SMAI-S25-L20:Discussions on MLP and BP

C. V. Jawahar

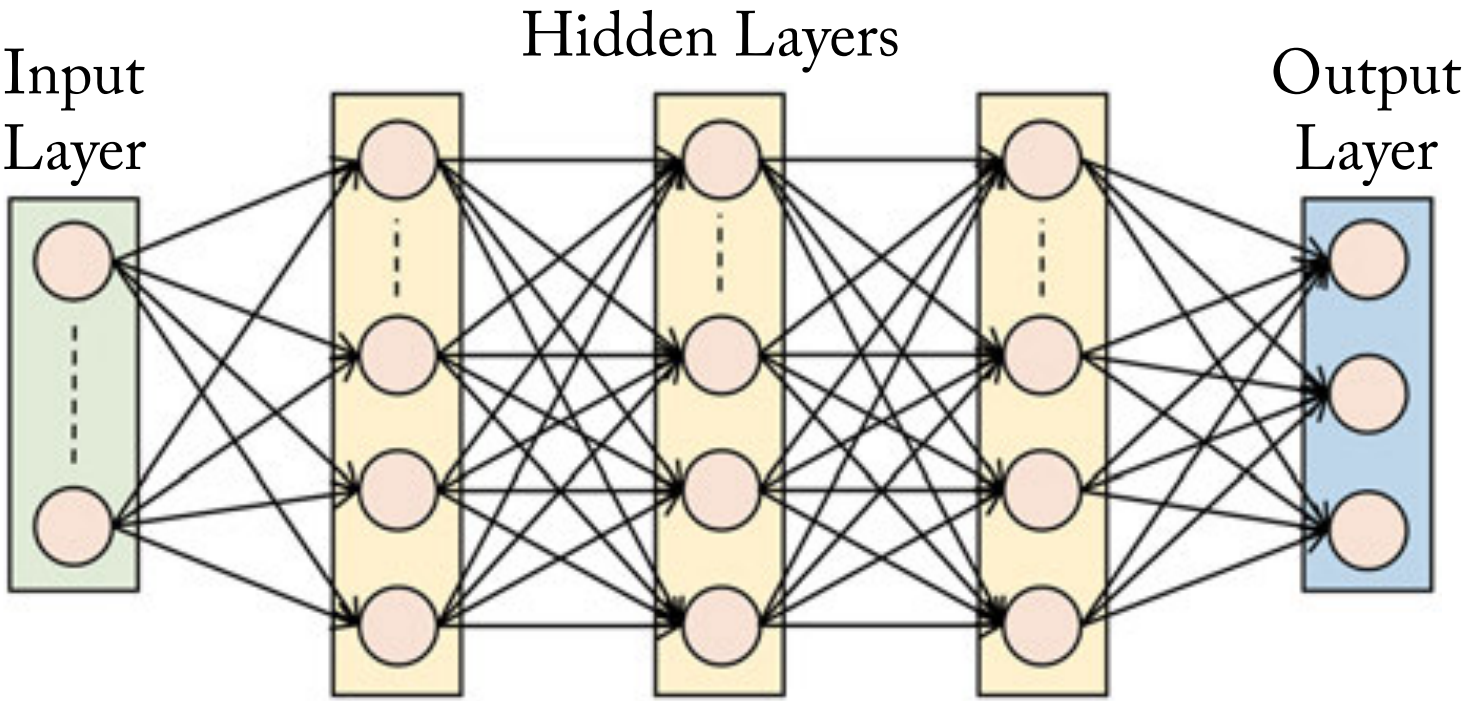
IIIT Hyderabad

April 1, 2025

Recap:

- Problems of interest:
  - Learn a function $y = f(\mathbf{W}, \mathbf{x})$ from the data: (a) Classification (b) Regression
  - Learn Feature Transformations $\mathbf{x}' = \mathbf{W}\mathbf{x}$ or $\mathbf{x}' = f(\mathbf{W}, \mathbf{x})$: (a) Feature Normalization (b) PCA
- Algorithms/Approaches:
  - Nearest Neighbour Algorithm
  - Linear Classification: $sign(\mathbf{w}^T\mathbf{x})$
  - Decide as ω_1 if $P(\omega_1|\mathbf{x}) \geq P(\omega_2|\mathbf{x})$ else ω_2.
  - Linear Regression: (a) closed form and (b) GD
  - Logistic Regression, Naive Bayes, SVM, Kernel (intro)
  - Perceptron, GD, MLP
- Supervised Learning:
  - Notion of Training, Validation and Testing; Performance Metrics
  - Data and Tools: Low rank data matrix, Distribution of Data; SVD
  - Notion of Loss Function, (eg. MSE), Regularization.
  - Role of Optimization, Convex and non-Convex optimization
  - Closed form solution, Gradient Descent, Eigen vector solns.
  - MDL, Model complexity, Overfitting

MLP



MLP

A typical MLP has

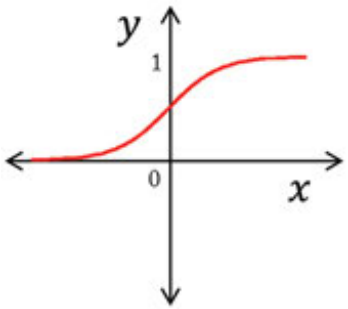
- **Nodes and Edges** Network has nodes (where simple computations take place and edges where information flow happen.
- **Layered architecture.** Network is understood and represented in layers.
- **Dense Connections** Successive layers are often fully connected.

Q1: Why do we need hidden layers?
Q: Why do we need activation functions (nonlinearities)?

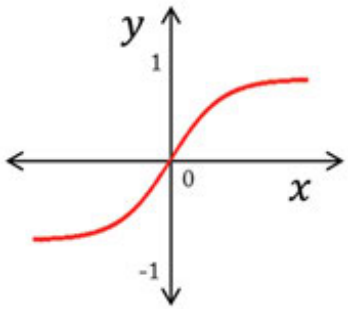
Question

- Draw networks that can solve AND, OR, EXOR with 0/1 logic. (assume neurons have a threshold activation. $\phi(x) = 1$ if $x \geq 0$)

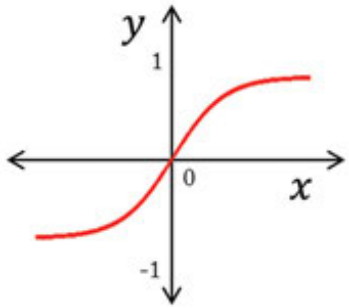
Activations



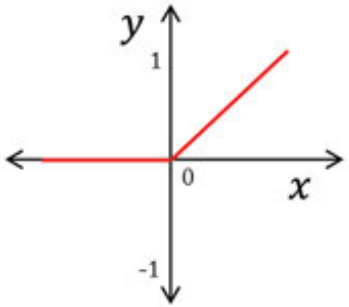
(a) Sigmoid



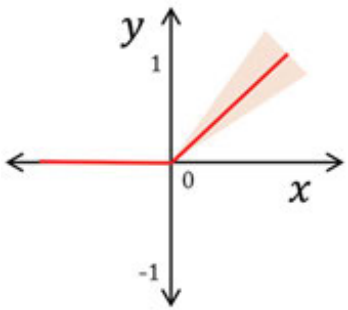
(b) Tanh



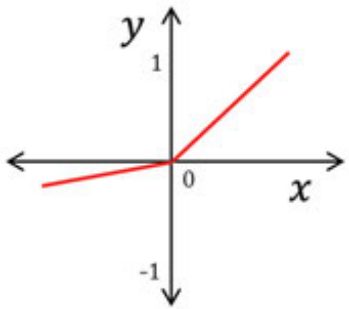
(c) Algebraic Sigmoid



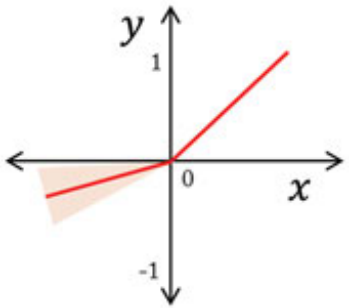
(d) ReLU



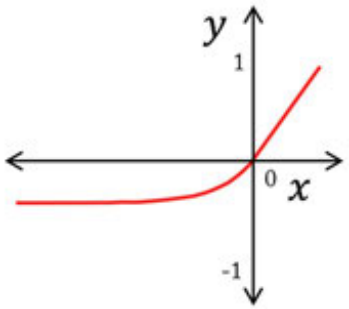
(e) Noisy ReLU



(f) Leaky ReLU/PReLU



(g) Randomized Leaky ReLU



(h) Exponential Linear Unit

Activations

1. **Sigmoid**

$$y = \frac{1}{1 + e^{-x}}$$

2. **tanh**

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

3. **ReLu**

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

4. **Leakly Relu**

$$y = \begin{cases} x & \text{if } x > 0 \\ cx & \text{if } x \leq 0 \end{cases}$$

5. **Noisy ReLu**

$$y = \max(0, x + \epsilon); \epsilon = \mathcal{N}(0, \sigma(x))$$

6. **Exponential Linear Unit**

$$y = \begin{cases} x & \text{if } x > 0 \\ a(e^{x} - 1) & \text{if } x \leq 0 \end{cases}$$

Typical MLP

- Not very deep
- Used as classifier (also regressor) with handcrafted small features.
- Expends features before classification.

What is the role of Hidden Layers?

- computing and enriching features or new representations.

MLP for Regression

- In many cases, the expected output need not be in $[0, 1]$ or $[-1, +1]$. It is advisable to use a linear activation ($\phi(x) = x$) in such cases.

MLP for Classification

- Argmax to softmax: $\frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$
- Cross entropy:

$$\mathcal{L}(\mathbf{o}, \mathbf{t}) = \sum_{i=1}^{K} t_i \log(o_i)$$

If $\mathbf{t}$ and $\mathbf{o}$ are probability distributions (desired and output), then the cross entropy loss is equivalent to the KL divergence between these two distributions.

Recap of Back Propagation

1. Initialize the network with random weights.
2. For all the samples, compute the output of the neural network $\mathbf{x}_{p+1}$
3. Compute the loss for the full batch of N samples as

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_{p+1}^{i}, \mathbf{y}_i)$$

4. Adjust all the parameters (such as weight matrices) as

$$\theta^{k+1} \leftarrow \theta^{k} - \Delta\theta$$

or

$$\theta^{k+1} \leftarrow \theta^{k} - \eta \frac{\partial L}{\partial \theta}$$

5. Repeat steps 2-4 until convergence.

Challenges in Training NNs

- Non-Convex Optimization
- Many parameters to Control

Why simple GD works?

- Second order methods (that needs Hessian matrix) is compute intensive
- Nature of Loss Function (simultaneously all gradients vanish?)
- Multiple effective solutions

Refinements over BP

Over years, backpropagation algorithm has been refined significantly with many minor but critical innovations. What all can change in the simple version we saw early?

1. **step 1:** Initialization can be smarter.
2. **step 2:** Computing loss over a full batch and updating it once is not the best.
3. **step 3:** Loss function can be different. There re many other loss functions available beyond MSE.
4. **step 4:** Update rule can be different. What we saw here is too simple.

Refinements over BP

- Better Initialization
- Effective Gradient Estimates
- Better Update Rule: Momentum

$$\theta^{k+1} \leftarrow \theta^k - \Delta\theta$$

$$\Delta\theta = \eta \frac{\partial L}{\partial \theta}$$

$$\Delta\theta = \eta \frac{\partial L}{\partial \theta} + \gamma \Delta\theta^{t-1}$$

- Choice of Optimizer (such as Adam (adaptive momentum estimation))
- Loss Function
- Termination

Regularization

- L1/L2 regularization
- Data Augmentation
- Drop out
- Normalization (including in the middle of the network)
- etc.

Questions?