

Natural Language Processing

Linear Models and Neural Networks

Felipe Bravo-Marquez

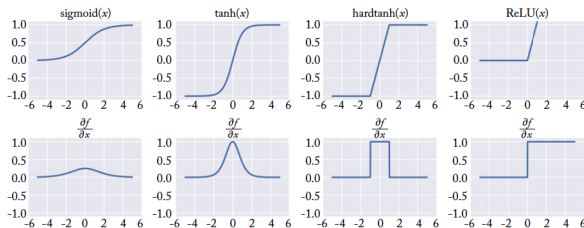
July 17, 2019

Introduction to Neural Networks

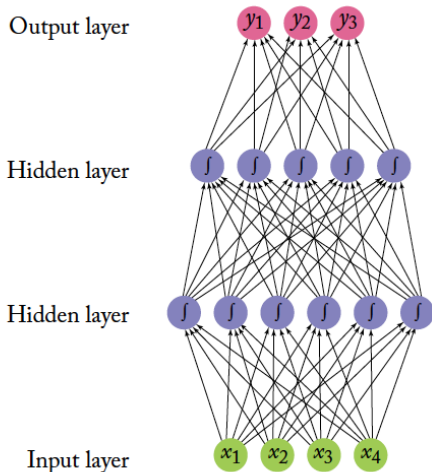
- Very popular machine learning models formed by units called **neurons**.
- A neuron is a computational unit that has scalar inputs and outputs.
- Each input has an associated weight w .
- The neuron multiplies each input by its weight, and then sums them (other functions such as **max** are also possible).
- It applies an activation function g (usually non-linear) to the result, and passes it to its output.
- Multiple layers can be stacked.

Activation Functions

- The nonlinear activation function g has a crucial role in the network's ability to represent complex functions.
- Without the nonlinearity in g , the neural network can only represent linear transformations of the input.



Feedforward Network with two Layers



Brief Introduction to Neural Networks

- The feedforward network from the picture is a stack of linear models separated by nonlinear functions.
- The values of each row of neurons in the network can be thought of as a vector.
- The input layer is a 4-dimensional vector (\vec{x}), and the layer above it is a 6-dimensional vector (\vec{h}^1).
- The fully connected layer can be thought of as a linear transformation from 4 dimensions to 6 dimensions.
- A fully connected layer implements a vector-matrix multiplication, $\vec{h} = \vec{x}W$.
- The weight of the connection from the i -th neuron in the input row to the j -th neuron in the output row is $W_{[i,j]}$.
- The values of \vec{h} are transformed by a nonlinear function g that is applied to each value before being passed on as input to the next layer.

⁰Vectors are assumed to be row vectors and superscript indices correspond to network layers.

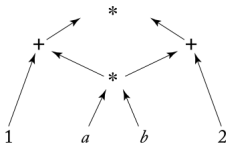
Brief Introduction to Neural Networks

- The Multilayer Perceptron (MLP) from the figure can be written as the following mathematical function:

$$\begin{aligned} NN_{MLP2}(\vec{x}) &= \vec{y} \\ \vec{h}^1 &= g^1(\vec{x}W^1 + \vec{b}^1) \\ \vec{h}^2 &= g^2(\vec{h}^1W^2 + \vec{b}^2) \\ \vec{y} &= \vec{h}^2W^3 \\ \vec{y} &= (g^2(g^1(\vec{x}W^1 + \vec{b}^1)W^2 + \vec{b}^2))W^3. \end{aligned} \tag{1}$$

The Computation Graph Abstraction

- One can compute the gradients of the various parameters of a network by hand and implement them in code.
- This procedure is cumbersome and error prone.
- For most purposes, it is preferable to use automatic tools for gradient computation [Bengio, 2012].
- A computation graph is a representation of an arbitrary mathematical computation (e.g., a neural network) as a graph.
- Consider for example a graph for the computation of $(a * b + 1) * (a * b + 2)$:



- The computation of $a * b$ is shared.
- The graph structure defines the order of the computation in terms of the dependencies between the different components.

The Computation Graph Abstraction

- The computation graph abstraction allows us to:
 1. Easily construct arbitrary networks.
 2. Evaluate their predictions for given inputs (forward pass)

Algorithm 5.3 Computation graph forward pass.

```
1: for i = 1 to N do
2:   Let  $a_1, \dots, a_m = \pi^{-1}(i)$ 
3:    $v(i) \leftarrow f_i(v(a_1), \dots, v(a_m))$ 
```

3. Compute gradients for their parameters with respect to arbitrary scalar losses (backward pass or backpropagation).

Algorithm 5.4 Computation graph backward pass (backpropagation).

```
1:  $d(N) \leftarrow 1$   $\triangleright \frac{\partial N}{\partial N} = 1$ 
2: for i = N-1 to 1 do
3:    $d(i) \leftarrow \sum_{j \in \pi(i)} d(j) \cdot \frac{\partial f_j}{\partial i}$   $\triangleright \frac{\partial N}{\partial i} = \sum_{j \in \pi(i)} \frac{\partial N}{\partial j} \frac{\partial j}{\partial i}$ 
```

- The backpropagation algorithm (backward pass) is essentially following the chain-rule of differentiation¹.

¹A comprehensive tutorial on the backpropagation algorithm over the computational graph abstraction:

Deep Learning Frameworks

Several software packages implement the computation-graph model. All these packages support all the essential components (node types) for defining a wide range of neural network architectures.

- TensorFlow (<https://www.tensorflow.org/>): an open source software library for numerical computation using data-flow graphs originally developed by the Google Brain Team.
- Keras: High-level neural network API that runs on top of Tensorflow as well as other backends (<https://keras.io/>).
- PyTorch: open source machine learning library for Python, based on Torch, developed by Facebook's artificial-intelligence research group. It supports dynamic graph construction, a different computation graph is created from scratch for each training sample. (<https://pytorch.org/>)

Questions?

Thanks for your Attention!

References I



Goldberg, Y. (2017).

Neural network methods for natural language processing.

Synthesis Lectures on Human Language Technologies, 10(1):1–309.