# Basic introduction to neural networks

Muhammed Shifas PV

University of Crete, Dept of Computer Science
shifaspv@csd.uoc.gr

CS-HY578: Speech Signal Processing, 9 May 2022

# About me

**Presently:** Apple Inc., Cambridge, UK.

**2017-2021:** PhD in Speech Processing, University of Crete, Greece.

**2014-2016:** M.Tech in Signal Processing, NIT Calicut, India.

**2009-2013:** B.Tech in Communication Engineering, Calicut University, India.

# Outline

# Outline

# Modelling biological neuron

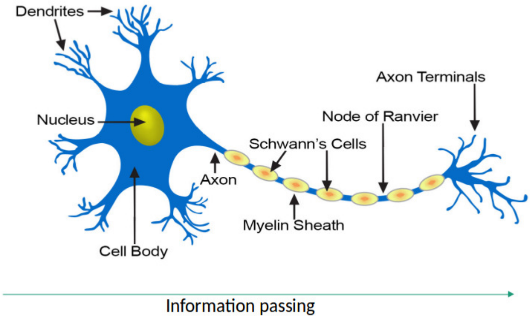



Figure: The biological structure of a neuron

# Mathematical equivalent of a neuron

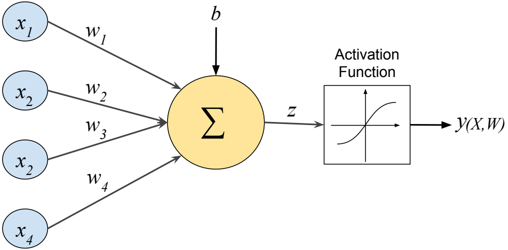



Figure: Mathematical equivalent of a neuron

$$y\left(\mathbf{x}^{(k)}\right) = \phi\left(z_i\right) = \phi\left(\mathbf{w}_i^\top \mathbf{x}^{(k)} + b_i\right) = \phi\left(\sum_{j=1}^{n} w_{ij} x_j^{(k)} + b_i\right) \tag{1}$$

- Output is a function of the input (data) and the weights.
- **Training:** optimize the weights such that to reach to the desired output.

# Mathematical equivalent of a neuron

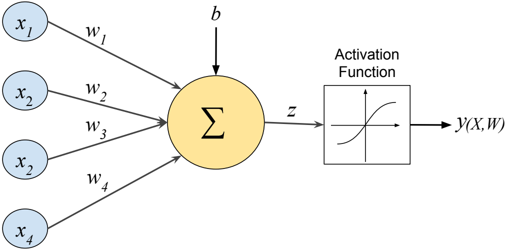



Figure: Mathematical equivalent of a neuron

$$y\left(\mathbf{x}^{(k)}\right) = \phi\left(z_i\right) = \phi\left(\mathbf{w}_i^\top \mathbf{x}^{(k)} + b_i\right) = \phi\left(\sum_{j=1}^{n} w_{ij} x_j^{(k)} + b_i\right) \tag{1}$$

- Output is a function of the input (data) and the weights.
- **Training:** optimize the weights such that to reach to the desired output.

# Neural Network: Network of Neurons

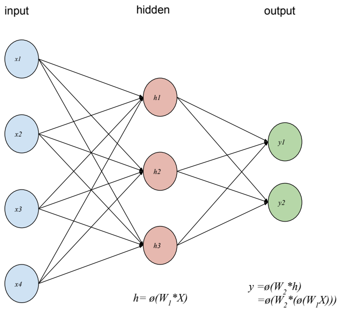



Figure: Neural network

## Higlight

- Intuition: any complex function can be approximated as a series of simple non-linear functions.

# Neural Network: Network of Neurons

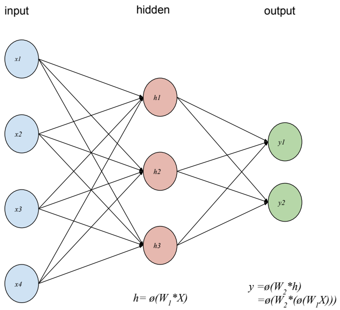



Figure: Neural network

## Higlight

- **Intuition: any complex function can be approximated as a series of simple non-linear functions.**

# Mathamatical Insight

$$h = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = \Phi \begin{pmatrix} w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \\ w_{30} & w_{31} & w_{32} & w_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \tag{2}$$

$$y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \Phi \begin{pmatrix} w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} \tag{3}$$

$$y = \Phi \begin{pmatrix} w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \\ w_{30} & w_{31} & w_{32} & w_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \tag{4}$$

# Non-linear activation functions: Φ

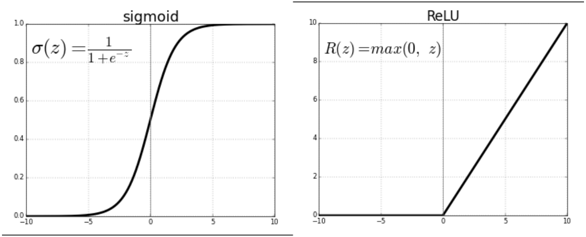



Figure: Commonly used activation functions

# Which function are we looking for ?

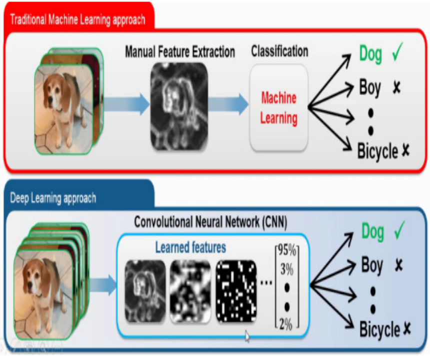
- **Assumtion:** there is a statistics, hidden in our data
- The statistics to model depends on the task: Speaker identification, emotion detection, enhancement

## Model input dimension

- Traditional approach: manually extracts the features and feed into the network.
- Advance models: feed the raw samples as it is into the models, letting the network to extracts the task relevant features.

# Training the Model: Teaching

- Training:Teaching the model by exploring to the already know data pair
  - Supervised: data (input, output)
  - Unsupervised: data (input,)



# Training the Model: Weight tuning

$$\hat{y} = \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \end{pmatrix} = \Phi \begin{pmatrix} w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} \tag{5}$$

- Loss: the measure of deviation of network prediction $\hat{y}$ from the true training set label $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$
- Penalize each wrong prediction (tune the weight) so that it to be a good predictor at the end.

# Parameter optimization: Gradient descent

Given a set of training input and output pairs

$$\{\{\boldsymbol{x}_1, \boldsymbol{t}_1\} \ldots, \{\boldsymbol{x}_n, \boldsymbol{t}_n\}\} \tag{6}$$

compute the loss function for each prediction

$$E = \quad = \frac{1}{2} \sum_{p=1}^{n} \sum_{i=1}^{K} (y_i(\boldsymbol{x}_p) - t_{pi})^2 \tag{7}$$

$$\frac{\partial E}{\partial w_i} = \left(\frac{\partial E}{\partial z}\right)\left(\frac{\partial z}{\partial w_i}\right) \tag{8}$$

with sigmoid as the activation function $\phi$, $y(\boldsymbol{z}) = \frac{1}{1+\exp(-z)}$

$$\frac{\partial E}{\partial w_i} = (y(\boldsymbol{x}) - t)y(\boldsymbol{x})(1 - y(\boldsymbol{x}))x_i \tag{9}$$

Walk on the direction to which gradient descend

$$w_{i+1} = w_i - \gamma \frac{\partial E}{\partial w_i} \tag{10}$$

# The logistic loss function (classifier)

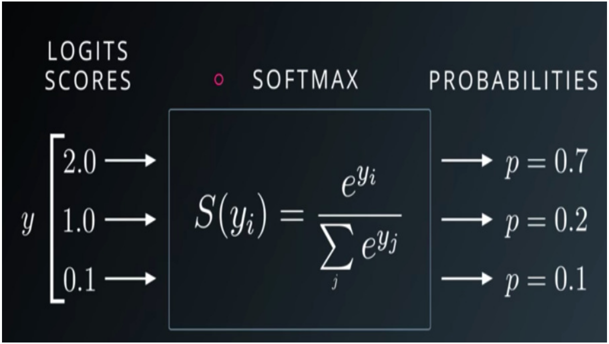
  

| | Image #1 | Image #2 | Image #3 |
|---|---|---|---|
| Dog | -0.39 | -4.61 | 1.03 |
| Cat | 1.49 | 3.28 | -2.37 |
| Horse | 4.21 | 1.46 | -2.27 |

SVM loss: Minimize the objective

$$L(y, \hat{y}) = \sum_{i \neq c} max(0, \hat{y}_i - \hat{y}_c + \Delta) \tag{11}$$

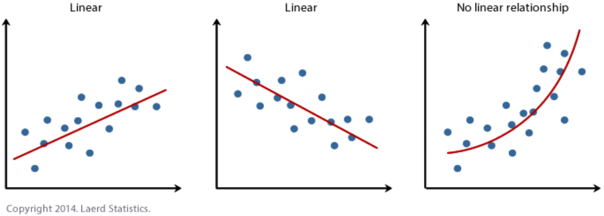
# Probabilistic Loss function




The cross entropy loss:

$$L(y, \hat{y}) = -\sum_i (y_i log(\hat{p}_i) + (1 - y_i) log(1 - \hat{p}_i)) \quad (12)$$

# The Regression loss

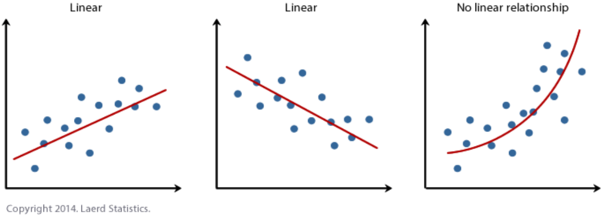



The target to be predicted is a continues value function: eg. speech enhancement.

The final layer of regression model:

$$\hat{y} = \begin{pmatrix} \hat{y}_1 \end{pmatrix} = \Phi \begin{pmatrix} w_{10} & w_{11} & w_{12} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} \qquad (13)$$

# The Regression loss




The target to be predicted is a continues value function: eg. speech enhancement.
The final layer of regression model:

$$\hat{y} = \begin{pmatrix} \hat{y}_1 \end{pmatrix} = \Phi \begin{pmatrix} w_{10} & w_{11} & w_{12} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} \tag{13}$$

# Regression loss

Mean square error:

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{14}$$

Mean absolute error:

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \tag{15}$$

$$n = \text{Total data points}$$

# Outline

# Fully Connected Neural Network

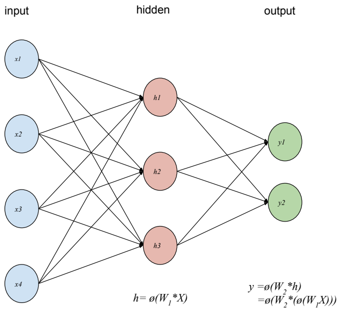



Figure: Fully Connected Network

## Network identity

- All nodes from the previous layer is connected to the next layer.

# Mathematical Insight

$$\hat{h} = \begin{pmatrix} \hat{h}_1 \\ \hat{h}_2 \\ \hat{h}_3 \end{pmatrix} = \begin{pmatrix} w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \\ w_{30} & w_{31} & w_{32} & w_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \tag{16}$$

$$\hat{y} = \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \end{pmatrix} = \begin{pmatrix} w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} \hat{h}_1 \\ \hat{h}_2 \\ \hat{h}_3 \end{pmatrix} \tag{17}$$

The number of parameters are linear with input/ hidden layer size

# Outline

# Why does a new network?

The fully connected network has some draw backs:

- It always gives a merged representation of the previous input/hidden layer
- Failed to capture the local information in the input signal.
- The complexity of the model increases rapidly as we build deeper networks.

# Convolutional Neural Network

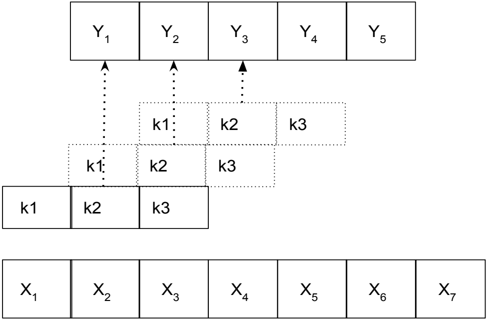



Figure: Convolution Network

$$Y[n] = (X * k)[n] = \sum_{i=1}^{n} X(n)k(n-m) \qquad (18)$$

# Convolution with multiple kernels

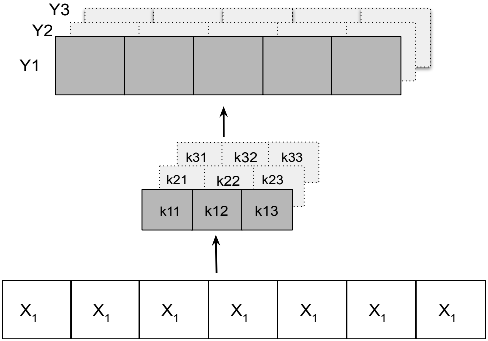



Figure: Convolution Network

$$Y_i[n] = (X * k_i)[n] = \sum_{i=1}^{n} X(n) k_i(n-m) \tag{19}$$

Number of kernels = number of channels

# Convolution channels: 2D

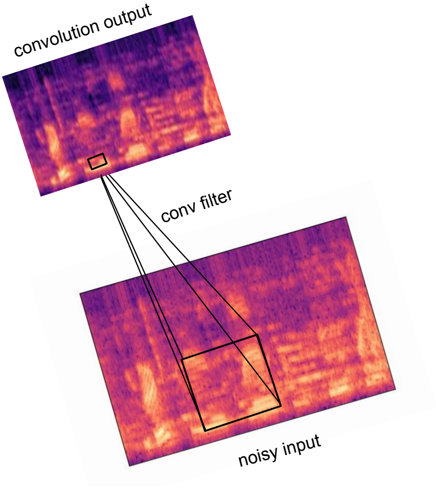



Figure: 2D Convolution

- Number of parameters are independent of the input size.
- Network parameter is independent of the input dimension.
- The kernel size is customizable.

# Dilated convolution

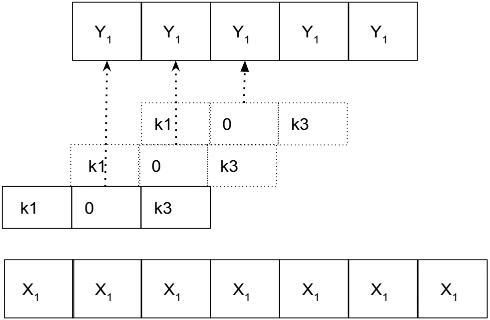



Figure: Dilated convolution Network

# Outline

# Why Recurrency?

- speech articulations are a highly correlated over time.
- Estimation of the current input phoneme can tell something about the phonemes follows.

  -eg. We will meet ....

- We must store and pass the information at the current instant to be consulted in the future predictions.
- Adding the Markove structure into the neural network

# Recurrency in Fully Connected Network

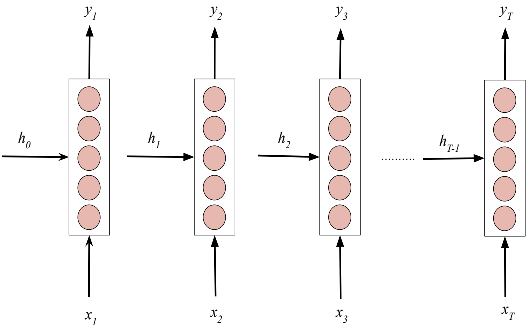



Figure: Fully connected Recurrent Network

# Fully Connected Long Short-Term Memory (FC-LSTM)

$$
\begin{aligned}
i_t &= \Phi(W_{xi}X_t + W_{hi}h_t + W_{ci}oc_{t-1} + b_i) \\
f_t &= \Phi(W_{xf}X_t + W_{hf}h_t + W_{cf}oc_{t-1} + b_f) \\
o_t &= f_t oc_{t-1} + i_t otanh(W_{xc}X_t + W_{hc}h_{t-1} + b_0) \\
y_t &= \Phi(W_{xo}X_t + W_{ho}h_t + W_{co}oc_{t-1} + b_y)
\end{aligned}
\tag{20}
$$

# Outline

# Acknowledgments

This project has received funding from the EU's H2020 research and innovation programme under the MSCA GA 675324

# Thank for your attention