

Machine Learning (CE 40717)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

October 14, 2024



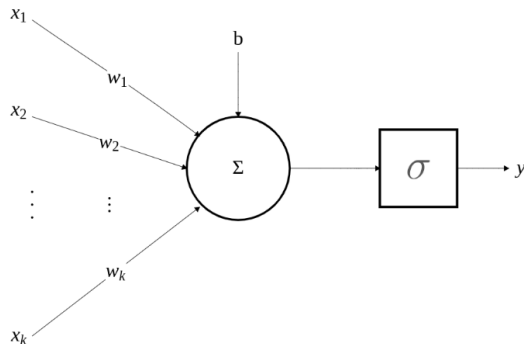
- ① Introduction
- ② Multi-Layer Perceptron (MLP)
- ③ Neural Networks
- ④ Training Neural Networks
- ⑤ References

- ① Introduction
- ② Multi-Layer Perceptron (MLP)
- ③ Neural Networks
- ④ Training Neural Networks
- ⑤ References

Perceptron Reminder

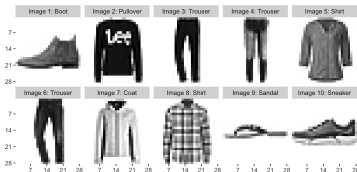
The building block of each neural network is Perceptron:

- $\{x_1, x_2, \dots, x_k\}$: input features
- $\{w_1, w_2, \dots, w_k\}$: feature weights
- b : bias term
- $\sigma(\cdot)$: activation function
- y : output of the neuron



Why Neural Networks?

- We can find explicit formulas for some problems (no machine learning)
 - $\Delta x = \frac{1}{2} a \cdot t^2 + v_0 \cdot t$
- We can model some problems assuming simple relationships (classical machine learning)
 - House price as a linear function of its features
 - $y = a_1 \cdot x_1 + a_2 \cdot x_2 + \dots + a_p \cdot x_p$
- How about classifying these images?



Why Neural Networks? Cont.

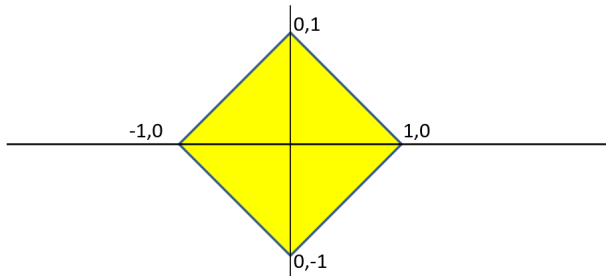
- No explicit formula exists to recognize a sneaker
- We recognize any sneaker intuitively
- Our brains use a complex function for this recognition
- **Deep neural networks** can learn this complex function



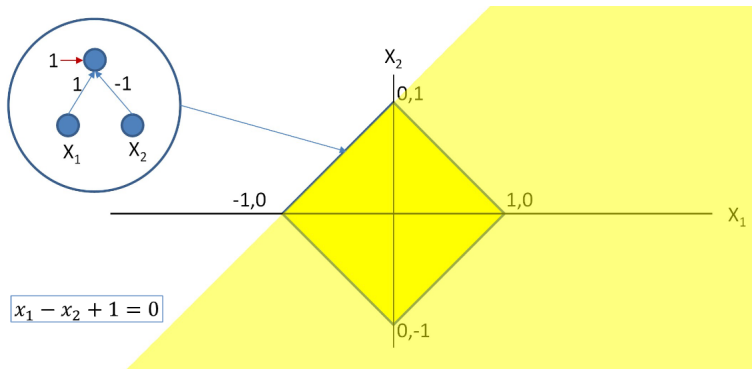
- ① Introduction
- ② Multi-Layer Perceptron (MLP)
- ③ Neural Networks
- ④ Training Neural Networks
- ⑤ References

Example: MLP for Complex Patterns

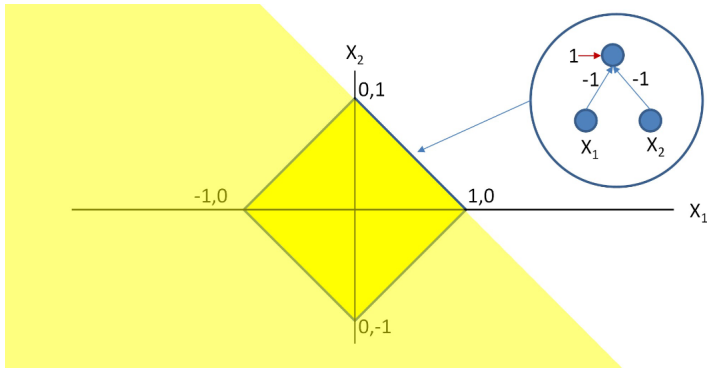
- What network to learn this area?
- Example is adapted from [1].



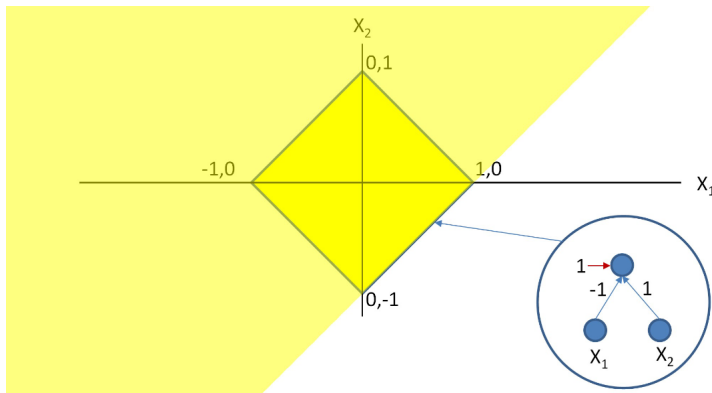
Example: MLP for Complex Patterns Cont.



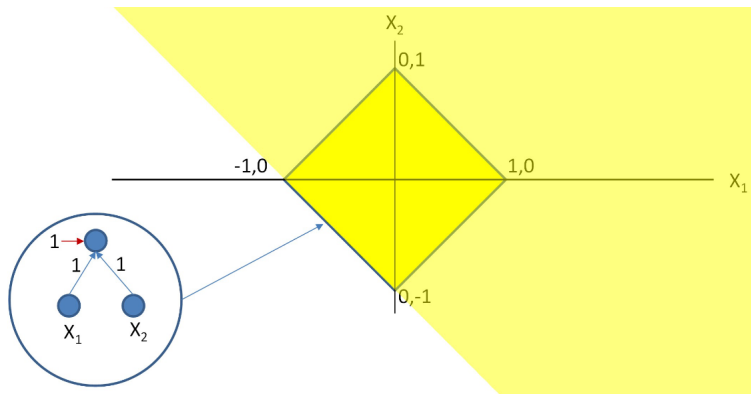
Example: MLP for Complex Patterns Cont.



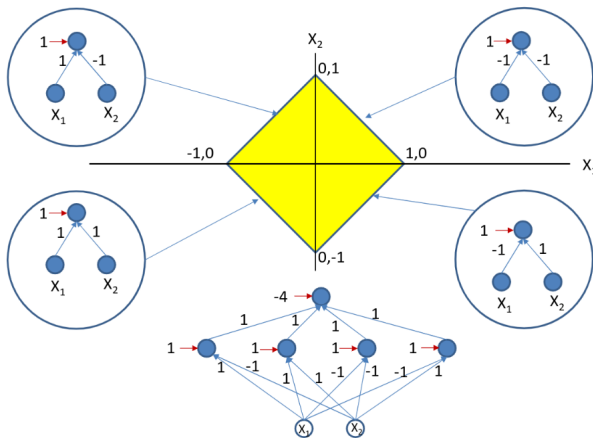
Example: MLP for Complex Patterns Cont.



Example: MLP for Complex Patterns Cont.

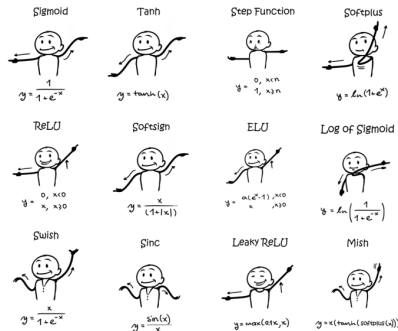


Example: MLP for Complex Patterns Cont.



MLP Capacity

- Increasing **width and depth** allows us to approximate **complex decision boundaries**
- An **activation function** makes a neuron's output **non-linear**, allowing the network to learn complex data
- Not limited** to Boolean or step functions
- With appropriate activation functions, neural networks can **approximate any real-valued function** (More details later)



Adapted from Sefiks

- 1 Introduction
- 2 Multi-Layer Perceptron (MLP)
- 3 Neural Networks**
- 4 Training Neural Networks
- 5 References

Single Hidden Layer Neural Network

- Hidden layer pre-activation:

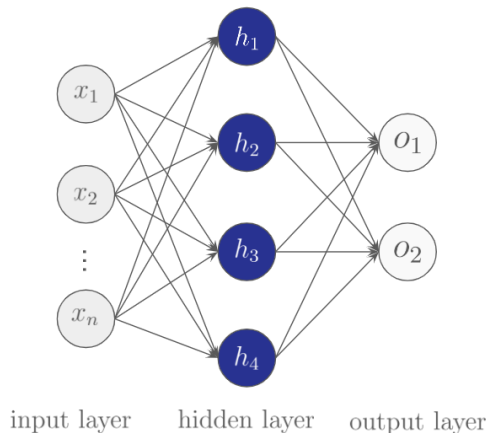
$$a_i(x) = b_i^{(1)} + \sum_j W_{ij}^{(1)} \cdot x_j$$

- Activated hidden layer:

$$h(x) = \sigma(a(x))$$

- Output layer:

$$o(x) = o(b^{(2)} + W^{(2)} h^{(1)}(x))$$



Multi-Hidden Layer Neural Network

- Let $h_i^0 = x_i$ for $i \in \{1, 2, \dots, n\}$
- For $\ell \in \{0, 1, \dots, L\}$:

$$a_j^{(\ell+1)} = b_j^{(\ell)} + \sum_i W_{ij}^{(\ell)} \cdot h_i^{(\ell)}$$

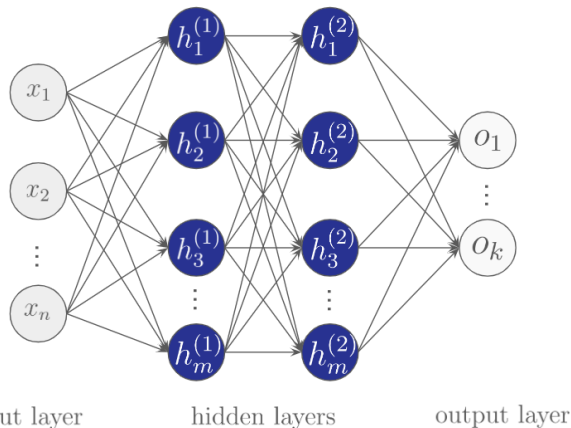
$$h_j^{(\ell+1)} = \sigma(a_j^{(\ell+1)})$$

- Learnable parameters:

$$b_j^{(\ell)}, W_{ij}^{(\ell)}$$

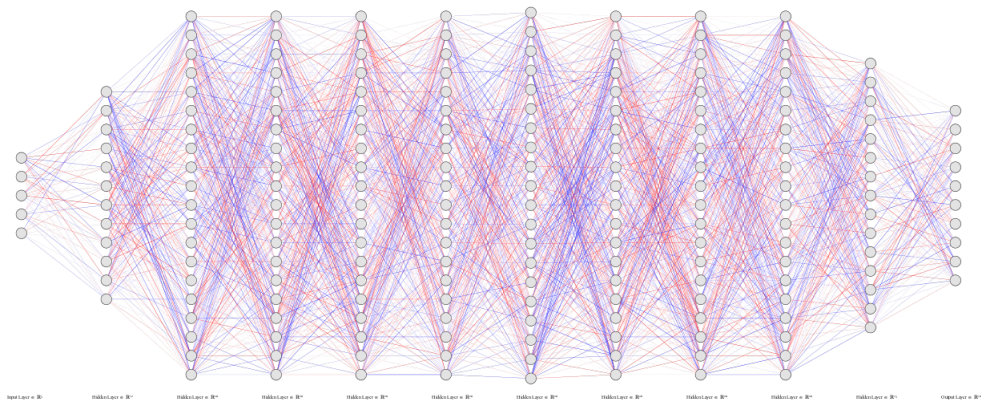
- Number of learnable parameters:

$$(n+1)m_1 + (m_1+1)m_2 + \dots + (m_L+1)k$$



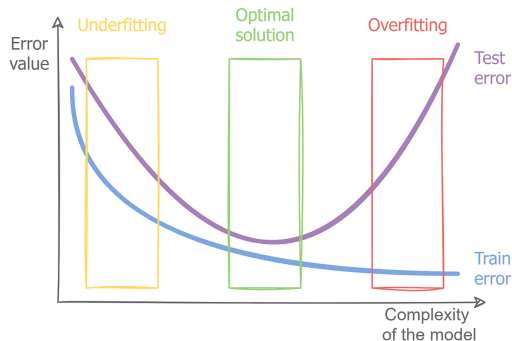
Deep Neural Network Architecture

- More than a few hidden layers: **Deep Neural Network (DNN)**
- Designing a neural network architecture is **more of an art than a science**.



Network Width and Depth

- **Width:** More neurons, more complexity
- **Depth:** More layers, more abstraction
- **Balance:**
 - Too narrow/shallow: risk of underfitting
 - Too wide/deep: risk of overfitting

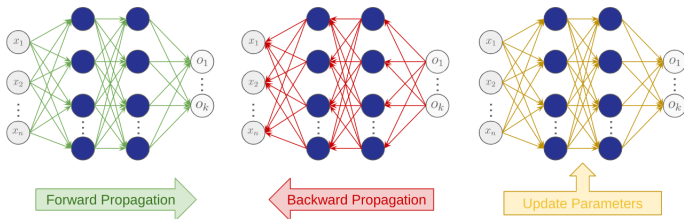


Adapted from Towards Data Science

- ① Introduction
- ② Multi-Layer Perceptron (MLP)
- ③ Neural Networks
- ④ Training Neural Networks
- ⑤ References

Training Phases

- **Initialize weights and biases:** These values control how the network initially processes information (More details later)
- **Forward pass:** Pass the input through the network to get an output
- **Calculate the error:** Compare the network's output to the correct answer to measure the difference (called the 'loss' - More details later)
- **Backpropagation:** Use the loss value to adjust the weights and biases to improve the network's accuracy

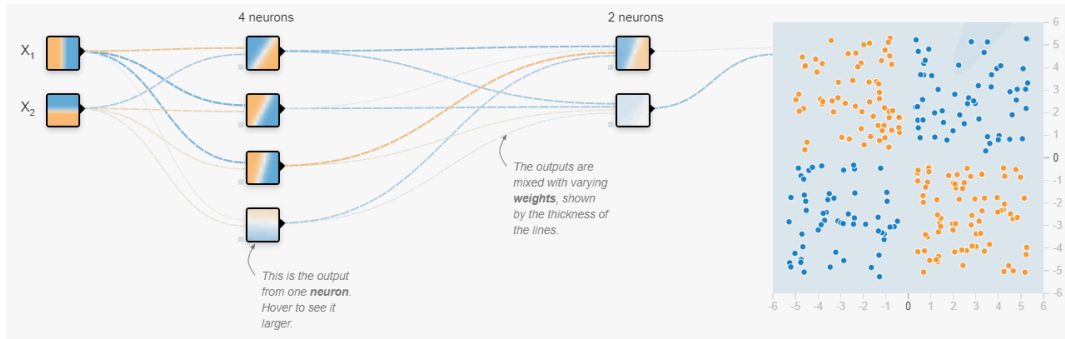


Forward Propagation

- This is the pass where we send input data through the network to make a prediction (likely inaccurate at first).
- The prediction is made by calculating weighted sums and applying an activation function at each layer

$$o = a^{(L)} = \sigma^{(L)} \left(b^{(L)} + W^{(L)} \sigma^{(L-1)} \left(\dots f^{(1)} (b^{(1)} + W^{(1)} x) \dots \right) \right)$$

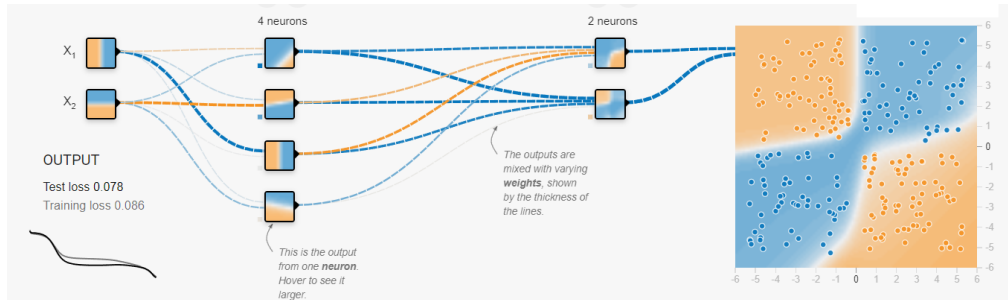
Forward Propagation Cont.



Before producing predictions. Adapted from TensorFlow playground: Daniel Smilkov and Shan Carter.

Forward Propagation Cont.

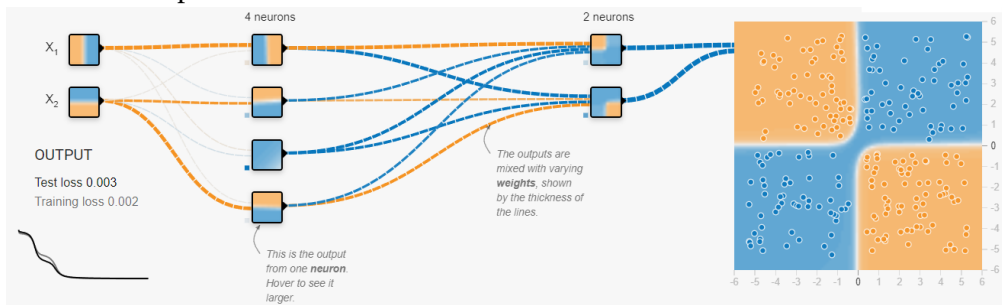
- The goal is to adjust the network's parameters to improve the predictions
- The loss is calculated after the forward pass, telling us how far off our predictions are from the true values



Loss values for predictions. Adapted from TensorFlow playground: Daniel Smilkov and Shan Carter.

BackPropagation and Parameter Update

- The network uses the **loss** to adjust its **weights and biases** through a process called **backpropagation**
- Backpropagation calculates how much weights should change to reduce the error
- This will be explained in more detail in the next lecture



Predictions get better as the weights get updated. Adapted from TensorFlow playground: Daniel Smilkov and Shan Carter.

- 1 Introduction
- 2 Multi-Layer Perceptron (MLP)
- 3 Neural Networks
- 4 Training Neural Networks
- 5 References**

Contributions

These slides are authored by:

- Sogand Salehi
- Erfan Sobhaei

- [1] R. Ramakrishnan, “Deep learning course at carnegie mellon university.” <https://deeplearning.cs.cmu.edu/F23/index.html>, 2023.
Accessed: 2024-09-04.
- [2] E. Mousavi and K. Alishahi, “Deep learning course at sharif university of technology.” [https://https://dnncourse.github.io/lectures](https://dnncourse.github.io/lectures), 2023.
Accessed: 2024-09-04.
- [3] D. Smilkov and S. Carter, “A neural network playground.” playground.tensorflow.org, 2022.
Accessed: 2024-10-14.
- [4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [5] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques for Building Intelligent Systems*. O'Reilly Media, 2019.