sigma(A_j(x)) + k_j),$$

con $x \in \mathbb{R}^d, \beta_j, k_j \in \mathbb{R}, A_j \in \mathcal{A}(\mathbb{R})$.

Proof part 2

Procedemos a definir $\tilde{h}_i(x)$ como sigue

$$\tilde{h}_i(x) = \sum_{j=1}^{n}(\beta_j(b\sigma(aA_j(x) + t_x) + t_y) + k_j) = \sum_{j=1}^{n}(\tilde{\beta}_j\sigma(\tilde{A}_j(x)) + \tilde{k}_j), \tag{59}$$

con $x \in \mathbb{R}^d, \tilde{\beta}_j, \tilde{k}_j \in \mathbb{R}, \tilde{A}_j \in \mathcal{A}(\mathbb{R})$, por lo que está claro que $\tilde{h}(x) \in \mathcal{H}^+_{\sigma,n}(\mathbb{R}^d, \mathbb{R}^s)$. Observemos que la hipótesis del enunciado establece que

$$\begin{aligned} Grafo(\gamma) &= \{(x, \gamma(x)) \colon x \in \mathbb{R}\} && (60)\\ &= Grafo(\gamma) = \phi(Grafo(\sigma)) && (61)\\ &= \phi(\{(x, \sigma(x)) \colon x \in \mathbb{R}\}) && (62)\\ &= \{(ax + t_x, b\sigma(x) + t_y) \colon x \in \mathbb{R}\}. && (63) \end{aligned}$$

Proof part 3

Por lo que $\tilde{h}$ así definida es a su vez

$$\tilde{h}_i(x) = \sum_{j=1}^{n} (\beta_j \gamma A_j(x) + k_j) \tag{64}$$

es decir que $\tilde{h}(x) \in \mathcal{H}^+_{\gamma,n}(\mathbb{R}^d, \mathbb{R}^s)$. Así que vía ϕ se ha definido una inyección de $\mathcal{H}^+_{\sigma,n}(\mathbb{R}^d, \mathbb{R}^s)$ a $\mathcal{H}^+_{\gamma,n}(\mathbb{R}^d, \mathbb{R}^s)$, esto es

$$\mathcal{H}^+_{\sigma,n}(\mathbb{R}^d, \mathbb{R}^s) \subseteq \mathcal{H}^+_{\gamma,n}(\mathbb{R}^d, \mathbb{R}^s). \tag{65}$$

Proof part 4

Además, ϕ con las hipótesis exigidas es invertible, con inversa:

$$\phi^{-1}((x,y)) = \begin{bmatrix} \frac{1}{a} & 0 \\ 0 & \frac{1}{b} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} -\frac{t_x}{a} & -\frac{t_y}{b} \end{bmatrix}. \tag{66}$$

Proof part 5

Así que razonando de igual manera que en el apartado anterior se tiene la inclusión

$$\mathcal{H}^{+}_{\sigma,n}(\mathbb{R}^d, \mathbb{R}^s) \supseteq \mathcal{H}^{+}_{\gamma,n}(\mathbb{R}^d, \mathbb{R}^s), \tag{67}$$

por lo que podemos concluir que

$$\mathcal{H}^{+}_{\gamma,n}(\mathbb{R}^d, \mathbb{R}^s) = \mathcal{H}^{+}_{\sigma,n}(\mathbb{R}^d, \mathbb{R}^s).$$

Proof part 6

$$\mathcal{H}^+_{\sigma,n}(\mathbb{R}^d, \mathbb{R}^s) = \mathcal{H}^+_{\gamma,n}(\mathbb{R}^d, \mathbb{R}^s) \subset \mathcal{H}_{\gamma,n+1}(\mathbb{R}^d, \mathbb{R}^s) \subset \mathcal{H}^+_{\gamma,n+1}(\mathbb{R}^d, \mathbb{R}^s) = \mathcal{H}^+_{\sigma,n+1}(\mathbb{R}^d, \mathbb{R}^s).$$

Por lo que para un n arbitrariamente grande, se acaba de probar lo buscado.

$$\mathcal{H}_{\gamma}(\mathbb{R}^d, \mathbb{R}^s) = \mathcal{H}_{\sigma}(\mathbb{R}^d, \mathbb{R}^s).$$

Why is util

Choose the one with lest computational cost and potential solutions would be the same.

References I

Alham Fikri Aji, Nikolay Bogoychev, Kenneth Heafield, and Rico Sennrich. In neural machine translation, what does transfer learning transfer? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7701–7710, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.688. URL https://aclanthology.org/2020.acl-main.688.

Sebastien Bubeck and Mark Sellke. A universal law of robustness via isoperimetry. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=z710SKqTFh7.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

References II

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

Qi Meng, Wei Chen, Shuxin Zheng, Qiwei Ye, and Tie-Yan Liu. Optimizing neural networks in the equivalent class space. 02 2018.

Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314, 2018. URL http://arxiv.org/abs/1808.03314.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *CoRR*, abs/1911.02685, 2019. URL http://arxiv.org/abs/1911.02685.