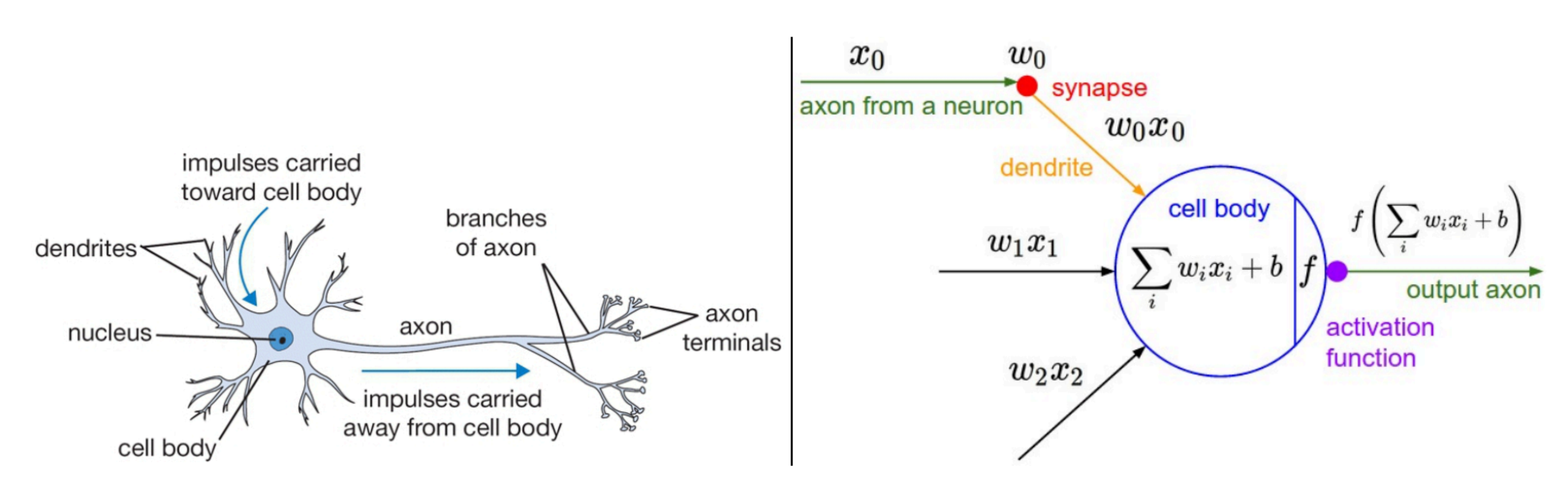
NEURAL NETWORK1

http://cs231n.github.io/neural-networks-1/

Hayeong Lee
hy-kiera
github.com/hy-kiera

Biological motivation and connections



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Firing rate (->activation function f) : represents the frequency of the spikes along the axon

Single neuron as a linear classifier

Linear classifiers -> neuron

Like(activation near one)

Dislike(activation near zero)

With an appropriate loss function on the neuron's output,

We can turn a single neuron into a linear classifier.

A single neuron can be used to implement a binary classifier

Single neuron as a linear classifier

- Binary Softmax classifier(logistic regression)

$$\sigma(\sum_i w_i x_i + b) \longrightarrow \begin{array}{l} P(y_i = 1 \mid x_i ; w) \\ P(y_i = 0 \mid x_i ; w) = 1 - P(y_i = 1 \mid x_i ; w) \end{array}$$

=> cross-entropy loss

Since the sigmoid function is restricted to be between 0-1,

the predictions of this classifier are based on whether the output of the neuron is grater than 0.5.

Single neuron as a linear classifier

- Binary SVM classifier

  => max-margin hinge loss

- Regularization interpretation

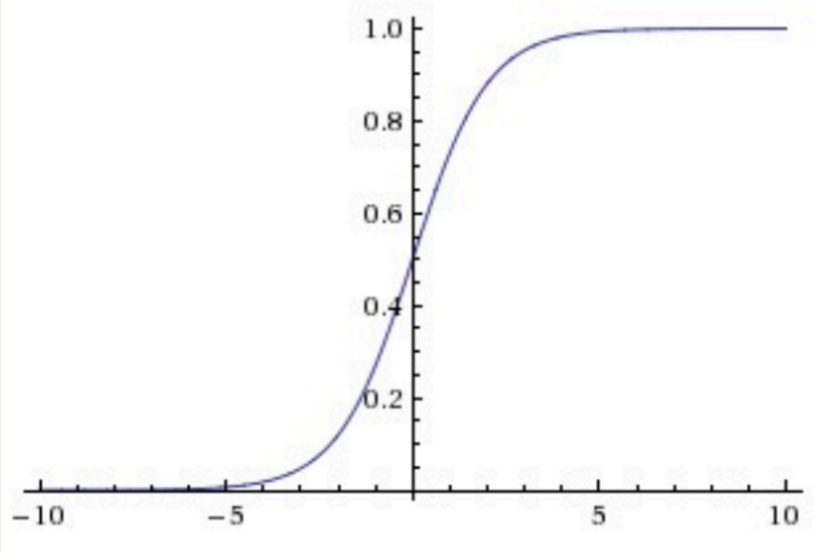
  => The regularization loss in both SVM/Softmax cases -> gradual forgetting

  ∵ effect of driving all synaptic weights w towards zero after every parameter update

Commonly used activation functions

- Sigmoid

$$\sigma(x) = \frac{1}{(1 + e^{-x})}$$



- takes a real-value number

- “squashes” it into range between 0 and 1

Large negative number -> 0 (not firing at all)

Large positive number -> 1 (fully-saturated firing at an assumed maximum frequency)

-> nice interpretation as the firing rate of neuron

Commonly used activation functions

- Sigmoid – two major drawbacks

1. Sigmoid saturate and kill gradients

   when the neuron’s activation saturates at either tail of 0 or 1,

   the gradient at these regions is almost zero because of backpropagation

2. Sigmoid output are not zero-centered

   neurons in later layers of processing in a Neural Network

   would be receiving data that is not zero-centered

   -> implications on the dynamics(zig-zagging dynamics) during gradient descent

   ∵ if the data coming into a neuron is always positive,

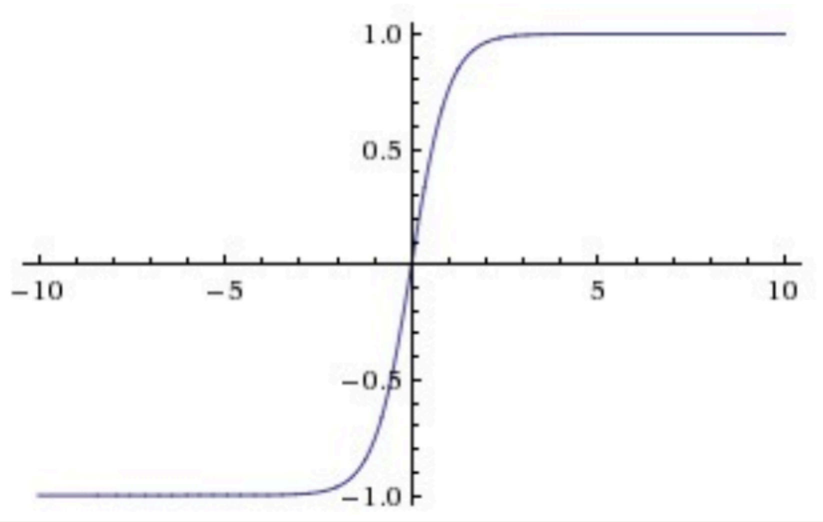
   then the gradient on the weights w will during backpropagation become either

   all be positive, or all negative (depending on the gradient of the whole expression f)

Commonly used activation functions

- Tanh

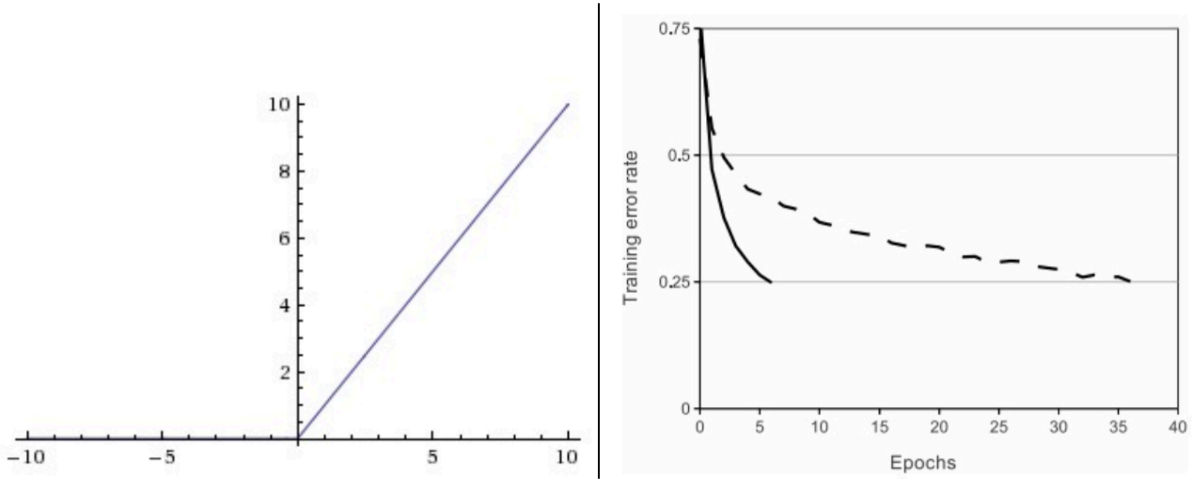
$$\tanh(x) = 2\sigma(2x) - 1$$



- squashes a real-valued number to the range [-1, 1] its activations saturate
- its output is zero-centered

Commonly used activation functions

- ReLU



Left: Rectified Linear Unit (ReLU) activation function, which is zero when x < 0 and then linear with slope 1 when x > 0. **Right:** A plot from Krizhevsky et al. (pdf) paper indicating the 6x improvement in convergence with the ReLU unit compared to the tanh unit.

$$f(x) = \max(0, x)$$

-> simply thresholded at zero

Commonly used activation functions

- ReLU – pros and cons

pros

1. It was found to greatly accelerate the convergence at stochastic gradient descent compared to sigmoid/tanh functions.(non-saturating form ∵linear)
2. Can be implemented by simple thresholding a matrix of activation at zero

cons

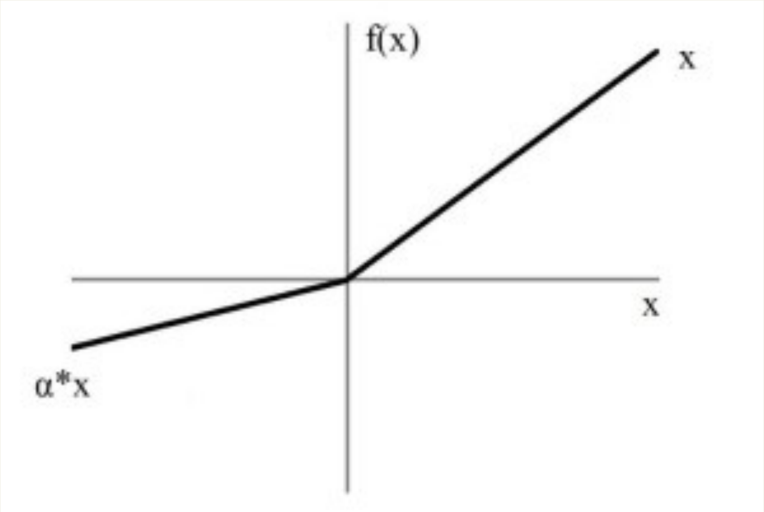
1. ReLU units can be fragile during training and can "die".

   A large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any datapoint again.

   -> the gradient flowing through the unit will forever be zero from that point on.

Commonly used activation functions

- Leaky ReLU



$$f(x) = 1(\mathrm{x} < 0)\alpha x + 1(x \geq 0)x$$

where α is a small constant

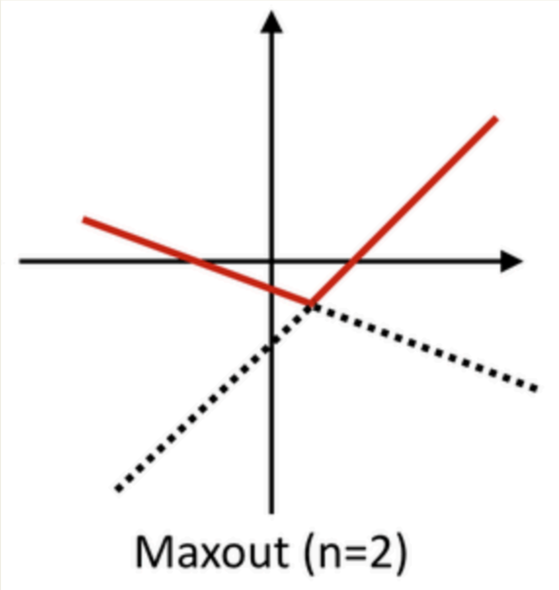
- It has a small negative slop (of 0.01, or so)
- The slope in the negative region can also be made into a parameter of each neuron, as seen in PReLU neurons

-> However, the consistency of the benefit across tasks is presently unclear

Commonly used activation functions

- Maxout

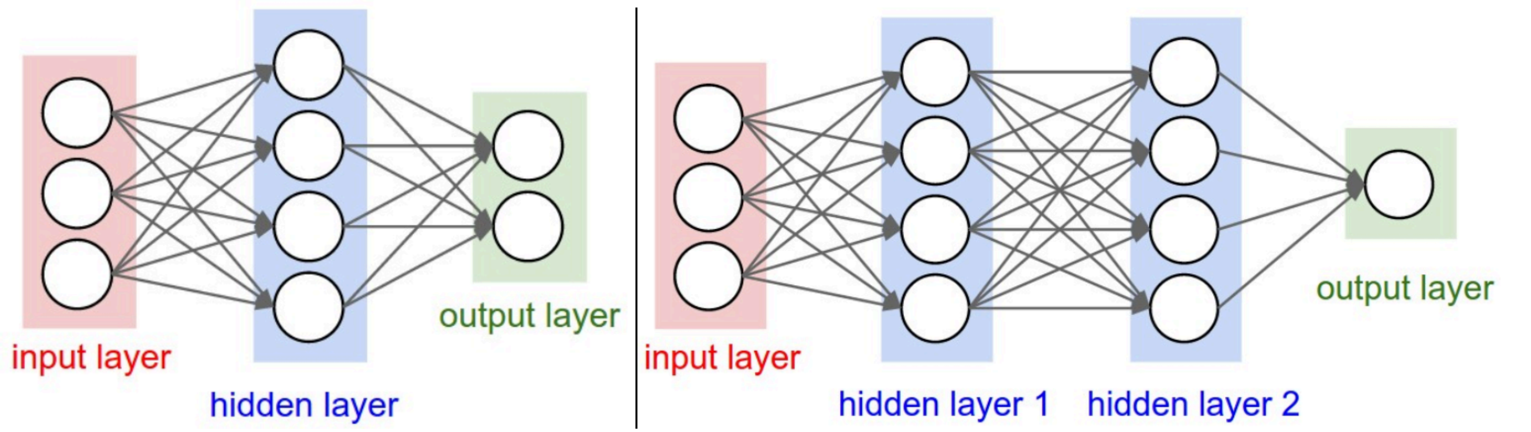
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$



- Enjoy all the benefits of a ReLU unit(linear regime of operation, no saturation)
- Do not have ReLU’s drawbacks(dying ReLU)
- It doubles the number of parameters for every single neuron, leading to a high total number of parameters

Neural Network architectures

- Layer-wise organization

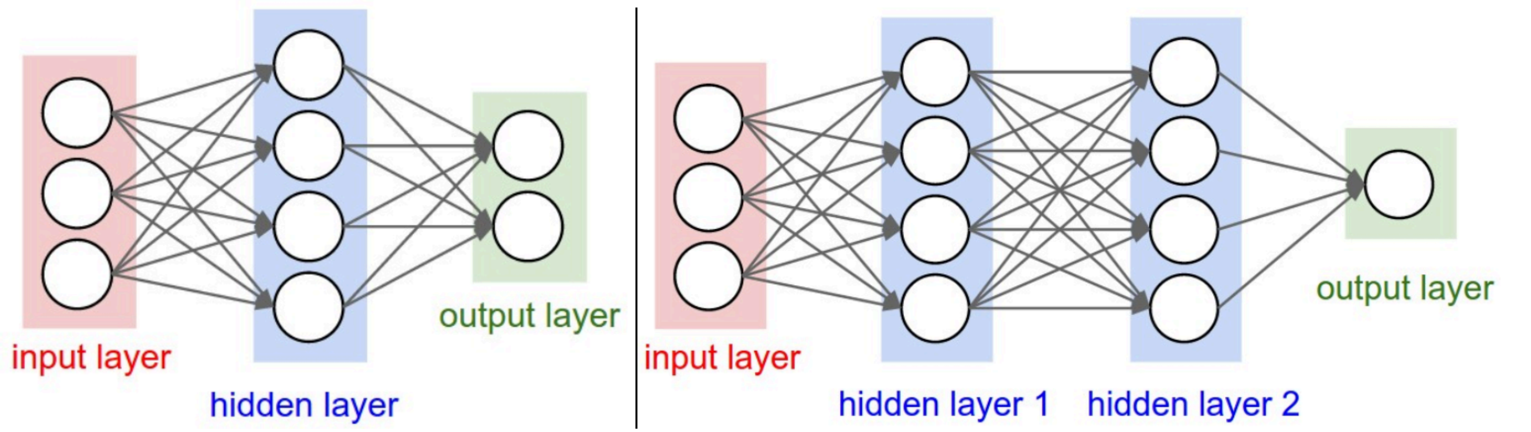


Left: A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs. **Right:** A 3-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer. Notice that in both cases there are connections (synapses) between neurons across layers, but not within a layer.

- Neural Networks : collection of neurons that are connected in an acyclic graph
- Fully-connected layer : neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections

Neural Network architectures

- Layer-wise organization



Left: A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs. **Right:** A 3-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer. Notice that in both cases there are connections (synapses) between neurons across layers, but not within a layer.

- Output layer : the output layer neurons most commonly do not have an activation function

∵ the last output layer is usually taken to represent the class scores

Neural Network architectures

- Representational power

Neural Network with at least one hidden layer are "universal approximators"

given any continuous function $f(x)$ and some $\varepsilon > 0$,

there exists a NN $g(x)$ with one hidden layer such that $\forall x, |f(x) - g(x)| < \mathcal{E}$.

=> The fact that deeper networks (with multiple hidden layers) can work better than a single-hidden-layer networks is an empirical observation, despite the fact that their representational power is equal.

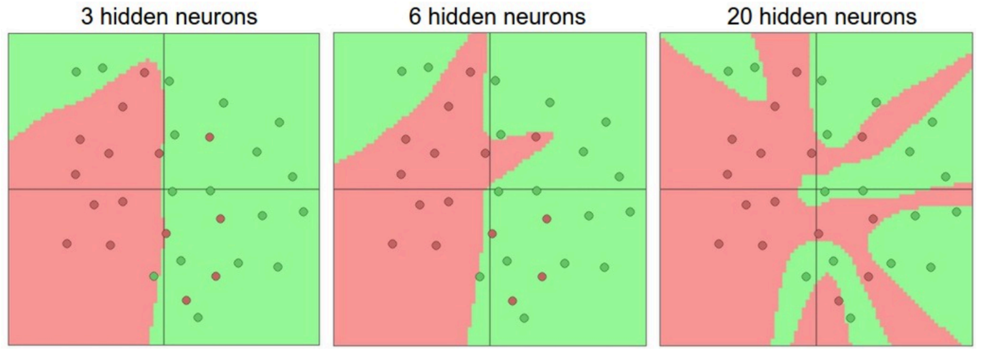
Neural Network architectures

- Setting number of layers and their sizes

As we increase the size and number of layers in a NN, the “capacity” of the network increases.

-> the space of representable functions grows

since the neurons can collaborate to express many different functions.



Larger Neural Networks can represent more complicated functions. The data are shown as circles colored by their class, and the decision regions by a trained neural network are shown underneath. You can play with these examples in this ConvNetsJS demo.

can express more complicated functions

Overfitting

: when a model with high capacity fits the noise(outliers) in the data instead of the (assumed) underlying relationship