AI in the Sciences and Engineering HS 2025: Lecture 2

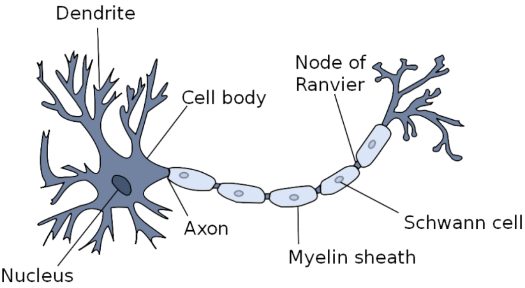
Siddhartha Mishra

Computational and Applied Mathematics Laboratory (CamLab)
Seminar for Applied Mathematics (SAM), D-MATH (and),
ETH AI Center (and) Swiss National AI Institute (SNAI) ,
ETH Zürich, Switzerland.

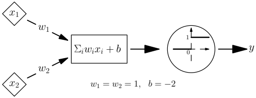
Key Aim of this Course: *Learn Physics modeled by PDEs from data using Neural Networks*

History: McCollough-Pitts 1943

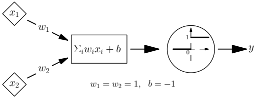
- A biological Neuron.



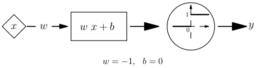
Threshold Logic unit



(a) AND



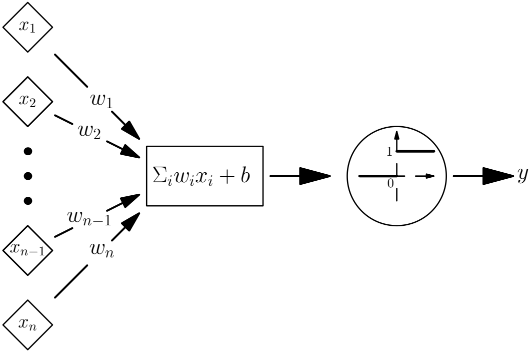
(b) OR



(c) NOT

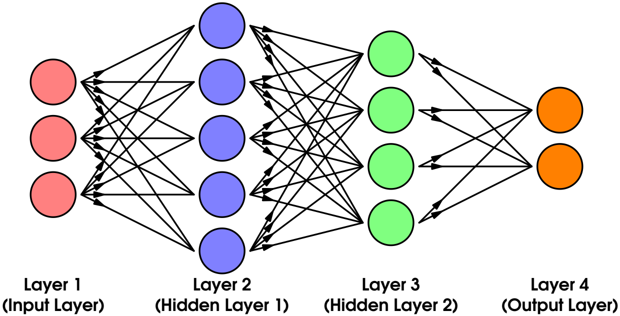
- Weights are Adjustable but NOT Learned

The perceptron of Rosenblatt (1957)

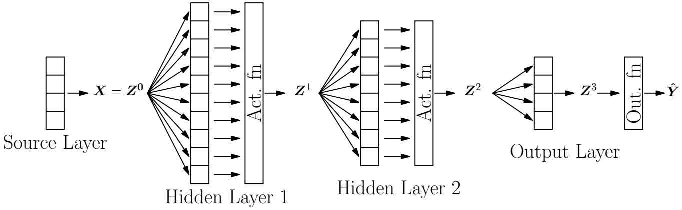


- Uses an activation function.
- Weights are learnable
- Capable of classifying data into 2 classes

Multi-layer Perceptron (MLP) is a direct descendant



Deep Neural network: Multi-layer perceptron



MLP: Basic structure

- Given an input $z \in \mathbb{R}^d$, MLP outputs a $f^*(z) \in \mathbb{R}^o$:

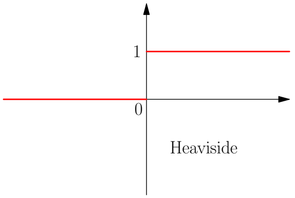
$$f^*(z) = \sigma_o \odot C_K \odot \sigma \odot C_{K-1} \ldots \ldots \ldots \odot \sigma \odot C_2 \odot \sigma \odot C_1(z).$$

- At the k-th Hidden layer:

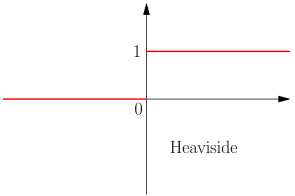
$$\begin{aligned} z^{k+1} &:= \sigma(C_k z^k) \\ &= \sigma(W^k z^k + b^k), \ (W^k, z^k, b^k) \in \left(\mathbb{R}^{d_{k+1} \times d_k}, \mathbb{R}^{d_k}, \mathbb{R}^{d_{k+1}}\right). \end{aligned}$$

- Weights: $W = \{W^k\}_k$, Biases: $B = \{b^k\}_k$.
- Parameters: $\theta = \{W, B\} \in \Theta \subset \mathbb{R}^{\sum\limits_k (d_k+1) d_{k+1}}$.
- Hence, for every $\theta \in \Theta$, MLP returns $f^*_\theta(z)$.

Activation functions σ (scalar–applied component wise)

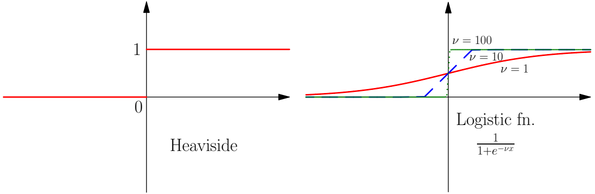


Activation functions σ (scalar–applied component wise)



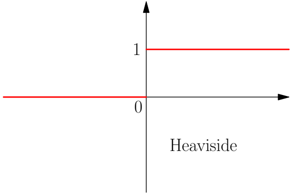
- McCullock-Pitts neuron
- Zero gradient – bad for backpropagation
- Not used anymore

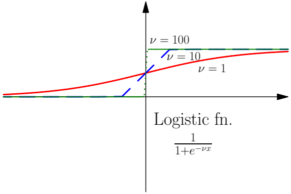
Activation functions σ (scalar–applied component wise)



- McCullock-Pitts neuron
- Zero gradient – bad for backpropagation
- Not used anymore

Activation functions σ (scalar–applied component wise)

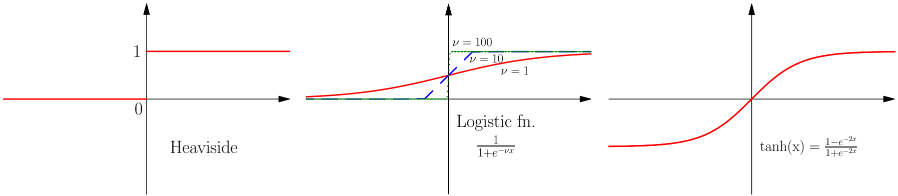




- McCullock-Pitts neuron
- Zero gradient – bad for backpropagation
- Not used anymore

- Smooth approximation to Heaviside func.
- Sigmoidal function – used in most proofs
- Good for binary classification
- Not symmetric

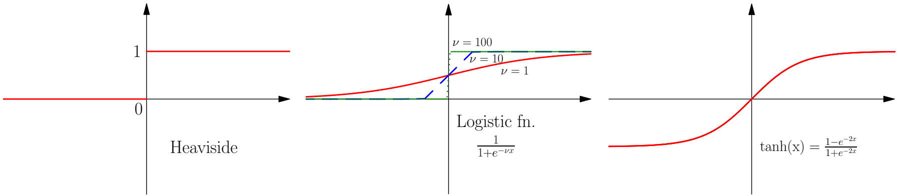
Activation functions σ (scalar–applied component wise)



- McCullock-Pitts neuron
- Zero gradient – bad for backpropagation
- Not used anymore

- Smooth approximation to Heaviside func.
- Sigmoidal function – used in most proofs
- Good for binary classification
- Not symmetric

Activation functions σ (scalar–applied component wise)

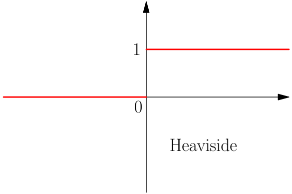


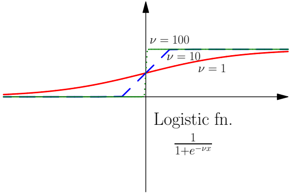
- McCullock-Pitts neuron
- Zero gradient – bad for backpropagation
- Not used anymore

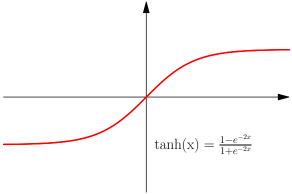
- Smooth approximation to Heaviside func.
- Sigmoidal function – used in most proofs
- Good for binary classification
- Not symmetric

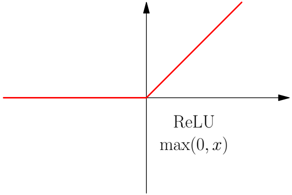
- Symmetric unlike Logisitic func.
- Smooth
- Vanishing gradients away from 0.

Activation functions σ (scalar–applied component wise)

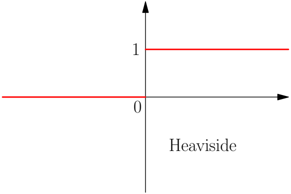


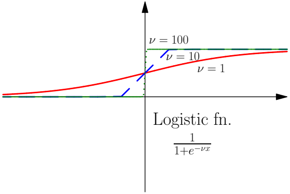


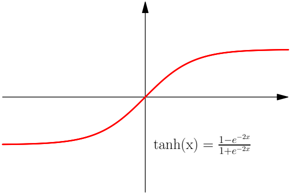


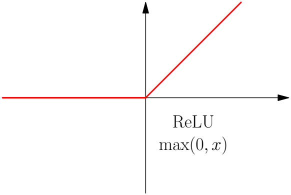


Activation functions σ (scalar–applied component wise)



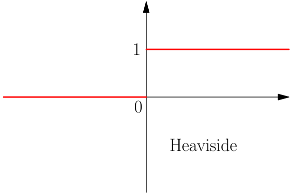


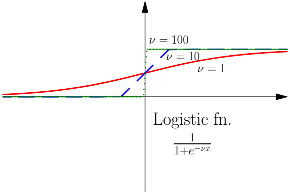


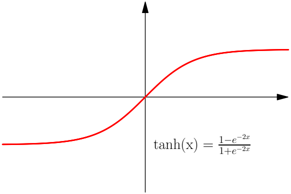


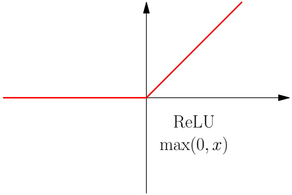
- Easy to compute
- Reduces vanishing gradient problem
- Scale invariant
- Issue of dying neurons

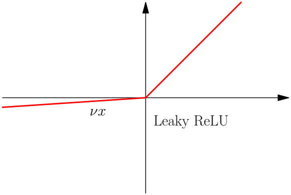
Activation functions σ (scalar–applied component wise)



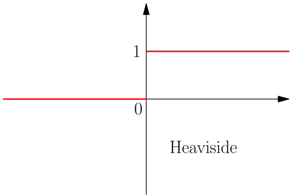


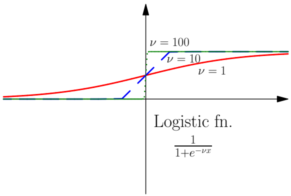


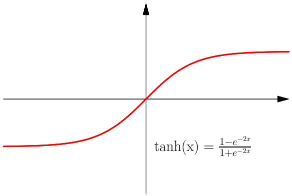


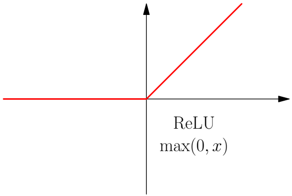


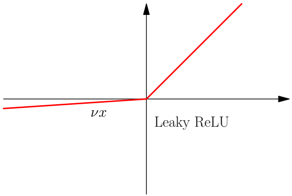
Activation functions σ (scalar–applied component wise)











Any many more ...

MLP

- Given an input $z \in \mathbb{R}^d$, MLP outputs a $f^*(z) \in \mathbb{R}^o$:

$$f^*(z) = \sigma_o \odot C_K \odot \sigma \odot C_{K-1} \ldots\ldots\ldots \odot \sigma \odot C_2 \odot \sigma \odot C_1(z).$$

- At the k-th Hidden layer:

$$\begin{aligned} z^{k+1} &:= \sigma(C_k z^k) \\ &= \sigma(W^k z^k + b^k), \ (W^k, z^k, b^k) \in \left(\mathbb{R}^{d_{k+1} \times d_k}, \mathbb{R}^{d_k}, \mathbb{R}^{d_{k+1}}\right). \end{aligned}$$

- Weights: $W = \{W^k\}_k$, Biases: $B = \{b^k\}_k$.
- Parameters: $\theta = \{W, B\} \in \Theta \subset \mathbb{R}^{\sum\limits_k (d_k+1)d_{k+1}}$.
- Hence, for every $\theta \in \Theta$, MLP returns $f^*_\theta(z)$.

Universal Approximation Theorem

- Given any $f \in C(Y)$, for any tolerance $\epsilon > 0$, there exists parameters $\overline{\theta}$, such that the DNN $f^*_{\overline{\theta}}$ satisfies,

$$\|f - f^*_{\overline{\theta}}\|_{C(Y)} \leq \epsilon$$

- Proved by Cybenko, Barron, Hornik et al., Mhaskar and many more in the late 80's.
- Continuity of the target function can be replaced by Measurability
- But how to efficiently find the parameter $\overline{\theta}$ or an approximation $\widehat{\theta}$ such that:

$$\|f - f^*_{\widehat{\theta}}\|_{C(Y)} << 1.$$

Supervised Learning

- Availability of Labelled Data: (y_i, f_i)
- Input space $Y \subset \mathbb{R}^d$
- $\exists$ an underlying map: $f : Y \mapsto \mathbb{R}$
- Training Set: $\mathcal{S} = \{y_i \in Y\}, 1 \leq i \leq N$
- How to choose training set:
  - Random points: y_i i.i.d with respect to underlying distribution $\mu \in \mathrm{Prob}(Y)$.
  - Other choices might be necessary in some contexts.
- $f_i = f(y_i)$ for all $y_i \in \mathcal{S}$.

Loss Function

- For all $y_i \in \mathcal{S}$, Labelled data $f_i = f(y_i)$.
- For any $\theta \in \Theta$, Evaluate Neural Network to obtain $f^*_\theta(y_i)$
- Loss (Mismatch, Regret) in terms of $f(y_i) - f^*_\theta(y_i)$
- Popular choice of Loss functions:

$$J(\theta) := \frac{1}{N} \sum_{i=1}^{N} \underbrace{|f(y_i) - f^*_\theta(y_i)|_p^p}_{J_i(\theta)}, \quad 1 \leq p < \infty.$$

  - $p = 2 \Rightarrow$ Least Squares minimization.
  - $p = 1 \Rightarrow$ induces sparsity.

Training in Supervised Learning

- Solve the Minimization problem:

$$\theta^* := \arg\min_{\theta \in \Theta} J(\theta)$$

- Trained Neural network

$$f^*(y) = f^*_{\theta^*}(y), \quad \forall y \in Y.$$

Solving the Minimization problem

- The map $\theta \mapsto J(\theta)$ is Non-convex but differentiable (a.e) !!
- Use the Gradient Descent (GD) method:

$$\theta_{\ell+1} = \theta_\ell - \eta_\ell \nabla_\theta J(\theta_\ell), \quad \forall \ell \in \mathbb{N}.$$

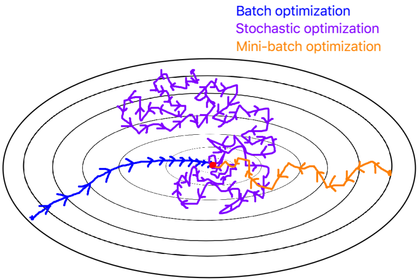
- (Adaptive) Learning rate η_ℓ.
- GD converges to a local minimum θ_* !!
- Issues in computing gradients: $\nabla_\theta J(\theta_\ell) = \sum\limits_{i=1}^{N} \nabla_\theta J_i(\theta_\ell)$
- As $N = \#(\mathcal{S})$, for Large training data sets:
  - Gradient computation is too slow.
  - Requires large memory

Stochastic Gradient Descent (SGD)

- At ℓ-th iterate of GD: choose $i_\ell \in (1, N)$ randomly and set:

$$\theta_{\ell+1} = \theta_\ell - \eta_\ell \nabla_\theta J_{i_\ell}(\theta_\ell), .$$

- Converges (convergence theory based on underlying SDE.)
- Fast (per iteration) but has high variance (noisy convergence) !!



Mini-Batch SGD

- Randomly shuffle training set $\mathcal{S}$ into Batches $\mathcal{S}_j$, each of size n.
- SGD iteration:

$$\theta_{\ell+1} = \theta_\ell - \eta_\ell \sum_{j \in \mathcal{S}_j} \nabla_\theta J_{i_\ell}(\theta_\ell).$$

- With N/n iterations, we stride over the training set to complete 1 Epoch.
- Reshuffle after each epoch.
- Standard (Full-Batch) GD: $n = N$
- SGD: $n = 1$.

Starting values for the optimizer

- Customary to use Random starting value $\theta_0 \in \Theta$.
- Heuristic scaling to minimize variance of the weights.
- Depends on activation functions
- Different $\theta_0 \Rightarrow$ SGD converges to different local minima.
- Also customary to use Multiple starting values in parallel (Retrainings)

How to assess training success ?

- Monitor training loss.
- If not low enough (Underfitting):
  - Change Network Size.
  - Train more
  - Change Architecture.
- But Goal is to reduce Generalization error:

$$\mathcal{E}_G := \int\limits_Y |f(y) - f^*(y)|_p^p d\mu(y).$$

- Error on Unseen data.

Validation Set

- Let $\mathcal{V} \subset Y$ with $\mathcal{V} \cap \mathcal{S} = \Phi$ be Validation test.
- Evaluate Validation loss:

$$\mathcal{E}_{\rm val} := \frac{1}{\#(\mathcal{V})} \sum_{y \in \mathcal{V}} |f(y) - f^*(y)|_p^p.$$

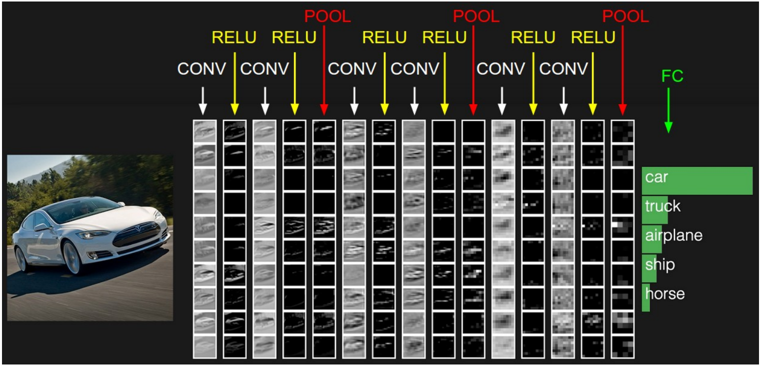
- $\#(\mathcal{V})$ is $5 - 10\%$ of $\#(\mathcal{S})$
- Sacrifice some training data to form Validation set.

Beyond MLP: CNNs

- For Fully connected networks (MLP) – very large sizes for Θ if $dim(Y) >> 1$
- CIFAR-10 image: Input vector is in $\mathbb{R}^{3072} \Rightarrow \theta \in \mathbb{R}^M$ for $M >> 1$
- Induce some sparsity structure on the weight matrices W^k: Discrete convolutions $\Rightarrow$ Convolutional Neural Networks
- Discrete Convolutions with fixed Kernel:

$$K_c[m] = \sum_{i=-s}^{s} k_i c[m-i]$$

Convnets


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse