Outline

- Artificial neurons. Perceptron and sigmoid
- Designing Feedforward neural network for classification

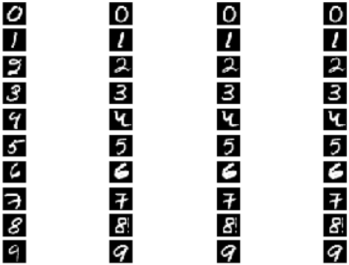
What is Neural Network and deep learning

Neural Network: Biological Neuron inspired, mathematical model. Inspiration is beautiful if you belive in connectionism.

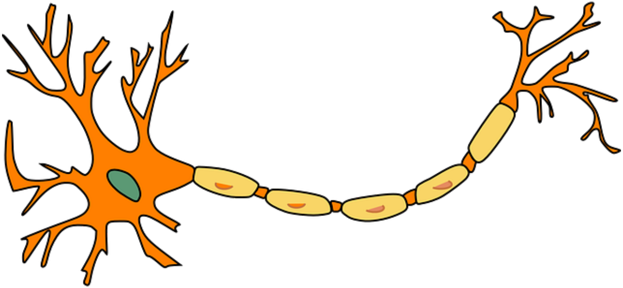
Connectionism(wikipedia): Connectionism is a set of approaches in the fields of artificial intelligence, cognitive psychology, cognitive science, neuroscience, and philosophy of mind, that models mental or behavioral phenomena as the emergent processes of **interconnected networks of simple units**. **There are many forms of connectionism, but the most common forms use neural network models.**

Deep learning: Set of techinque for training deep neural network.

Writing Algorithm for digit recognition. Human vs Machine



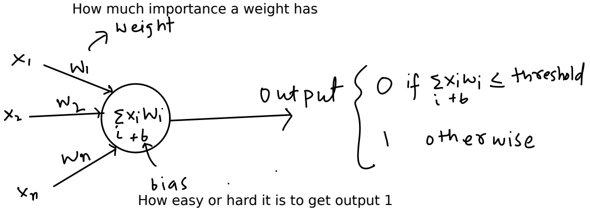
Bilogical neuron



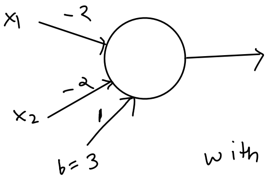
credit:https://pixabay.com/en/neuron-nerve-cell-axon-dendrite-296581

Perceptron

Developed in the 1950s and 1960s by the Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts.



Perceptron as NAND



with threshold = 0
this perceptron is NAND gate

Perceptron and training (How to learn w_i and b)

1. As we change weights in perceptron out may flip completly from 0 to 1.
2. Training perceptron or network of perceptron is hard

Can we model output in the range $[0\,1]$ more gradually as a function of inputs

A New type of artificial neuron. Sigmoid

For input $\boldsymbol{x}$, and $\boldsymbol{w} \in \mathbb{R}^n$, and bias $b \in \mathbb{R}$ sigmoid function is

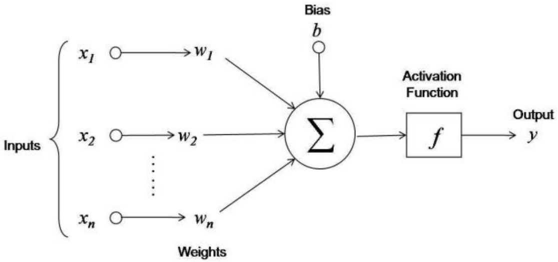
$$\sigma(\boldsymbol{x}) = \frac{1}{1 + \exp -[\sum_{i=1}^{n}(w_i x_i + b)]}$$

Using calculaus we can show that
$\triangle\sigma(\boldsymbol{x}) \approx \sum_{i=1}^{n} \frac{\partial\sigma(x)}{\partial w_i} \triangle w_i + \frac{\partial\sigma(x)}{\partial b} \triangle b$

Small change in output if small change in w_i and b

Without Brain Stuff: Artificial Neuron Activation Model

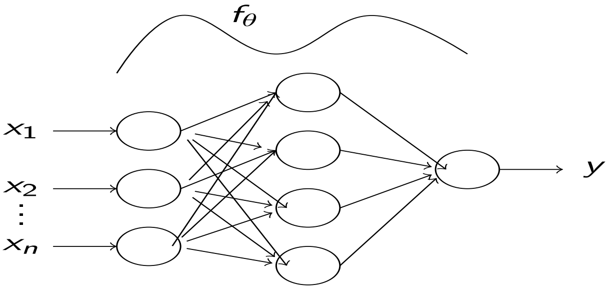
A typical model for artificial neuron activation is



$y = f(g(X; W, b))$, where f is non linear activation function.

Affine function g (pre-activation function) measures **global($\sum_i w_i x_i + b$) or local (e.g convolution $X * W + b$)** similarity, correlation.

Feedforward Neural Networks(Fully connected layer)



End to end classifier

- Neural networks computes the function $\boldsymbol{y_i} = f_\theta(\boldsymbol{x_i})$.
- Highly non linear end to end.
- Parameters θ can be learned via gradient descent by minimizing $\sum_{i=1}^{N}(\boldsymbol{y_i} - f_\theta(\boldsymbol{x_i}))^2$.

Locally connected

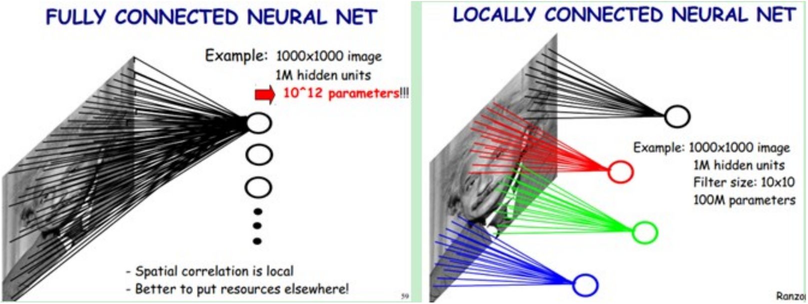


image credit:https://irenelizihui.files.wordpress.com/2016/02/cnn1.png

Representing operation between two layer via matrix multiplication and pointwise operation

How to learn the weights and biases of various AN(artificial neuron in the network?)

Let say we want to classify a digit $\boldsymbol{x} \in R^{28x28=784}$ in MNIST dataset. So we build a Feedworad neural network f giving us output $\boldsymbol{y} = f(\boldsymbol{x})$ where $\boldsymbol{y}$ is 10 dimenstional output like $[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$

We use gradient descent or variant Link of it to find weights and biases in AN such that network predicts correct output for training example or minimizes some cost function.
Main ides of gradient descent is

$$w_{k+1} = w_k - \eta \frac{\partial \text{cost or loss function}}{\partial w_k}$$

like **squared loss** i.e.

$$\ell(D;\theta) = \sum_{i=1}^{N}(\boldsymbol{y_i} - f_\theta(\boldsymbol{x_i}))^2$$

given that network outputs continous value vector/scalar. or **cross entropy**

$$\ell(D;\theta) = -\sum_{i=1}^{N} \boldsymbol{y_i}^T f_\theta(\boldsymbol{x_i})$$

given that network outputs probabilty of different classes and class label $\boldsymbol{y_i}$ is **onehot** encoded. Here θ represents all the weights and biases of all the ANs and other paramters in the neural network f.

Backpropogation

Thank you!

Thank you!