Introduction to Recurrent Neural Networks

using python.

Juan Contreras López
Iván Andrés Trujilllo Abella

Facultad de Ingenieria
Pontificia Universidad Javeriana

Background classification problem

Fisher(1936) proposes the linear Discriminant, the problem consist in predict a binary outcome according to a set of features, for instance find the variables that could predict the bankruptcy.
The main objective is construct a $z = \mathrm{w}^{\mathrm{t}}\mathrm{x}$ score whose indicate the probability of belonging to a class.

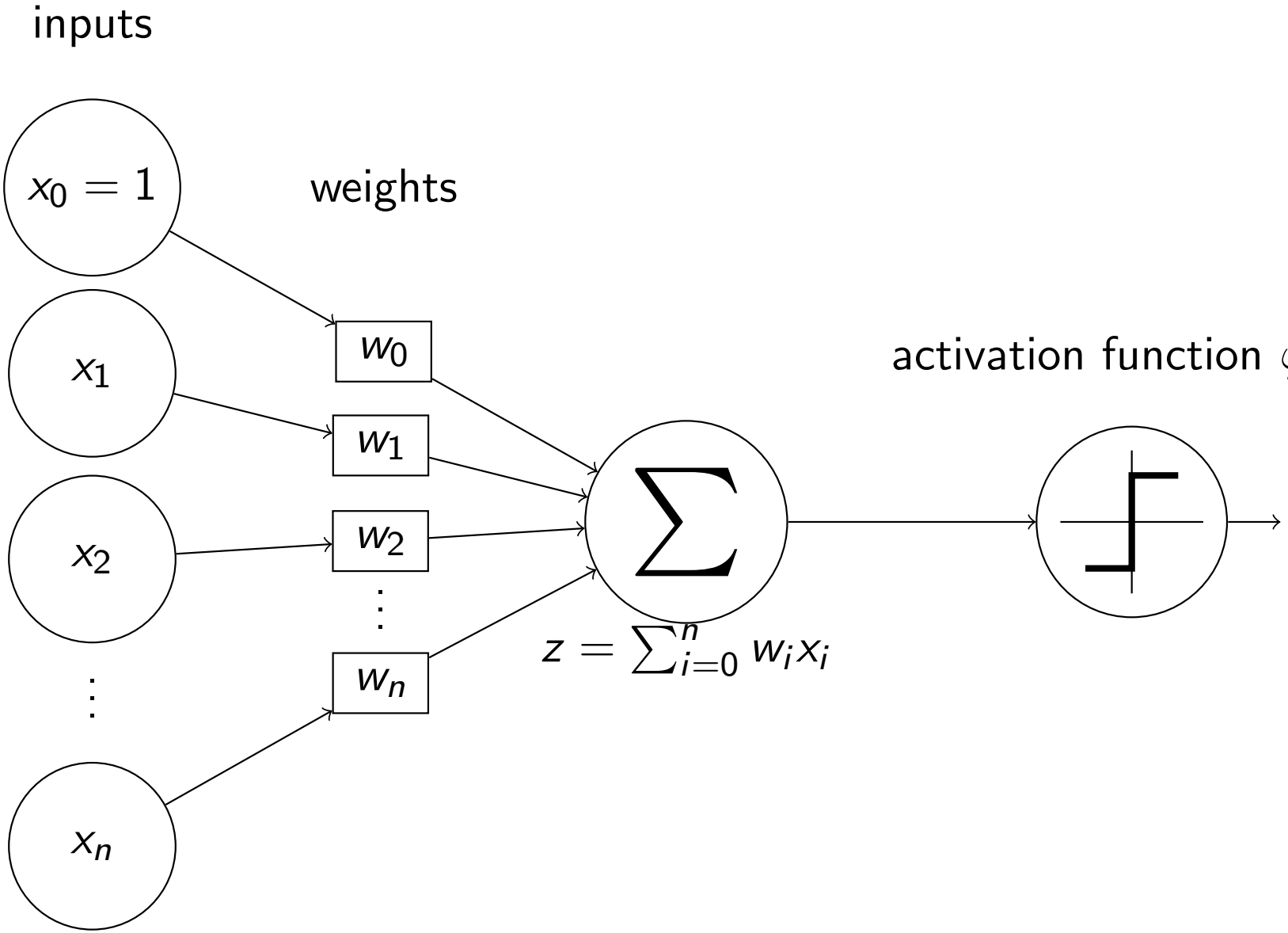
Binary classification problem

$y_0, y_1 \in Y$ and $y_0 \cup y_1 = Y$ and $y_0 \cap y_1 = \emptyset$ both class are exhaustive and are defined without ambiguity. A vector of weights and a vector of features for a

$$w^t x = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} = \sum_{i=1}^{n} w_i x_i = z_i$$

Perceptron

Rossenblant(1958) We can define a vector input x and a vector of weights w and a activation function φ that take as input the inner product of both vectors defined previously $\varphi(w^t x)$.


inputs
$x_0 = 1$
weights
x_1
x_2
x_n
w_0
w_1
w_2
w_n
$\sum$
$z = \sum_{i=0}^{n} w_i x_i$
activation function $\varphi(z)$

Activation function

$\varphi()$ could be defined as the sigmoid function.

$$\begin{aligned} p(y=1) &= \varphi(z) \\ p(y=0) &= 1-\varphi(z) \end{aligned} \tag{1}$$

Percepton works with a step function:

$$\varphi(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$$

Returning a binary outcome.

What set of w values we must choose

Cost function

The cost function could be defined in a soft or a hard way.

$$J(w)_{hard} = \sum_{i=1}^{n} = \max(-y_i \hat{y}, 0) \tag{2}$$

J only count the number of mismatches. However this function not is differentiable.

$$J(w)_{soft} = \sum_{i=1}^{n} = \max(-y_i z_i, 0) \tag{3}$$

if $y_i z_i < 0$ then lost function > 0
if $y_i z_i > 0$ the lost function $= 0$.

Insights

The update of weights is according to the data bias or mistakes, however when the model match to the class then

$$\Delta w_i = (y_i - \hat{y}_i) = 0 \tag{4}$$

where y_i is the real observed data, and $\hat{y}_i$ is the predicted class. when the $y_i = -1$ and $\hat{y} = 1$ then $\Delta w = -2$, in otherwise $y_i = 1$ and $\hat{y}_i = -1$ then $\Delta w = 2$. in summary:

$$\Delta w_i = \begin{cases} 0 & y_i = \hat{y}_i \\ -2 & y_i < \hat{y}_i \\ 2 & y_i > \hat{y}_i \end{cases}$$

insights

Then when there are mistakes

$$\varphi(w_{i+1}^{t}\mathrm{x_i}) = \varphi((w_i + \Delta w_i)^{t}\mathrm{x_i}) = y_i \tag{5}$$

This mean that weights for the vector of features of the sample i are update to predict the correct class.

$$w_{i+1} = w_i + \eta \Delta w_i \mathrm{x_i} \tag{6}$$

Perceptron algorithm

```
initialize w:
for each x in sample :
estimate y(x)
w = w + update(w)
```

descent gradient

An algorithm to optimization. We can think in a greedy algorithm, searching points, for instance to fit a line to a data points, $e(\beta_0^v, \beta_1^f)$ then we have an associeted error fo β_0 $e_{\beta_0^v} = \sum_{i=1}^{n}(y_i - \beta_0^v + \beta_1^f x_i)^2$.

Gradient descendent

To talk about more deeply about gradient we need talk about exploding gradient problem
Assume the following convex function :

$$ax^2 + bx + c \tag{7}$$

for practical examples and guarantee a minimum global point we said that $a = 2, b = -3, c = 5$.

$$\begin{aligned} f^1 &= 2ax + b = 0 \\ x^* &= \frac{-b}{2a} \end{aligned} \tag{8}$$

Therefore the minimum point is obtained in $x^* = 0.3$.

Gradient descent

```
import numpy as np
import matplotlib.pyplot as plt
def quadratic(a,b,c,x ):
  return a*x**2 + b*x + c
def Dxquadratic(a,b,x):
  return 2*a*x + b
def Dxxquadratic(a,x):
  return a*x
def dx0quadratic(a,b):
  return (-b)/(2*a) # take in mind the left ritgh precedence
def evaldx0(a,b,c):
  point = dx0quadratic(a,b)
  return quadratic(a,b,c,point)
```

Gradient descent

```
def GDS_Quadratic( x0, learning_rate=0.01, iterations_max=100,
    error_max = 0.00001, a=2,b=-3):
  gradient = Dxquadratic # We defined previously
  xi = x0
  iters = 0
  error = 100
  while (iters < iterations_max) and (error > error_max):
    xj = xi - learning_rate * gradient(a,b, xi)
    error = abs(xi-xj)
    xi = xj
    iters += 1
  return xj,iters
GDS_Quadratic(100, learning_rate=0.3)
```

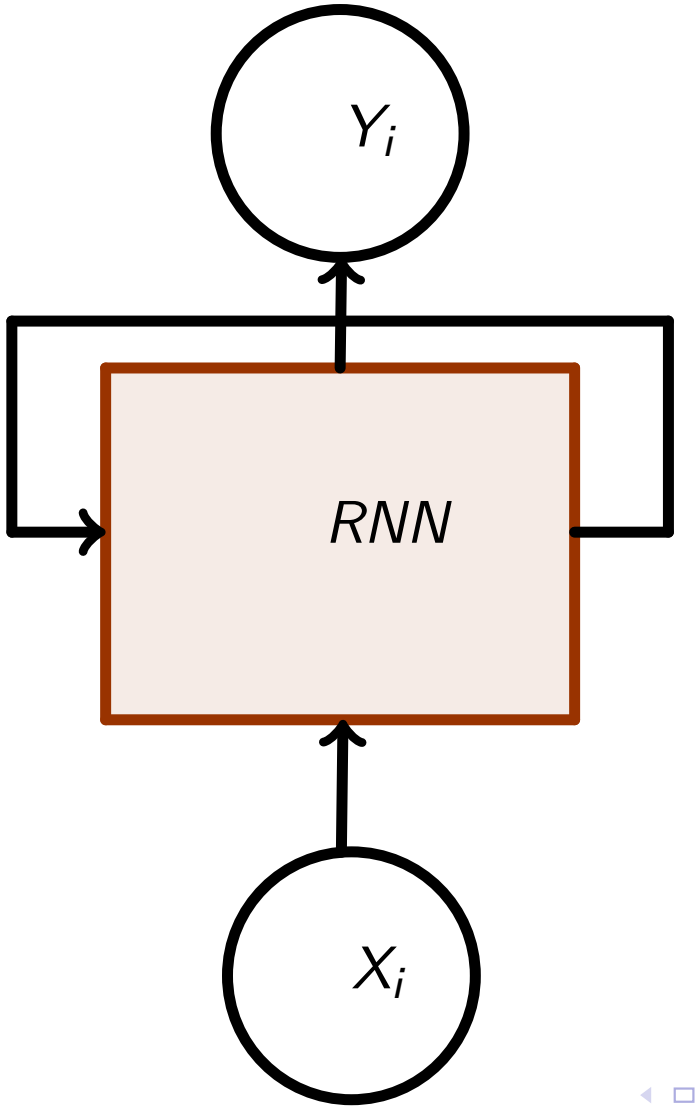
Recurrent Neural Networks

used in speech recognition, due the ability of learn of sequence data.

- series data
- Words or text
- speech recognition

RNN simple

illustration



Different architectures according to the size of input-output

Application

Sentiment analysis only produce a integer output whereas the inputs could be different. Another example is machine translation.

- one to one
- one to many
- many to one
- many to many

Mathematical notation

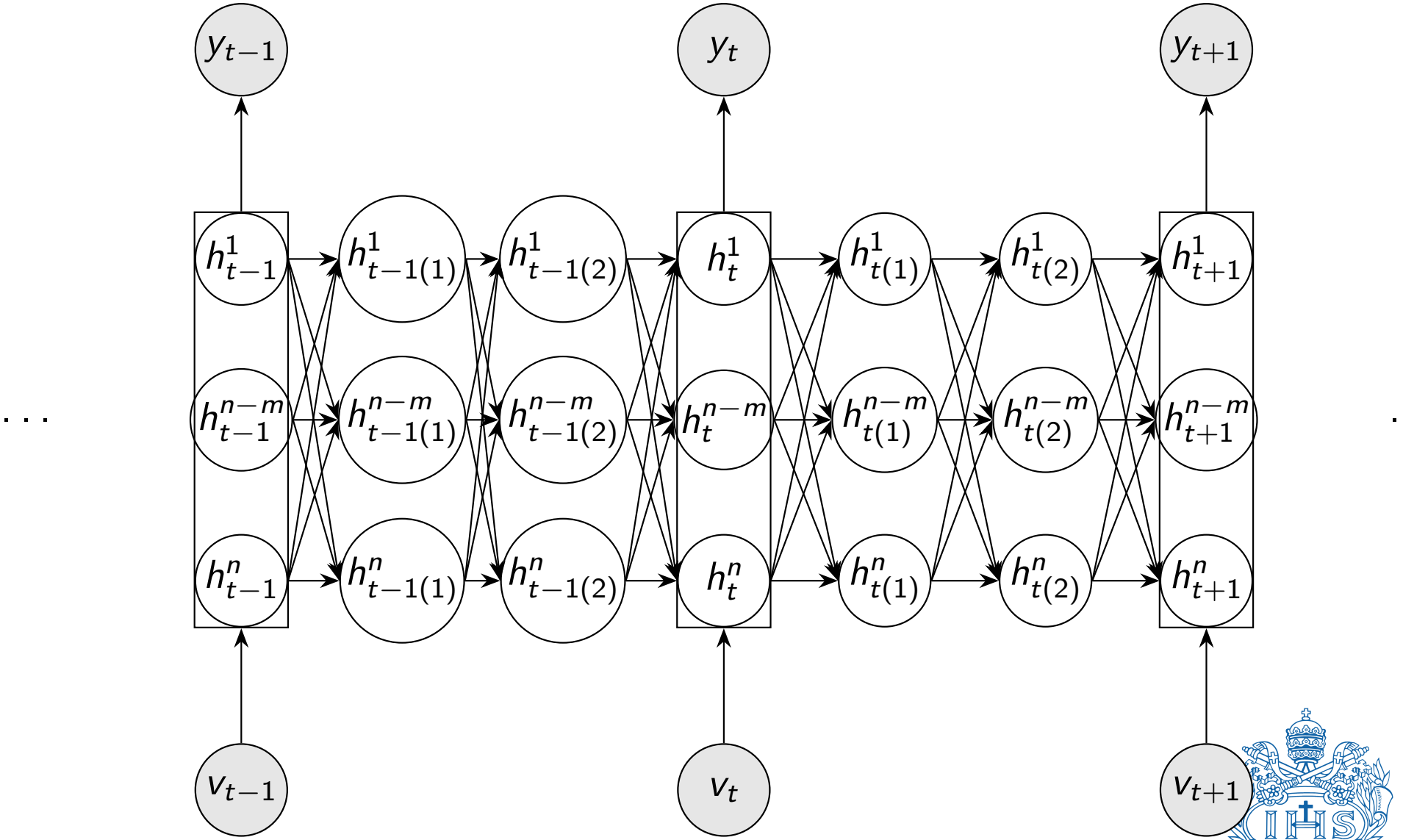
$$
\begin{aligned}
a_t &= f(W_{aa} * a_{t-1} + w_{ax} Xt + ba) \\
y_t &= g(w_{ya} a_t + b_y)
\end{aligned}
\tag{9}
$$

Problems with standard neural networks

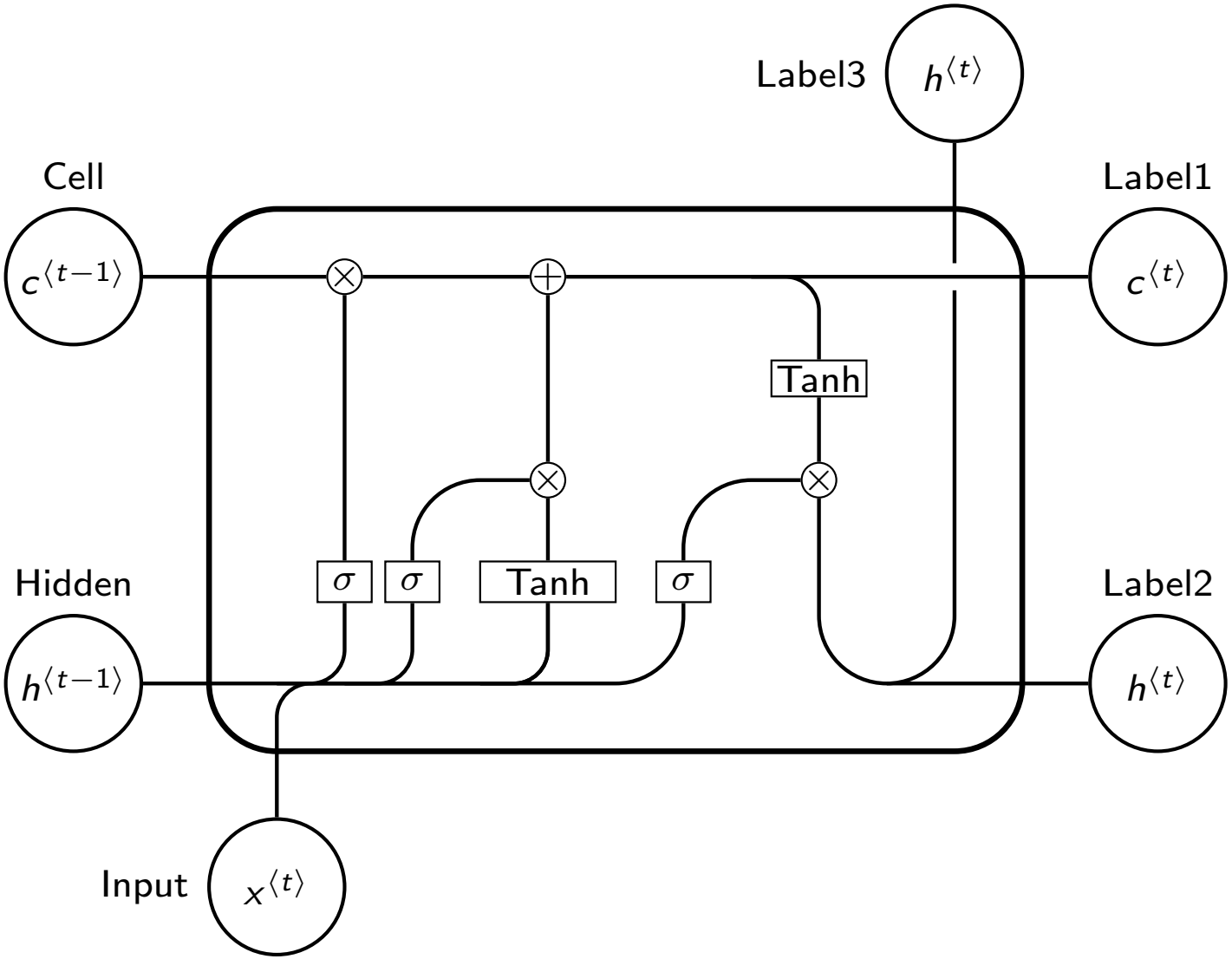
a problem of RNN is that not uses information of posteior inputs.

RNN drawbacks

Not capturing long run dependencies.However, LSTM tackle this pobrlem and are the most used today. this NN added to the recurrent process, the possibility of keep, add or delete information with a cell, regarding with a simple recurrent neural network the cell of LSTM added a input a output more.



Pontificia Universidad JAVERIANA Bogotá


Label3
$h^{\langle t \rangle}$
Cell
$c^{\langle t-1 \rangle}$
Label1
$c^{\langle t \rangle}$
Tanh
σ
σ
Tanh
σ
Hidden
$h^{\langle t-1 \rangle}$
Label2
$h^{\langle t \rangle}$
Input
$x^{\langle t \rangle}$

Recurrent Neural Networks in text

named entity $X^{(i)<t>}$ the $i-th$ training observation, $T_x^{(i)}$ is the length of input.
one-hot codification to vectors, then we can get the following

x: **in the input appear the** $word_1$ **first after** $word_2$ **and finally** $word_3$**.**

$$Dictionary = \begin{pmatrix} Word_1 \\ word_2 \\ \vdots \\ word_n \end{pmatrix}$$

We can follow the $T_x^{(i)} = 9$.

Encoding the sentence

$|x| = 9$ therefore must be $x^1, x^2, ..., x^9$ encoded vectors. indexing in one-hot sense x $|Dictonary| = n$ for this case, then we have that:

$$x_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, x2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, x_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} .....$$

remember that $|x_i| = |Dictionary|$.

Word embedding

Performance in time series prediction

it also evaluated with some metrics, such as

$$MSE = \frac{1}{n}\sum(\hat{y} - y)^2$$

$$RMSE = \sqrt{MSE}$$

$$MAE = \frac{1}{n}\sum|\hat{y} - y|$$

Forecasting day-ahead electricity prices: A comparison of time series and neural network models taking external regressors into account.

Malte Lehna, Fabian Scheller, Helmut Herwartz (2022).

With H-index 152, is ranked Q1 in econometrics and energy.

Objective

Forecasting day-ahead electricity prices: A comparison of time series and neural network models taking external regressors into account.

Compare the performance of four techniques to forecasting german day-ahead electricity price.

DATA

Forecasting day-ahead electricity prices: A comparison of time series and neural network models taking external regressors into account.

hourly data for twelve mobths from October 2017 to September 2018.

Methodology

Forecasting day-ahead electricity prices: A comparison of time series and neural network models taking external regressors into account.

Compare the performance across the most common metrics in time series prediction, for the models:

- SARIMAX
- LSTM
- CNN-LSTM
- VAR

the test period is for three horizons: 1 day, three days, thirty days. We will present the results only for both: LSTM and VAR.

mirror logarithmic

Results

Forecasting day-ahead electricity prices: A comparison of time series and neural network models taking external regressors into account.

LSTM model have a better performance on average, the two-stage VAR is better for shorter prediction horizons.

Time series or RNN-LSTM

VAR model is a classic model, that have a good performance in terms of meaning and significance statistics.