Foundations of Machine Learning in Python

Moritz Wolter

August 30, 2022

High-Performance Computing and Analytics Lab

Overview

Neural networks

Estimation, Overfitting and Regularization

Classification

Neural networks

The wonders of the human visual system

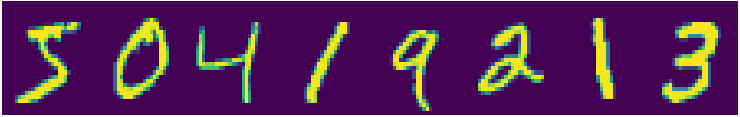
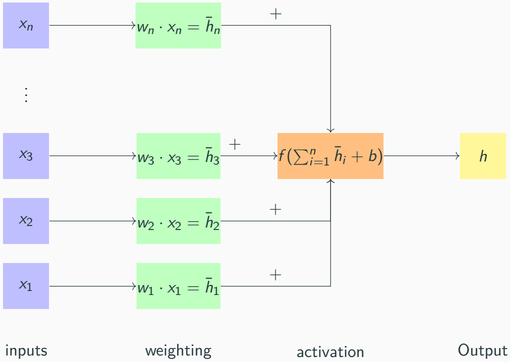


Figure: Most humans effortlessly recognize the digits 5 0 4 1 9 2 1 3.

The perceptron

Can computers recognize digits? Mimic biological neurons,



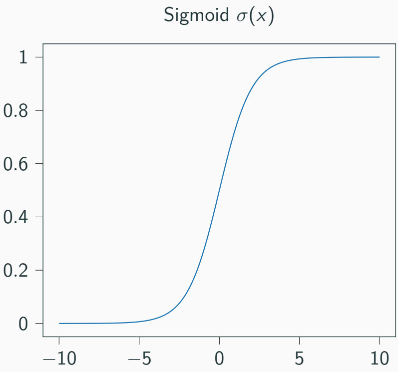
Formally a single perceptron is defined as

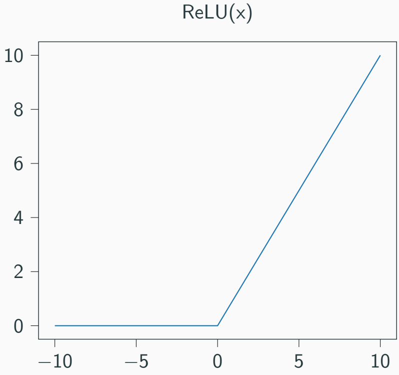
$$f(\mathbf{w}^T \mathbf{x}) = h \tag{1}$$

with $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{x} \in \mathbb{R}^n$ and $h \in \mathbb{R}$.

The activation function f

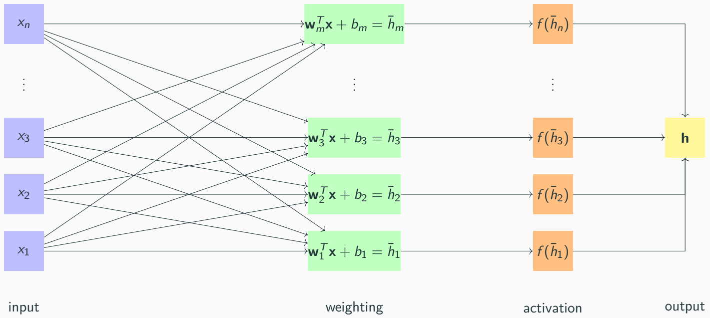
Two popular choices for the activation function f.





Arrays of perceptrons

Let's extend the definition to cover an array of perceptrons:



Every input is connected to every neuron. In matrix language, this turns into

$$\bar{\mathbf{h}} = \mathbf{W}\mathbf{x} + \mathbf{b}, \qquad \mathbf{h} = f(\bar{\mathbf{h}}). \tag{2}$$

With $\mathbf{W} \in \mathbb{R}^{m,n}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{h}, \bar{\mathbf{h}} \in \mathbb{R}^m$.

The loss function

To choose weights for the network, we require a quality measure. We already saw the mean squared error cost function,

$$C_{\text{mse}} = \frac{1}{2}\sum_{k=1}^{n}(\mathbf{y}_k - \mathbf{h}_k)^2 = \frac{1}{2}(\mathbf{y} - \mathbf{h})^T(\mathbf{y} - \mathbf{h}) \tag{3}$$

This function measures the squared distance from each desired output. $\mathbf{y}$ denotes the desired labels, and $\mathbf{h}$ represents network output.

The gradient of the mse-cost-function

Both the mean squared error loss function and our dense layer are differentiable.

$$\frac{\partial C_{\text{mse}}}{\partial \mathbf{h}} = \mathbf{h} - \mathbf{y} = \triangle_{\text{mse}} \tag{4}$$

The $\triangle$ symbol will re-appear. It always indicates incoming gradient information from above. If the labels are a vector of shape $\mathbb{R}^m$, $\triangle$ and the network output $\mathbf{h}$ must share this dimension.

The gradient of a dense layer

The chain rule tells us the gradients for the dense layer[Nie15]

$$\delta \mathbf{W} = [f'(\bar{\mathbf{h}})]\mathbf{x}^T, \qquad \delta \mathbf{b} = f'(\bar{\mathbf{b}}) \odot \triangle, \tag{5}$$

$$\delta \mathbf{x} = \mathbf{W}^T [f'(\bar{\mathbf{h}}) \odot \triangle], \tag{6}$$

where $\odot$ is the element-wise product. δ denotes the gradient for the value following it [Gre+16].

Good news! Jax can take care of these computations for you!

You can choose to verify these equations by completing the optional deep learning project.

The gradient of a dense layer

The chain rule tells us the gradients for the dense layer[Nie15]

$$\delta\mathbf{W} = [f'(\bar{\mathbf{h}})]\mathbf{x}^T, \qquad \delta\mathbf{b} = f'(\bar{\mathbf{b}}) \odot \triangle, \tag{5}$$

$$\delta\mathbf{x} = \mathbf{W}^T[f'(\bar{\mathbf{h}}) \odot \triangle], \tag{6}$$

where $\odot$ is the element-wise product. δ denotes the gradient for the value following it [Gre+16].
Good news! Jax can take care of these computations for you!
You can choose to verify these equations by completing the optional deep learning project.

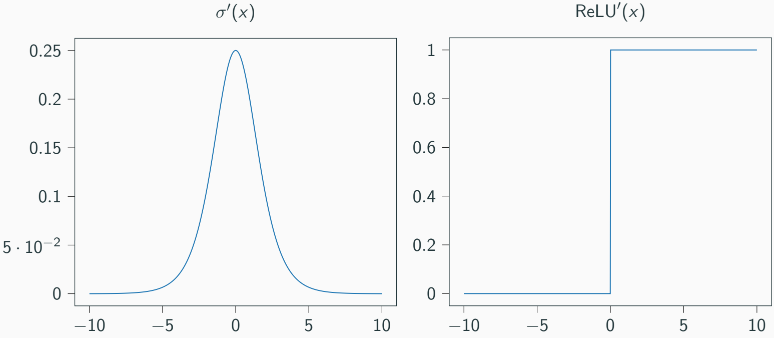
On the board, derive: Recall the chain rule $(g(h(x)))' = g'(h(x)) \cdot h'(x)$. Looking at

$$\bar{\mathbf{h}} = f(\bar{\mathbf{h}}) \Rightarrow \delta\bar{\mathbf{h}} = f'(\bar{\mathbf{h}}) \odot \triangle \tag{7}$$

Derivatives of our activation functions

$$\sigma'(x) = \sigma \cdot (1 - \sigma(x)) \tag{8}$$

$$\mathrm{ReLU}' = H(x) \tag{9}$$



Perceptrons can learn functions

TODO

Multi-layer networks

TODO

Backpropagation

TODO

Estimation, Overfitting and Regularization

Denoising a signal

TODO

Classification

The cross-entropy loss

TODO

MNIST digit

Modified National Institute of Standards and Technology database [**dumoulin2016guide**]