Mathematical techniques in data science

Lecture 19: Feed-forward neural networks

April 8th, 2019

Schedule

Week	**Chapter**
1	Chapter 2: Intro to statistical learning
3	Chapter 4: Classification
4	Chapter 9: Support vector machine and kernels
5, 6	Chapter 3: Linear regression
7	Chapter 8: Tree-based methods + Random forest
8	
9	**Neural networks**
12	PCA $\rightarrow$ Manifold learning
11	Clustering: K-means $\rightarrow$ Spectral Clustering
10	Bayesian methods + UQ
13	Reinforcement learning/Online learning/Active learning
14	Project presentation

Recap: Logistic regression

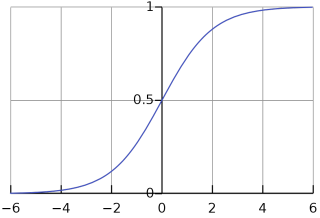
Logistic regression

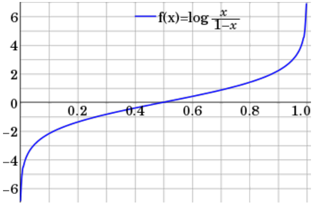
- Suppose we work with binary outputs $\mathcal{Y} = \{0, 1\}$, and $\mathcal{X}$ is a subset of $\mathbb{R}^d$
- Goal: Given input X, we want to model the probability that $Y = 1$

$$P[Y = 1|X = x]$$

Logistic function and logit function

Transformation between $(-\infty, \infty)$ and $[0, 1]$





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

$$logit(p) = \log \frac{p}{1 - p}$$

Logistic regression: Assumptions

Assumption

Given $X = x$, Y *is a Bernoulli random variable with parameter* $p(x) = P[Y = 1|X = x]$ *and*

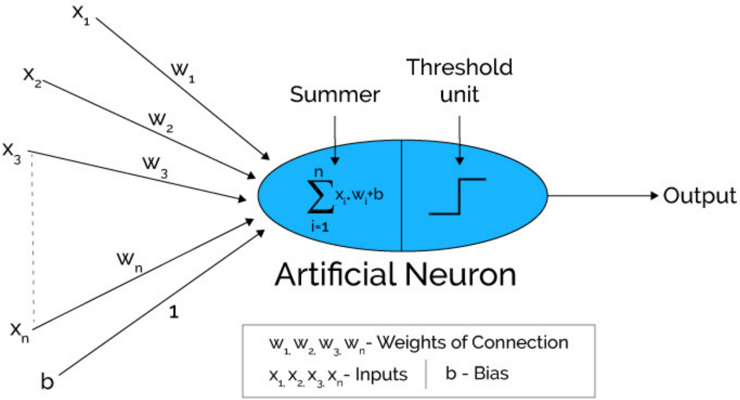
$$logit(p(x)) = \log \frac{p(x)}{1 - p(x)} = \log \frac{P[Y = 1|X = x]}{P[Y = 0|X = x]} = x^T \beta$$

for some vector $\beta \in \mathbb{R}^{d+1}$*.*

Note: Here we denote

$$x^T \beta = \beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d$$

Logistic neuron



Logistic regression with more than 2 classes

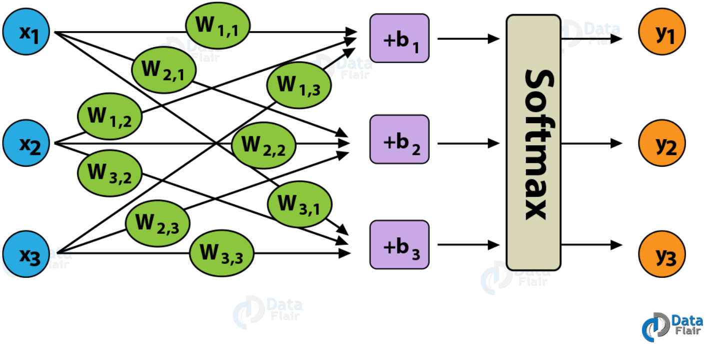
- We use the categorical distribution instead of the Bernoulli distribution

$$P[Y = k|X = x] = p_k(x), \quad \sum_{k=1}^{K} p_k(x) = 1.$$

- Model

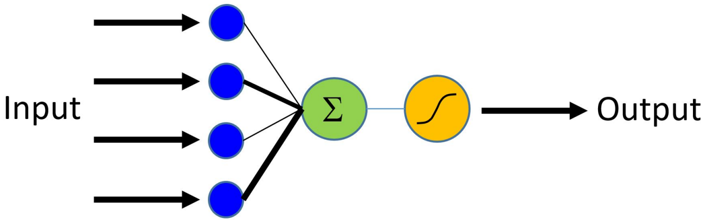
$$p_k(x) = \frac{e^{x^T \beta^{(k)}}}{\sum_{k=1}^{K} e^{x^T \beta^{(k)}}}$$

Logistic regression: Assumptions

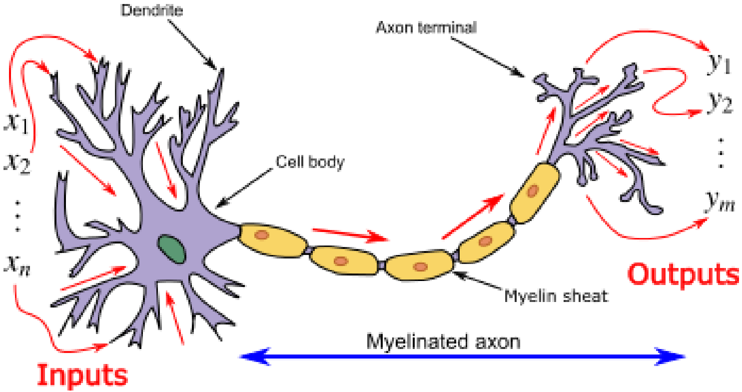


Feed-forward neural networks

Logistic neuron

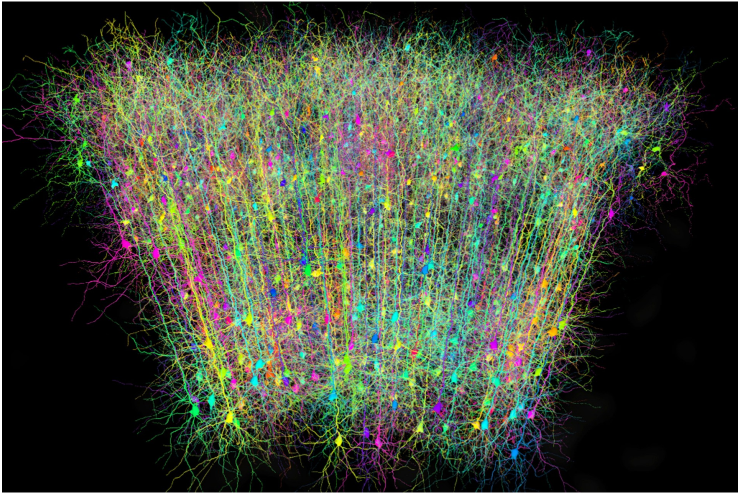


Why neuron?

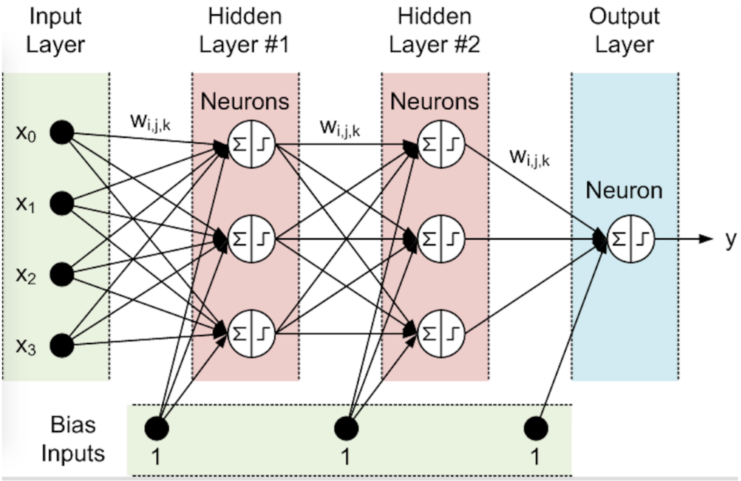


Dendrite
Axon terminal
y_1
y_2
x_1
x_2
x_n
y_m
Cell body
Outputs
Myelin sheat
Myelinated axon
Inputs

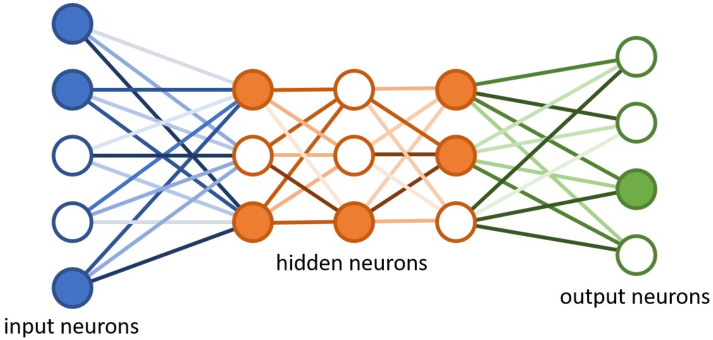
Neural circuit



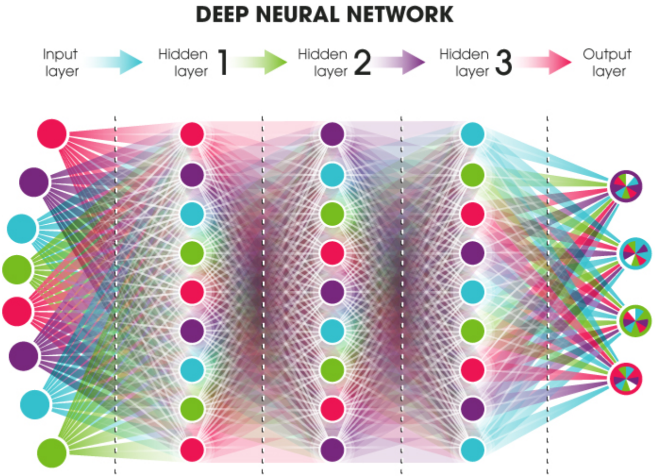
Feed-forward neural networks



Feed-forward neural networks

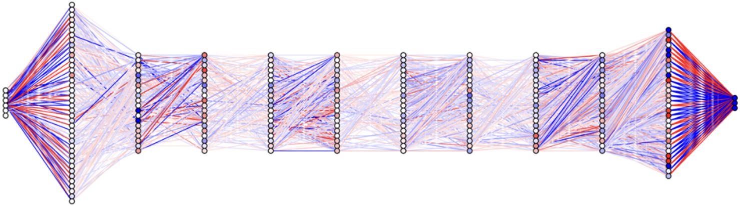


Feed-forward neural networks



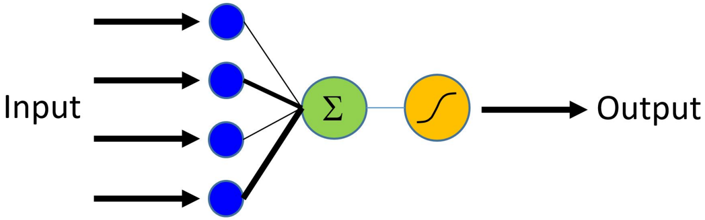
neuralnetworksanddeeplearning.com - Michael Nielsen, Yoshua Bengio, Ian Goodfellow, and Aaron Courville, 2016.

Feed-forward neural networks



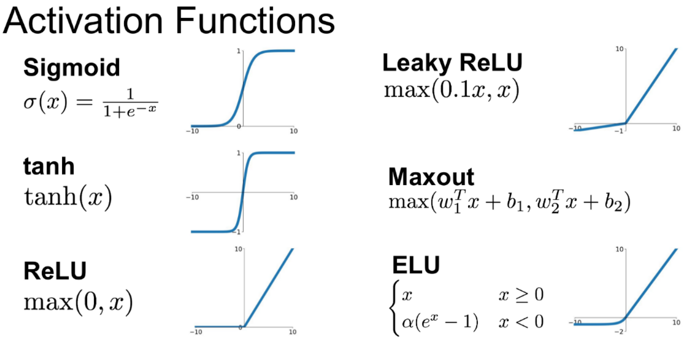
Activation functions

Activation functions



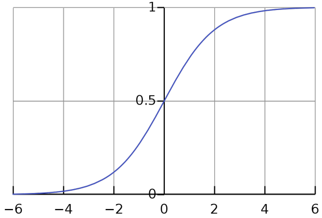
If we do not apply an activation function, then the output signal would simply be a simple linear function of the input signals

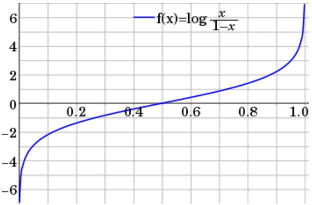
Activation functions



Logistic function (sigmoid function)

Transformation between $(-\infty, \infty)$ and $[0, 1]$

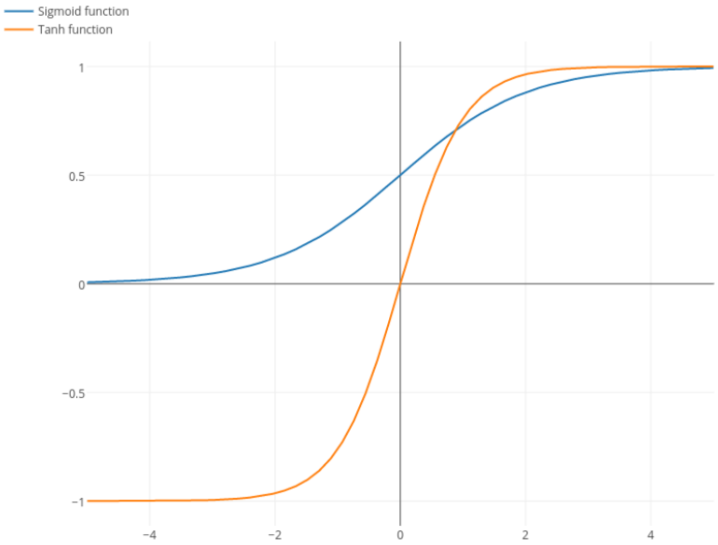




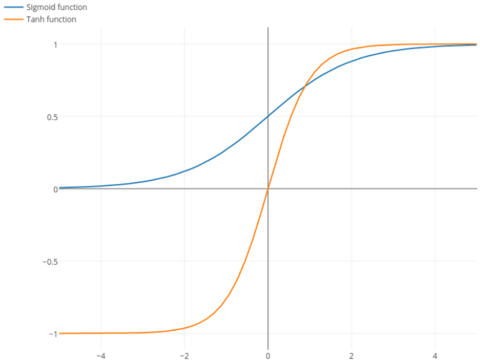
$$f(x) = \frac{e^x}{1+e^x}$$

$$logit(p) = \log \frac{p}{1-p}$$

Hyperbolic tangent

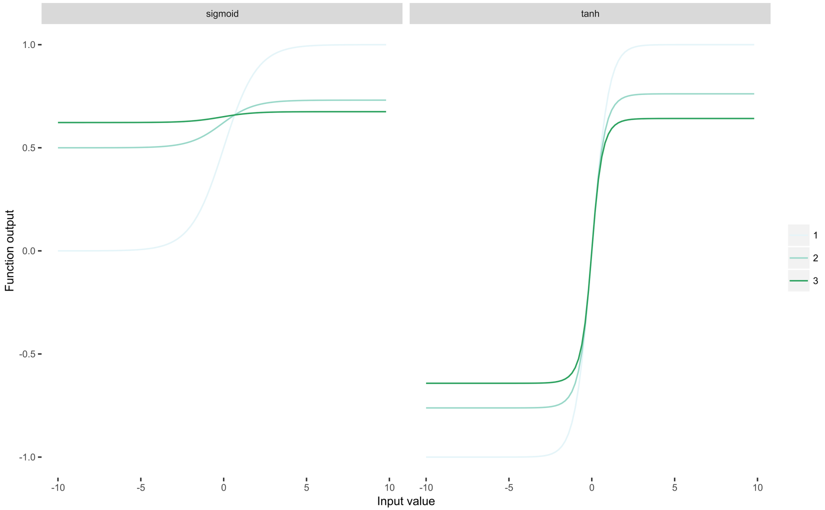


Hyperbolic tangent



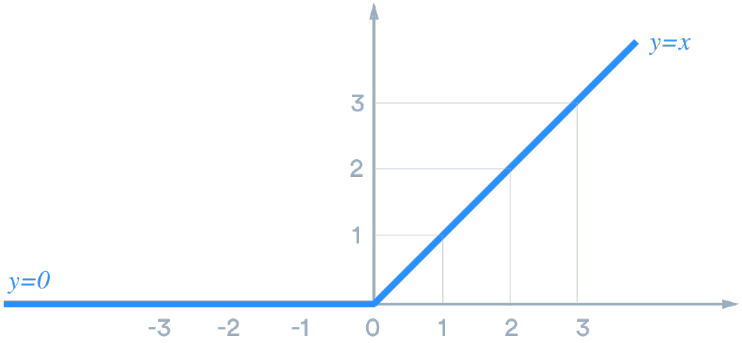
Issue: vanishing gradient problem

Hyperbolic tangent

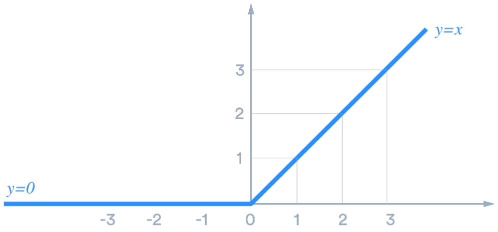


Vanishing gradient problem

Rectified linear unit (ReLU)



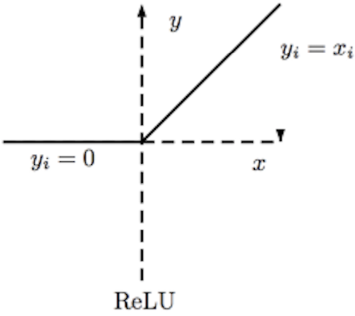
Rectified linear unit (ReLU)

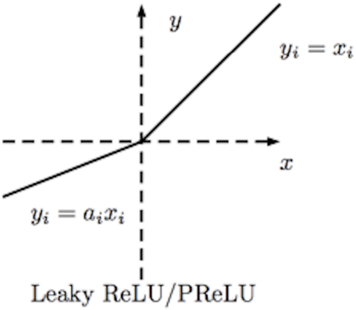


Advantage: model sparsity, cheap to compute (no complicated math), partially address the vanishing gradient problem

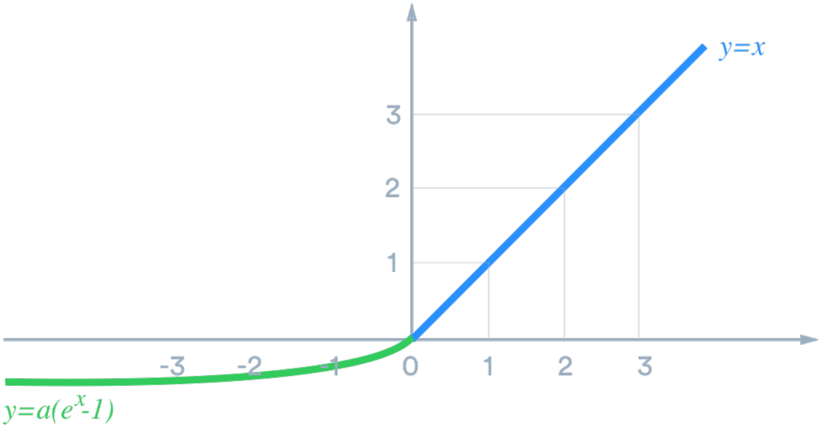
Issue: Dying ReLU

Leaky relu





Exponential Linear Unit (ELU, SELU)



Module: tf.keras.activations

Functions

`deserialize(...)`

`elu(...)`: Exponential linear unit.

`exponential(...)`

`get(...)`

`hard_sigmoid(...)`: Hard sigmoid activation function.

`linear(...)`

`relu(...)`: Rectified Linear Unit.

`selu(...)`: Scaled Exponential Linear Unit (SELU).

`serialize(...)`

`sigmoid(...)`

`softmax(...)`: Softmax activation function.

`softplus(...)`: Softplus activation function.