Recurrent Networks

Next token prediction with Vanilla RNNs

Prof. Simone Calderara
Lecturer: **Pietro Buzzega**

Agenda

8.1 Tokenization

8.2 Vanilla RNN

8.1
Tokenization

Tokenization

Points are **vectors** of numeric values. Images are **matrices** of numeric values.

Today we will tackle a **Neural Language Processing** problem. What are **words**?

Tokenization

Points are **vectors** of numeric values. Images are **matrices** of numeric values.

Today we will tackle a **Neural Language Processing** problem. What are **words**?

Words could be expressed as lists of characters, but what are **characters**?

Tokenization

Points are **vectors** of numeric values. Images are **matrices** of numeric values.

Today we will tackle a **Neural Language Processing** problem. What are **words**?

Words could be expressed as lists of characters, but what are **characters**?

Characters are **items** with an **intrinsic meaning**. We cannot naturally represent them in a vector of features such as numbers.

We could also use diphthongs or triphthongs as basics blocks to create our text data.

In NLP, we will name the basic item of our sequence a **token**.

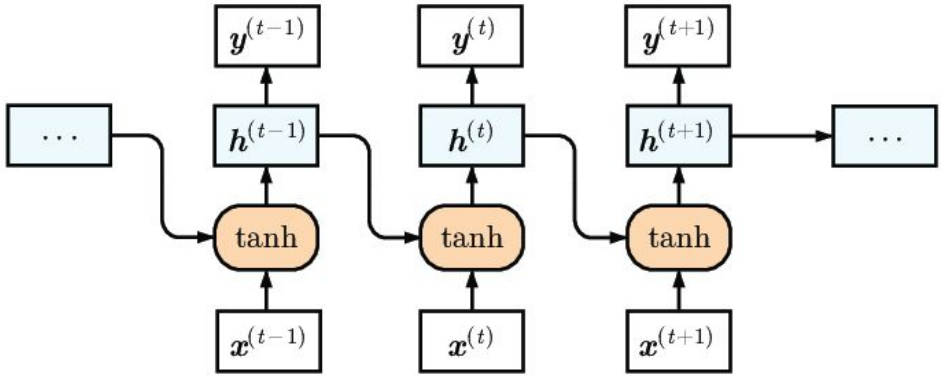
8.2

Vanilla RNN

Vanilla RNN

Recurrent Neural Networks have a hidden state that retains temporal information. At every timestep, our RNN cell encodes additional information into it.

We can use the hidden state to predict the next token (or more than one) or classify the sequence: basically, it can be seen as a **matrix of temporal features**.



Vanilla recurrent cell:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$