Deep Learning

Chapter 1: A Single Neuron

David Rügamer

Department of Statistics – LMU Munich

Winter Semester 2020

A SINGLE NEURON

- In order to better understand the types of functions that neural networks can represent, let us begin with a very simple model: logistic regression.
- Recall: The hypothesis space of logistic regression can be written as

$$\mathcal{H} = \left\{ f : \mathbb{R}^p \to [0,1] \;\middle|\; f(\mathbf{x}) = \tau\left(\sum_{j=1}^{p} w_j x_j + b\right), \mathbf{w} \in \mathbb{R}^p, b \in \mathbb{R} \right\},$$

where $\tau(z) = (1 + \exp(-z))^{-1}$ is the logistic sigmoid function.

- It is straightforward to represent this function $f(\mathbf{x})$ graphically as a neuron.
- Note: $\mathbf{w}$ and b together constitute $\boldsymbol{\theta}$.

A SINGLE NEURON

We consider a logistic regression model for $p = 3$, i.e.
$f(\mathbf{x}) = \tau(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$.

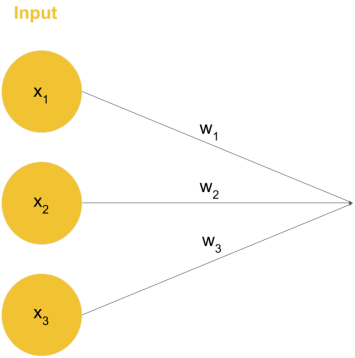
- First, features of $\mathbf{x}$ are represented by nodes in the "input layer".

- In general, a p-dimensional input vector $\mathbf{x}$ will be represented by p nodes in the input layer.

A SINGLE NEURON

We consider a logistic regression model for $p = 3$, i.e.
$f(\mathbf{x}) = \tau(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$.

- Next, weights $\mathbf{w}$ are represented by edges from the input layer.



- The bias term b is implicit. It is not shown/represented visually.

A SINGLE NEURON

Note: For an explicit graphical representation of the bias term, we can do a simple trick:

- we add a constant feature to the inputs $\tilde{\mathbf{x}} = (1, x_1, ..., x_p)^\top$
- and the bias term to the weight vector $\tilde{\boldsymbol{w}} = (b, w_1, ..., w_p)$.

The graphical representation is then:

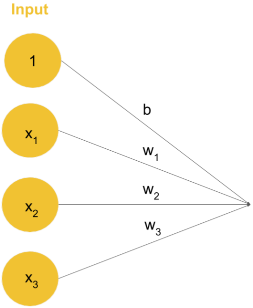
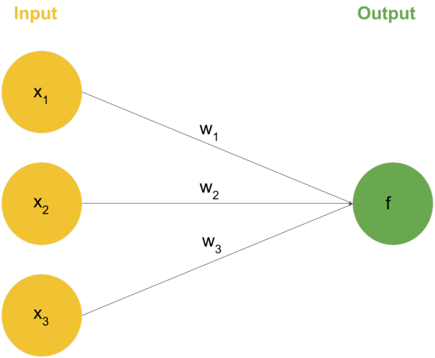


Figure: Weights and bias of the neuron.

A SINGLE NEURON

We consider a logistic regression model for $p = 3$, i.e. $f(\mathbf{x}) = \tau(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$.

- Finally, the computation $\tau(w_1 x_1 + w_2 x2 + w_3 x_3 + b)$ is represented by the neuron in the "output layer".



Because this single neuron represents exactly the same hypothesis space as logistic regression, it can only learn linear decision boundaries.

A SINGLE NEURON

- Therefore, a neuron is just a graphical representation of a very specific kind of function.

- Every neuron performs a 2-step computation:
  - Step 1: compute the weighted sum of inputs (with bias).
  - Step 2: apply an **activation function** to the sum, which is usally a non-linear transformation of the input.

- With a single neuron, the activation function serves to constrain the output to the desired range of values. In case of logistic regression. For example, it squashes the output into $[0, 1]$.

- However, we will see that activation functions serve a far more important purpose. They are one of the main reasons that neural networks can represent extremely complicated functions.

A SINGLE NEURON

- One very nice thing about the graphical representation of the function is that you can picture the input vector being "fed" to the neuron on the left followed by a sequence of computations being performed from left to right. This is called a “**forward pass**”.

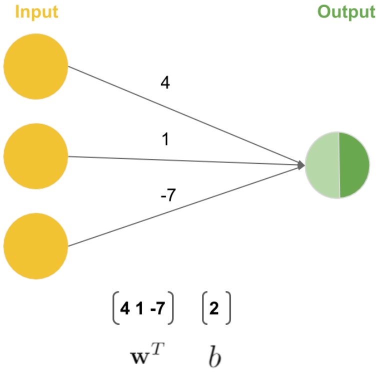


Figure: Weights (and bias) of the neuron.

A SINGLE NEURON

- One very nice thing about the graphical representation of the function is that you can picture the input vector being "fed" to the neuron on the left followed by a sequence of computations being performed from left to right. This is called a “**forward pass**”.

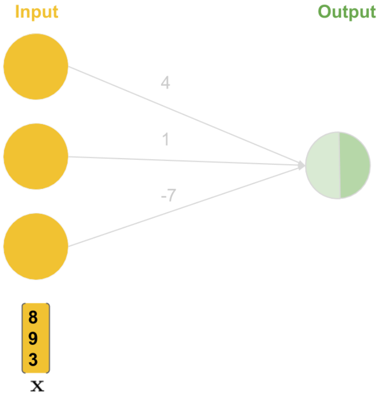


Figure: Feed the input on the left.

A SINGLE NEURON

- One very nice thing about the graphical representation of the function is that you can picture the input vector being "fed" to the neuron on the left followed by a sequence of computations being performed from left to right. This is called a “**forward pass**”.

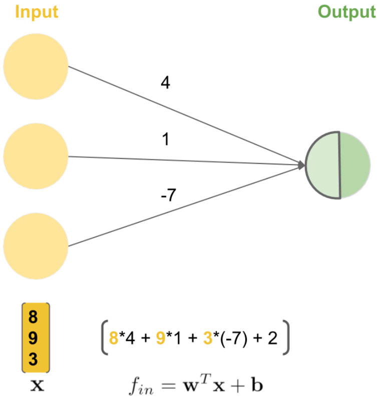


Figure: Step 1: Compute the weighted sum.

A SINGLE NEURON

- One very nice thing about the graphical representation of the function is that you can picture the input vector being "fed" to the neuron on the left followed by a sequence of computations being performed from left to right. This is called a “**forward pass**”.

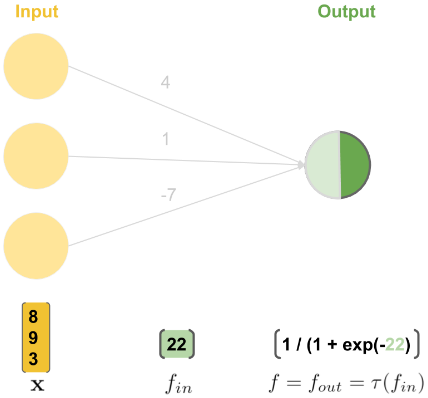
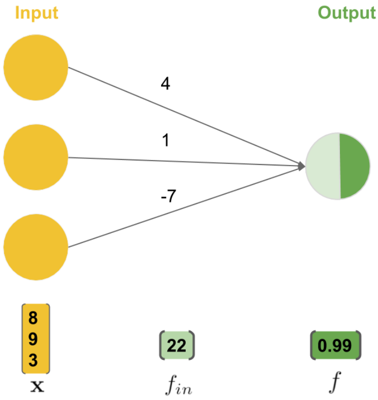


Figure: Step 2: Apply the activation function.

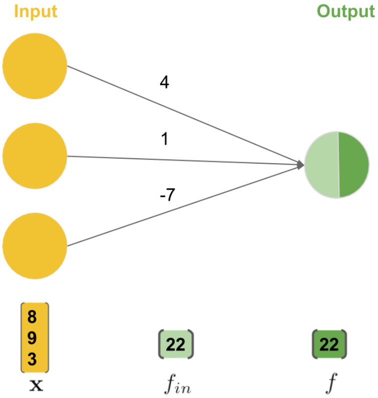
A SINGLE NEURON

- One very nice thing about the graphical representation of the function is that you can picture the input vector being "fed" to the neuron on the left followed by a sequence of computations being performed from left to right. This is called a “**forward pass**”.



A SINGLE NEURON

- Even though all neurons compute a weighted sum in the first step, there is considerable flexibility in the type of activation function used in the second step.
- For example, setting the activation function to the identity function allows a neuron to represent linear regression.



A SINGLE NEURON

- The hypothesis space that is formed by single neuron architectures is

$$\mathcal{H} = \left\{ f : \mathbb{R}^p \to \mathbb{R} \;\middle|\; f(\mathbf{x}) = \tau\left(\sum_{j=1}^{p} w_j x_j + b\right), \mathbf{w} \in \mathbb{R}^p, b \in \mathbb{R} \right\}.$$

- Both logistic regression and linear regression are subspaces of $\mathcal{H}$ (if τ is the logistic sigmoid / identity function).

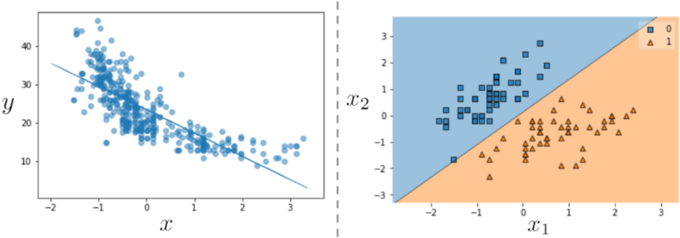


Figure: *Left*: A regression line learned by a single neuron. *Right*: A decision-boundary learned by a single neuron in a binary classification task.

A SINGLE NEURON: OPTIMIZATION

- To optimize this model, we minimize the empirical risk

$$\mathcal{R}_{\text{emp}} = \sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right),$$

where $L(y, f(\mathbf{x}))$ is a loss function. It compares the network's predictions $f(\mathbf{x})$ to the ground truth y.

- For regression, we typically use the L2 loss (rarely L1):

$$L(y, f(\mathbf{x})) = \frac{1}{2}(y - f(\mathbf{x}))^2$$

- For binary classification, we typically apply the cross entropy loss (also known as bernoulli loss):

$$L(y, f(\mathbf{x})) = y \log f(\mathbf{x}) + (1 - y) \log(1 - f(\mathbf{x}))$$

A SINGLE NEURON: OPTIMIZATION

- For a single neuron, in both cases, the loss function is convex and the global optimum can be found with an iterative algorithm like gradient descent.
- In fact, a single neuron with logistic sigmoid function trained with the bernoulli loss does not only have the same hypothesis space as a logistic regression and is therefore the same model, but will also yield to the very same result when trained until convergence.