

Foundations of Machine Learning

AI2000 and AI5000

FoML-22

Neural Networks - learning the basis functions

Dr. Konda Reddy Mopuri

Department of AI, IIT Hyderabad

July-Nov 2025



భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad



So far in FoML

- Intro to ML and Probability refresher
- MLE, MAP, and fully Bayesian treatment
- Supervised learning
 - a. Linear Regression with basis functions (regularization, model selection)
 - b. Bias-Variance Decomposition (Bayesian Regression)
 - c. Decision Theory - three broad classification strategies
 - Probabilistic Generative Models - Continuous & discrete data
 - (Linear) Discriminant Functions - least squares solution, Perceptron
 - Probabilistic Discriminative Models - Logistic Regression

Neural Networks - I



So far in the supervised learning

- We have seen models that are linear combinations of fixed basis functions

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$



So far in the supervised learning

- Fixed basis functions
 - Don't scale easily to complex settings (e.g., curse of dimensionality)
 - Need to adapt the basis functions to the data - e.g., SVM

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$



An alternative

- Fix the number of basis functions ✓
- But, allow them to be adaptive
- → parametric form and learn them during the training

(Artificial) Neural Networks



Feed-forward Neural networks

- The linear models we have seen so far = linear combination of fixed nonlinear basis functions
 - What is f for regression and classification?

$$y(\mathbf{x}, \mathbf{w}) = f \left(\sum_{j=1}^M w_j \phi_j(\mathbf{x}) \right)$$



Feed-forward Neural networks

- Let's extend this model
- Make the basis functions depend on 'learnable' parameters

$$y(\mathbf{x}, \mathbf{w}) = f \left(\sum_{j=1}^M w_j \underline{\phi_j(\mathbf{x})} \right)$$

Feed-forward Neural networks

- Neural networks - each basis function is a nonlinear function of a linear combination of inputs
 - Coefficients in the combination are adaptive parameters

$$y(\mathbf{x}, \mathbf{w}) = f \left(\sum_{j=1}^M w_j \phi_j(\mathbf{x}) \right)$$

Feed-forward Neural networks

- Basic neural network model as a series of transformations
 - M linear combinations of the i/p variables x_1, x_2, \dots, x_D

$$\underline{x} \in \mathbb{R}^D$$

$$a_j = \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

$j = 1, \dots, M$, and the superscript 1 indicates the first transformation or 'layer'

Feed-forward Neural networks

- a_j are referred to as the 'activations'
- Then passed through a differentiable, nonlinear activation function $h(\cdot)$

$$\underline{a_j} = \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

$$\underline{z_j} = \underline{h}(\underline{a_j})$$

$j = 1 \text{ to } M$

j^{th} hidden function
 j^{th} hidden units



Feed-forward Neural networks

- z_j are the o/ps of the basis functions
 - referred to as the 'hidden units'
 - General nonlinear functions - sigmoid/tanh/ReLU

$$a_j = \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

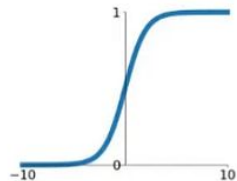
$$z_j = h(a_j)$$



Activation Functions

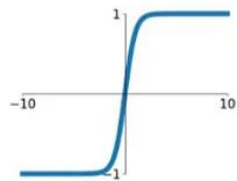
Sigmoid

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



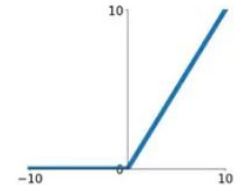
tanh

$$\tanh(x)$$



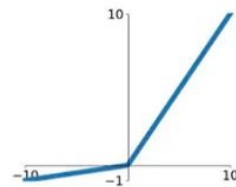
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

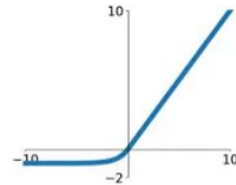


Maxout

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

ELU

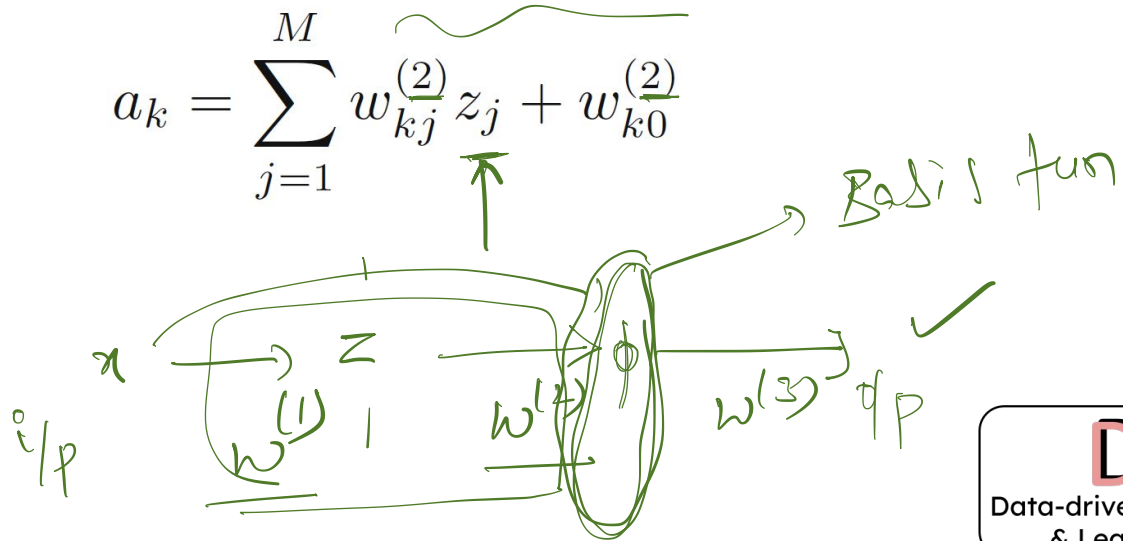
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Feed-forward Neural networks

- These values are again linearly combined to give o/p units
 - This transformation - second layer of the NN

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$



Feed-forward Neural networks

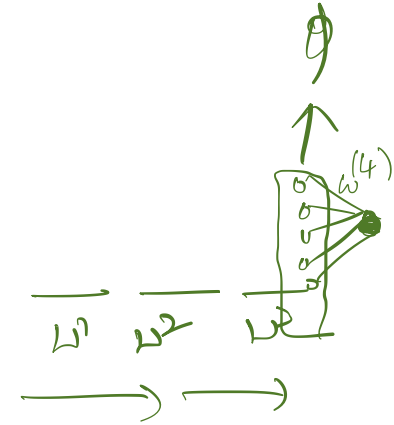
- Finally, the o/p unit activations are transformed through an appropriate activation function $\rightarrow y_k$

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} \underline{z_j} + w_{k0}^{(2)}$$

Feed-forward Neural networks

- Regression - identity ($y_k = a_k$) ✓
- Binary classification - sigmoid ($y_k = \sigma(a_k)$)
- Multiclass classification - softmax

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$



$(w^4 \phi)$



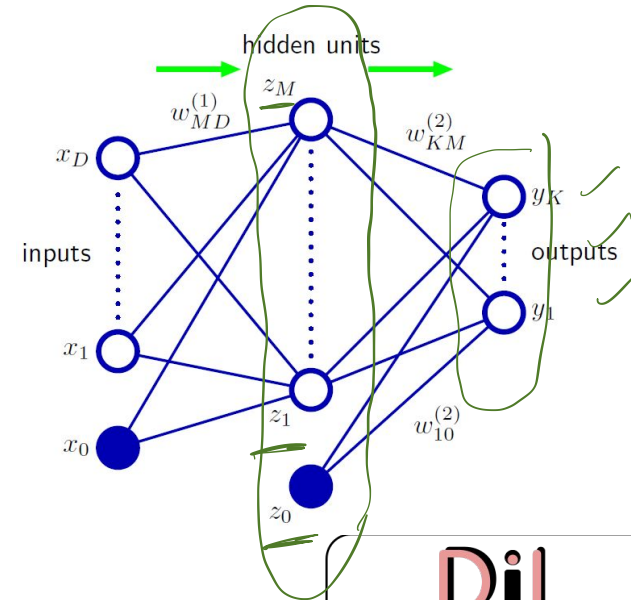
Feed-forward Neural networks

- Overall

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=1}^M w_{kj}^{(2)} h \left(\underbrace{\sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}}_{\text{layer 1}} \right) + \underbrace{w_{k0}^{(2)}}_{\text{layer 2}} \right)$$

softmax for multi class classification y_i -

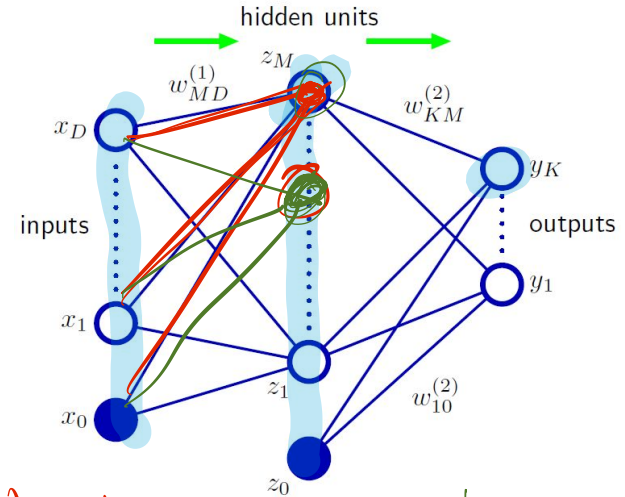
$$P(L_k | \mathbf{x}) = \frac{e^{y_k}}{\sum_{i=1}^K e^{y_i}}$$



Feed-forward Neural networks

- All the weights & biases are grouped into \mathbf{w}
 - Adjustable parameters
- Model is a nonlinear function from \mathbf{x} to \mathbf{y}
 - Figure: 2-layered MLP (Multilayer Perceptron)

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=1}^M w_{kj}^{(2)} h \left(\sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$



$\mathbf{x} \in \mathbb{R}^D$

MLP

$(d+1) \leftarrow w_1^{(1)} \quad w_2^{(1)} \rightarrow (d+1)$



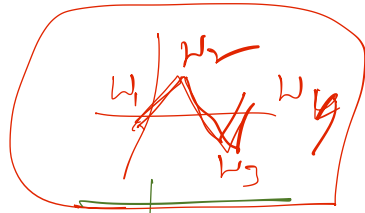
Feed-forward Neural networks

- What if the activation function on all the hidden units are linear?
 - An equivalent network without any hidden units can be found
 - Why? ✓
 - Rare - e.g., PCA

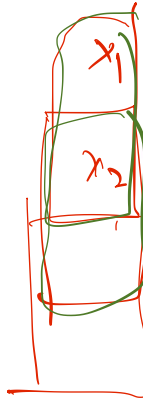


Feed-forward Neural networks

- Networks can be sparse ✓
 - Not all weights may be present

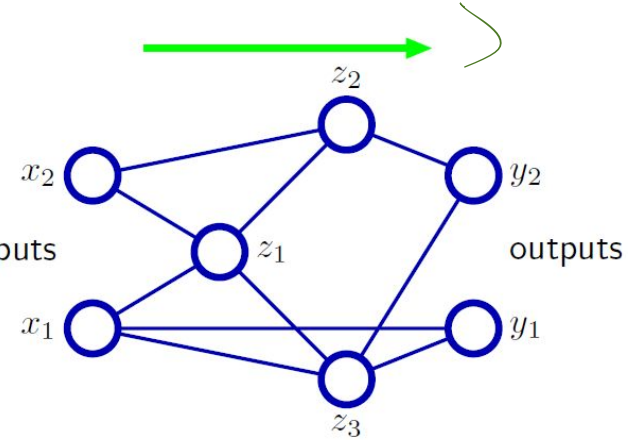


CNNs benefit with parameter sharing



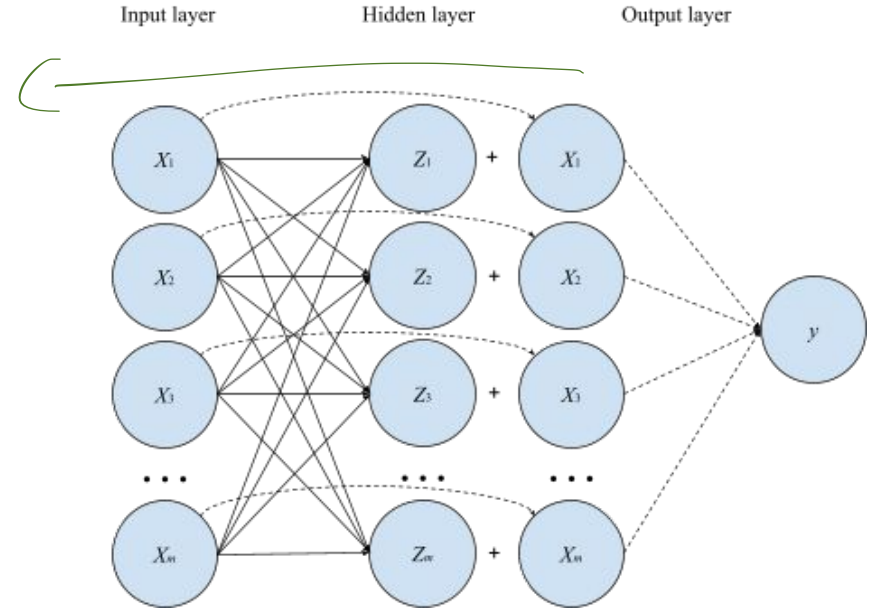
x_{L+K}
 a_L

x_{L+K}



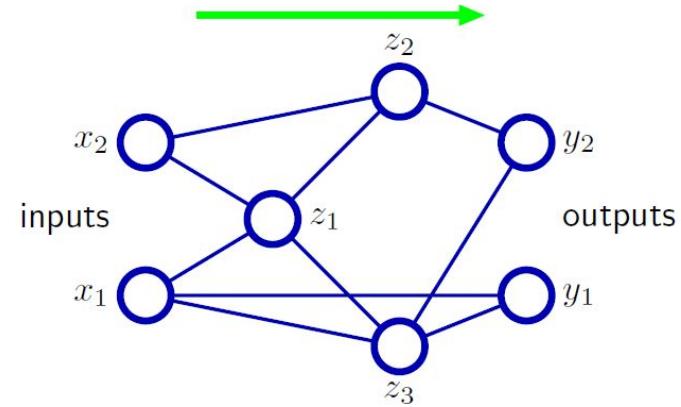
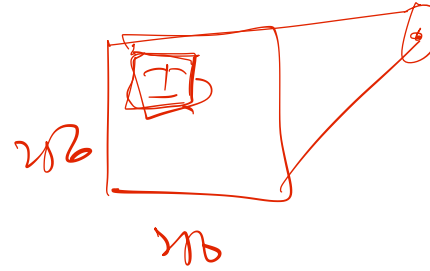
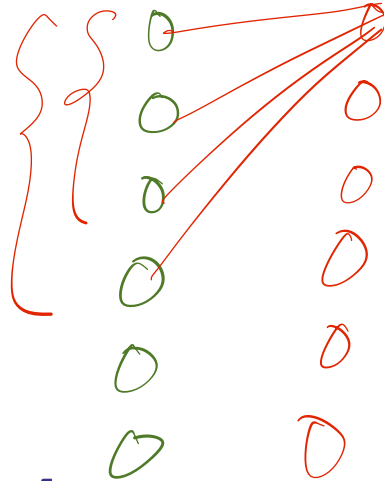
Feed-forward Neural networks

- Skip connects may be added



Feed-forward Neural networks

- FFNN - no closed directed cycles
 - o/p are deterministic functions of i/p's



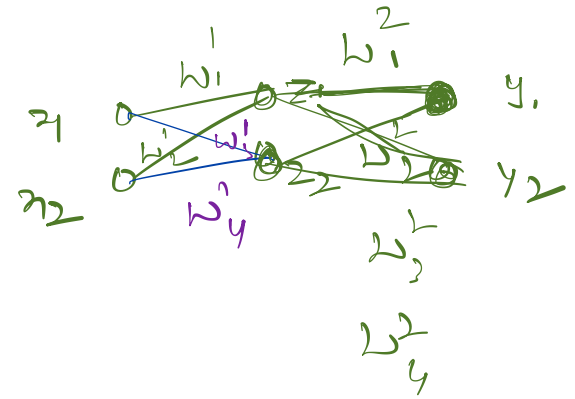
Feed-forward Neural networks

- Approximation capabilities of these networks have been well-studied
 - These networks are very general

$$z_1 \begin{bmatrix} w_1^1 \\ w_2^1 \end{bmatrix}$$

$$z_2 \begin{bmatrix} w_1^1 \\ w_2^1 \end{bmatrix}$$

$$y = w_1^2 \textcircled{1} + w_2^2 \textcircled{2}$$



Next

Training the Neural Networks