Deep Learning and its use in Natural Language Recognition

Aicha KARITE

Engineering Faculty, Euromed University of Fez, Fez Morocco
Fundacion Deusto, University of Deusto, 48007 Bilbao, Basque, Spain

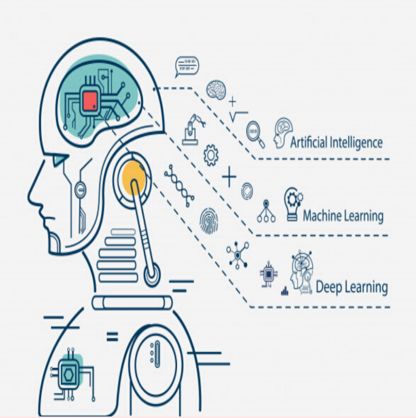
University of Deusto, Bilbao

06 Février 2020

Plan

Deep Learning

Deep Learning



- **Artificial Intelligence :** is a technique that enables a machine to mimic human behavior.
- **Machine Learning :** is a technique to achieve Artificial Intelligence through algorithms trained with data.
- **Deep Learning :** is a type of Machine Learning inspired by the structure of the human brain."The analogy to deep learning is that the rocket engine is the deep learning models and the fuel is the huge amounts of data we can feed to these algorithms."

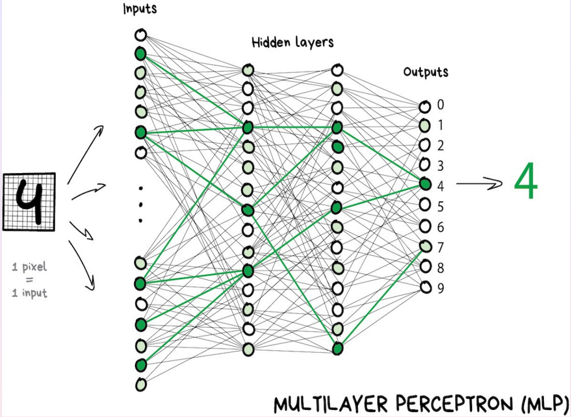


FIGURE – Graphical representation of a neural network of depth $L = 3$. The input layer has $d = 784$ nodes (one for each component of a data point $X_i \in \mathbb{R}^d$), the output layer has $m = 9$ nodes ; the width of the hidden layers equals 15.

Computation of the aggregation z :

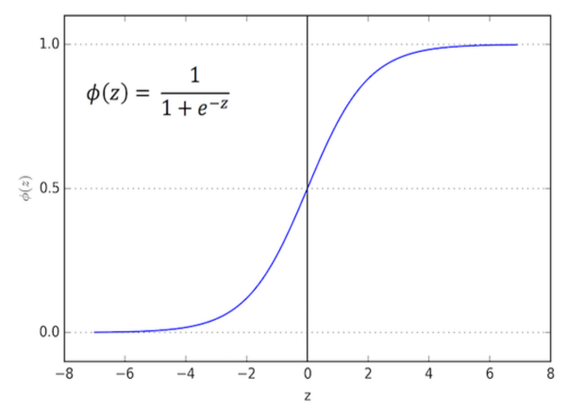
$$z = f(b + x \cdot w) = f\left(b + \sum_{i=1}^{n} x_i w_i\right)$$
$$x \in d_{1\times n}, \quad w \in d_{n\times 1}, \quad b \in d_{1\times 1}, z \in d_{1\times 1}$$

where :

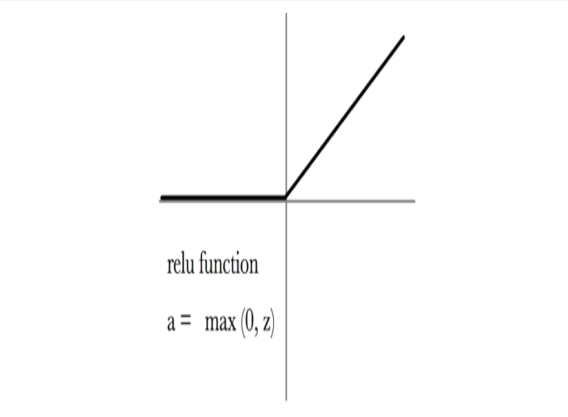
- b = bias,
- x = input to neuron,
- w = weights,
- n = the number of inputs from the incoming layer,
- i = a counter from 0 to n.0

(most used) Non-linear Activation function

- **Sigmoid activation function :** The Sigmoid activation function is an activation function that creates a flexible S-shaped (Sigmoid curve) with a minimum value approaching from zero and a maximum value approaching 1.



- **Rectified linear unit (ReLU) :** Rectified Linear Unit i.e. ReLU is another very popular activation function within machine learning. These days ReLU is most widely used as an activation function in deep learning problems. It is almost used in all convolutional neural network these days.



How Many Hidden Layers/Neurons to Use in Artificial Neural Networks ?

Here are some guidelines to know the number of hidden layers and neurons per each hidden layer in a classification problem :

1) Based on the data, draw an expected decision boundary to separate the classes.
2) Express the decision boundary as a set of lines. Note that the combination of such lines must yield to the decision boundary.
3) The number of selected lines represents the number of hidden neurons in the first hidden layer.
4) To connect the lines created by the previous layer, a new hidden layer is added. Note that a new hidden layer is added each time you need to create connections among the lines in the previous hidden layer.
5) The number of hidden neurons in each new hidden layer equals the number of connections to be made.

Example

Let's start with a simple example of a classification problem with two classes as shown in figure. Each sample has two inputs and one output that represents the class label. It is much similar to XOR problem.

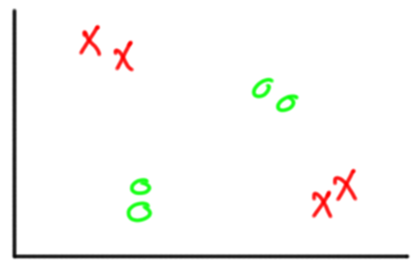
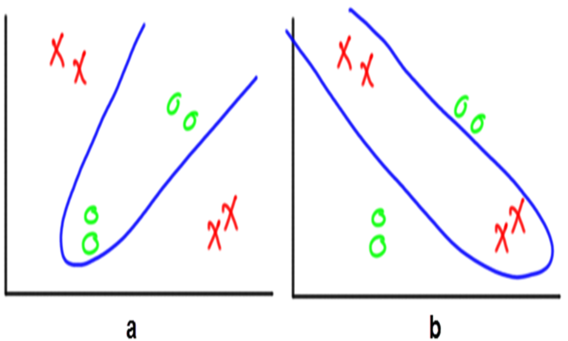


FIGURE – Classification Problem.

In order to add hidden layers, we need to answer these following two questions :

- What is the required number of hidden layers ?
- What is the number of the hidden neurons across each hidden layer ?

Following the previous procedure, the first step is to draw the decision boundary that splits the two classes.



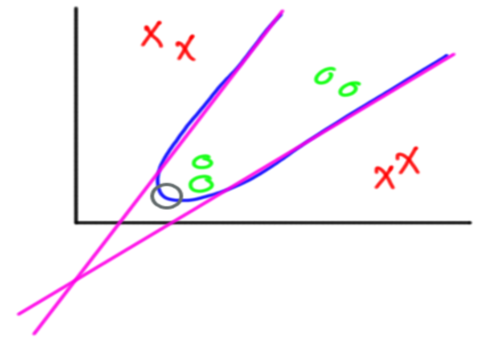
Following the guidelines, next step is to express the decision boundary by a set of lines.

$$y = w_1 * x_1 + w_2 * x_2 + \cdot + w_i * x_i + b$$

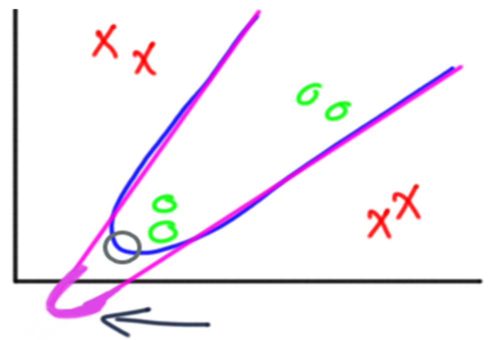
where :

- x_i is the input,
- w_i is its weight,
- b is the bias,
- and y is the output.

In this example, the decision boundary is replaced by a set of lines. The lines start from the points at which the boundary curve changes direction. At such point, two lines are placed, each in a different direction.



Up to this point, we have a single hidden layer with two hidden neurons. There will be two outputs, one from each classifier (i.e. hidden neuron). But we are to build a single classifier. As a result, the outputs of the two hidden neurons are to be merged into a single output. In other words, the two lines are to be connected by another neuron.



The use of Deep Learning

Deep learning is used in a lot of things like :

- Customer Support : When most people converse with customer support agents the conversation seems so real they don't even realize that it's a robot on the other side.
- Medical Care : Neural Network Detect cancer cells and analyze MRI images.
- Self Driving Cars (what seem like science fiction is now a reality).
- Natural language processing (so what is NLP ?).

Natural Language Processing (NLP)

Natural Language Processing

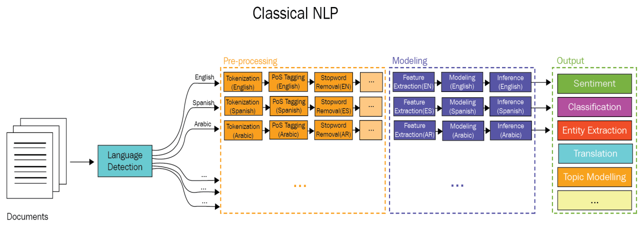
Natural language processing (NLP) : is concerned with the interactions between computers and human natural languages in general and in particular how to use computers to process and analyze natural language data (e.g., text, voice, etc.).

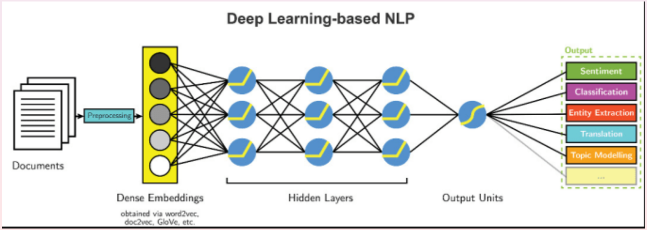
Applications of NLP

- Sentimental Analysis
- Chatbot
- Speech Recognition
- Machine Translation
- Spell checking, keyword search, finding synonyms .
- Extracting information from websites such as : product price, dates, location, people or company names.

Deep Learning for NLP

Deep Learning for NLP





Word Embeddings

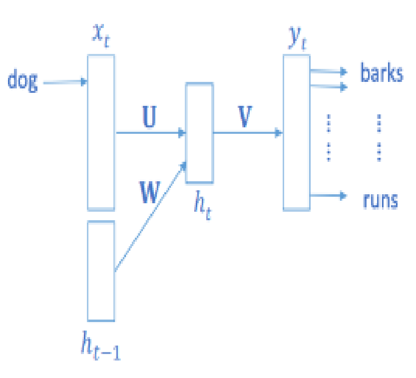
- Distributional vectors, also called word embeddings, are based on the so-called distributional hypothesis words appearing within similar context possess similar meaning. Word embeddings are pre-trained on a task where the objective is to predict a word based on its context, typically using a shallow neural network.
- A word embedding W : words $\to \mathbb{R}^n$ is a parameterized function mapping words in some language to high-dimensional vectors (perhaps 200 to 500 dimensions). For example, we might find :

$$W(\text{"}cat\text{"}) = (0.2, -0.4, 0.7, \dots)$$
$$W(\text{"}mat\text{"}) = (0.0, 0.6, -0.1, \dots)$$

There are a lot of popular methods for creating word vectors !

- Vector Space Model [Salton & McGill 83]
- Latent Semantic Analysis [Deerwester+ 90]
- Brown Clustering [Brown+ 92]
- Latent Dirichlet Allocation [Blei+ 03]
- Deep Neural Networks [Collobert & Weston 08]
- DSSM [Huang+ 13]
- Word2Vec [Mikolov+ 13]
- GloVe [Pennington+ 14]

Recurrent Neural Network (RNN) for Language Modeling



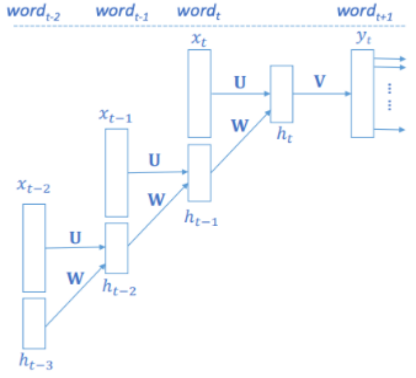
- x_t : input one-hot vector at time step t,
- h_t : encodes the history of all words up to time step t,
- y_t : distribution of output words at time step t,

- $z_t = Ux_t + Wh_{t-1}$
- $h_t = \sigma(z_t)$
- $y_t = g(Vh_t)$

  where
- $\sigma(z) = \dfrac{1}{1+\exp(-z)}$,

  $g(z_m) = \dfrac{\exp(z_m)}{\sum_k \exp(z_k)}$, $g(\cdot)$ is called the softmax function

RNN unfolds into a DNN over time



- $z_t = Ux_t + Wh_{t-1}$
- $h_t = \sigma(z_t)$
- $y_t = g(Vh_t)$

  where
- $$\sigma(z) = \frac{1}{1+\exp(-z)},$$
  $$g(z_m) = \frac{\exp(z_m)}{\sum_k \exp(z_k)}.$$

- This approach has one huge problem, when all neurons remember their past results, the number of connections in the network become so huge that it is technically impossible to adjust all the weights.
- The first decision was simple : limit the neuron memory. Let's say, to memorize no more than 5 recent results. But it broke the whole idea.
- A much better approach came later : to use special cells, similar to computer memory. Each cell can record a number, read it or reset it. They were called Long and Short-Term Memory (LSTM) cells.

Question ?

- Is Deep Learning the best solution for all NLP problems ?
- How to make the machine understand natural language like humans ?
- How can the machine understands the humans sentiments ?
- How can the machine generates new text for us just by knowing what we like to read ?

References

- Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita, *A Survey of the Usages of Deep Learning for Natural Language Processing*, https ://arxiv.org/pdf/1807.10854.pdf.
- LeCun Y., Bengio Y., and Hinton G., *Deep learning*, Nature, Vol. 521, 436–444, (2015).
- Hirschberg J., and Manning C. D., *Advances in natural language processing*, Science, Vol. 349, Issue. 6245, 261–266, (2015).
- Stojanovski D., Strezoski G., Madjarov G. and Dimitrovski I. *Deep Learning Architecture for Twitter Sentiment Analysis* Proceedings of SemEval-2016, 154–159, (2016).

Thank you for your attention

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 694126- DYCON).