Neural Networks
Lecture 2

Topics:

- ANN Features
- Problem Domain and Application of ANN
- Advantages of ANN
- Activation functions
- Adaline
- Linear Separable Problems

PPT courtesy : Hinton

FEATURES OF ARTIFICIAL NETWORK (ANN)

- ANNs are extremely powerful computational devices (Universal computers).
- ANNs are modeled on the basis of current brain theories, in which information is represented by weights.
- ANNs have massive parallelism which makes them very efficient.
- They can learn and generalize from training data so there is no need for enormous feats of programming.
- Storage is fault tolerant i.e. some portions of the neural net can be removed and there will be only a small degradation in the quality of stored data.

FEATURES OF ARTIFICIAL NETWORK (ANN)

- They are particularly fault tolerant which is equivalent to the “graceful degradation” found in biological systems.
- Data are naturally stored in the form of associative memory which contrasts with conventional memory, in which data are recalled by specifying address of that data.
- They are very noise tolerant, so they can cope with situations where normal symbolic systems would have difficulty.
- In practice, they can do anything a symbolic/ logic system can do and more.
- Neural networks can extrapolate and intrapolate from their stored information. The neural networks can also be trained. Special training teaches the net to look for significant features or relationships of data.

Domain and Applications of ANN

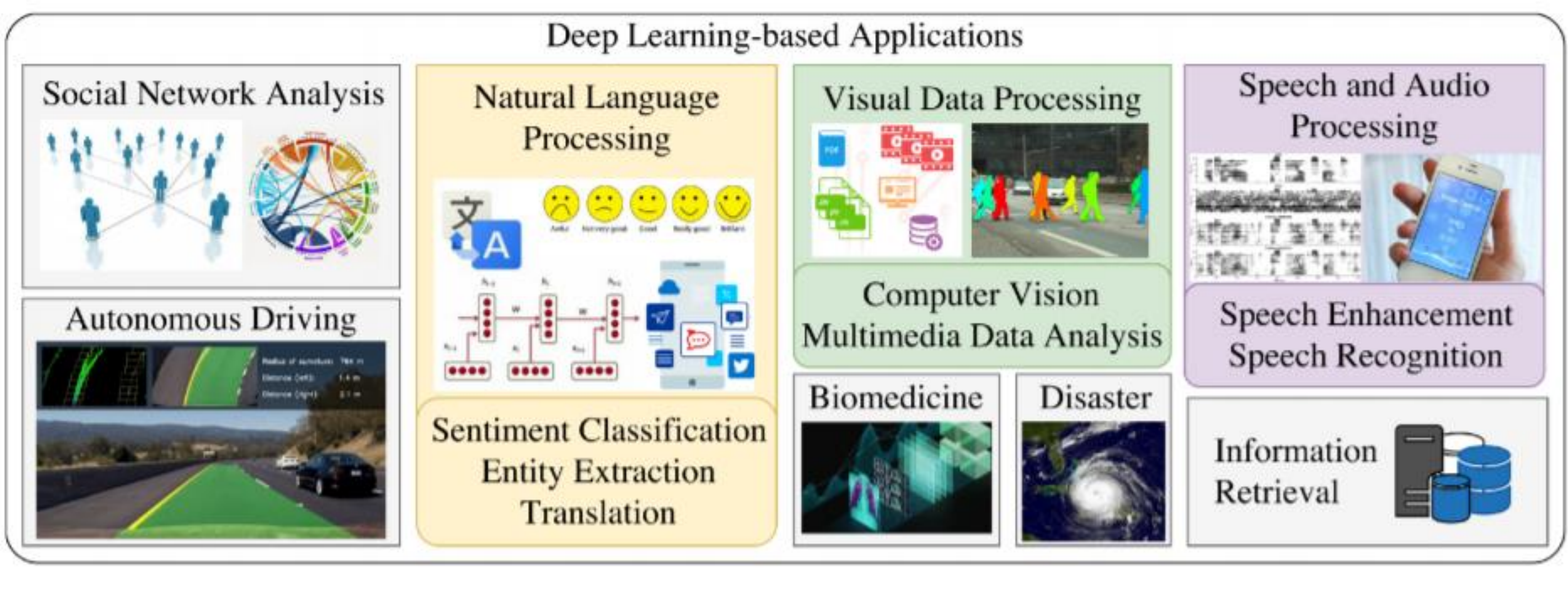


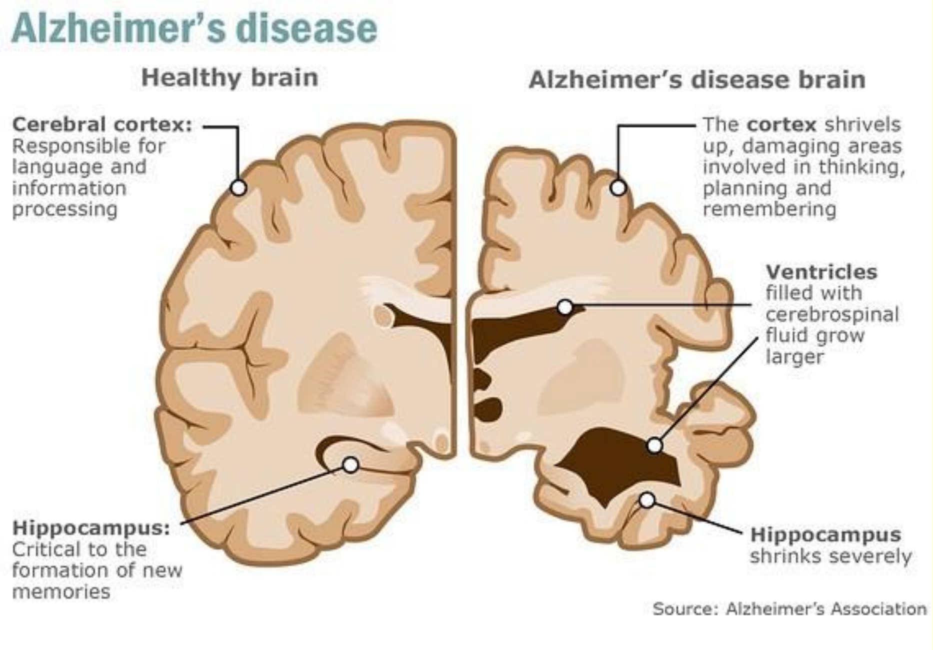
Fig: Some of the popular deep learning applications.

Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, ShuChing Chen, and S. S. Iyengar. 2018. A Survey on Deep Learning: Algorithms, Techniques, and Applications. ACM Comput. Surv. 51, 5, Article 92 (September 2018), 36 pages.

Computer Vision

OCR model for reading Captchas

- **Alzheimer Classification**



Advantages

- The main advantage of ANN is parallel processing. This makes it more useful.
- Due to their parallel processing structure, any failure in one neural element will not affect the rest of the process.
- Neural networks can be applied to any application and they can solve any complex problem.
- By implementing appropriate learning algorithms, an ANN can be made to learn without reprogramming.

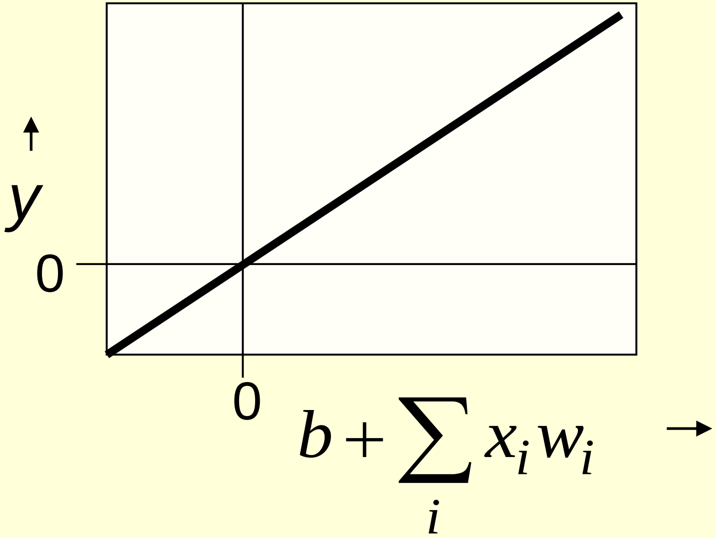
Disadvantages

- All the parallel processing requires a huge amount of processing power and time.
- There is a requirement for a “training” period before real-world implementation.

Linear neurons

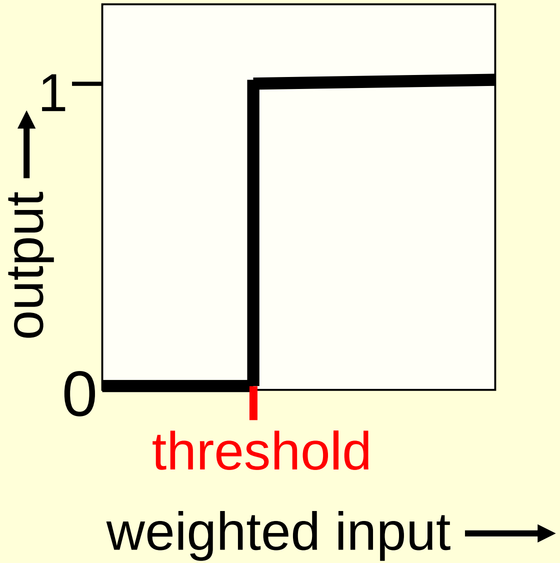
- These are simple but computationally limited
  - If we can make them learn we may get insight into more complicated neurons.

$$y = b + \sum_i x_i w_i$$



Binary threshold neurons

- McCulloch-Pitts (1943): influenced Von Neumann.
  - First compute a weighted sum of the inputs.
  - Then send out a fixed size spike of activity if the weighted sum exceeds a threshold.
  - McCulloch and Pitts thought that each spike is like the truth value of a proposition and each neuron combines truth values to compute the truth value of another proposition!



Binary threshold neurons

- There are two equivalent ways to write the equations for a binary threshold neuron:

$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 \text{ if } & z \geq \theta \\ 0 \text{ otherwise} & \end{cases}$$

$$\boxed{q = - b}$$

$$z = b + \underset{i}{\text{å}}\, x_i w_i$$

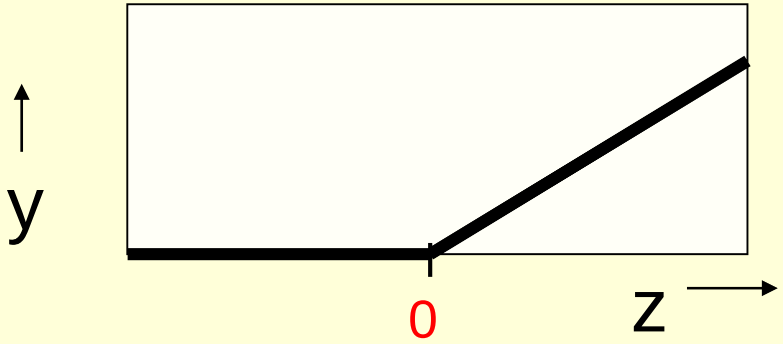
$$y = \begin{cases} 1 \text{ if } & z ^3\ 0 \\ 0 \text{ otherwise} & \end{cases}$$

Rectified Linear Neurons
(sometimes called linear threshold neurons)

They compute a linear weighted sum of their inputs.
The output is a non-linear function of the total input.

$$z = b + \sum_i x_i w_i$$

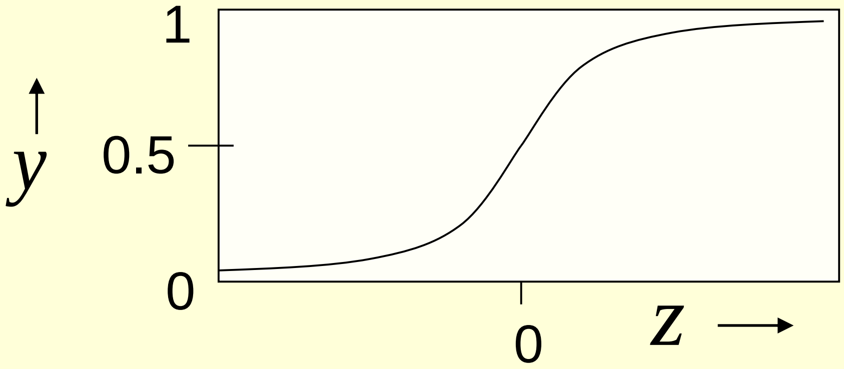
$$y = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$



Sigmoid neurons

- These give a real-valued output that is a smooth and bounded function of their total input.
  - Typically they use the logistic function
  - They have nice derivatives which make learning easy

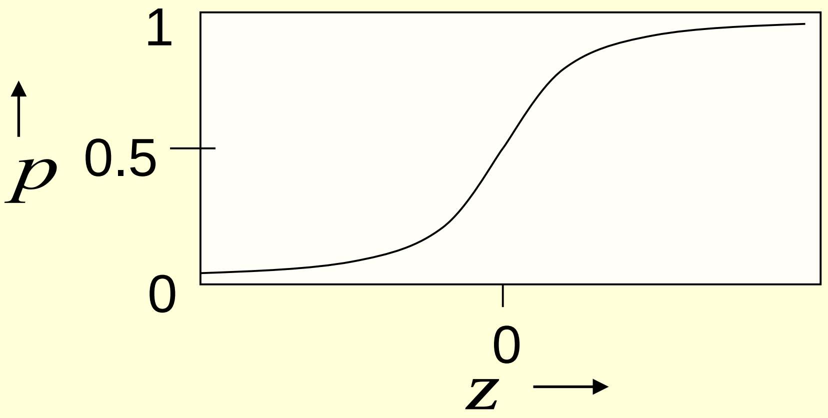
$$z = b + \sum_i x_i w_i \qquad y = \frac{1}{1 + e^{-z}}$$

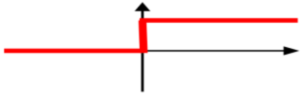
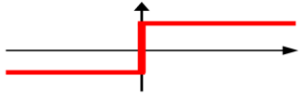
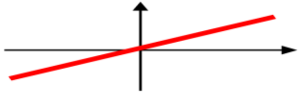
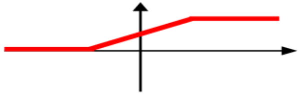
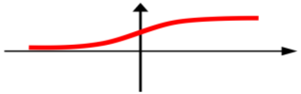
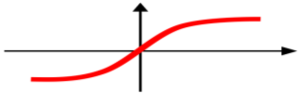
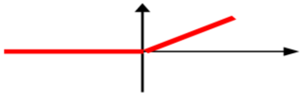
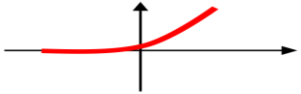


Stochastic binary neurons

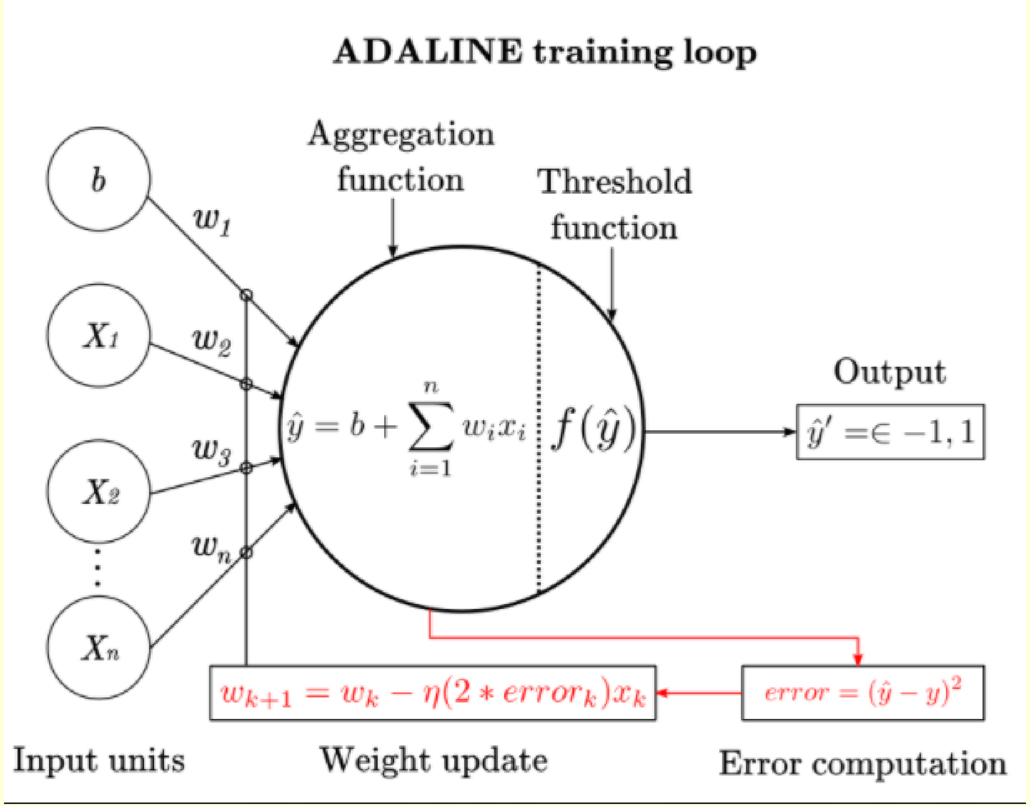
- These use the same equations as logistic units.
  - But they treat the output of the logistic as the probability of producing a spike in a short time window.
- We can do a similar trick for rectified linear units:
  - The output is treated as the Poisson rate for spikes.

$$z = b + \sum_i x_i w_i \qquad p(s=1) = \frac{1}{1+e^{-z}}$$



Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

ADALINE



https://pabloinsente.github.io/the-adaline

- Widrow and Hoff came up with the ADALINE idea on a Friday during their first session working together.
- At the time, implementing an algorithm in a mainframe computer was slow and expensive, so they decided to build a small electronic device capable of being trained by the ADALINE algorithm to learn to classify patterns of inputs.
- learning procedure is based on the outcome of a linear function rather than on the outcome of a threshold function

Linear Separable Problems

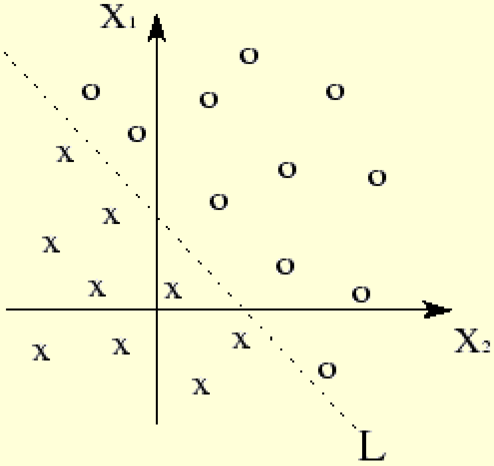


Figure : Linearly Separable Pattern

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$$Y = X_1 \oplus X_2$$

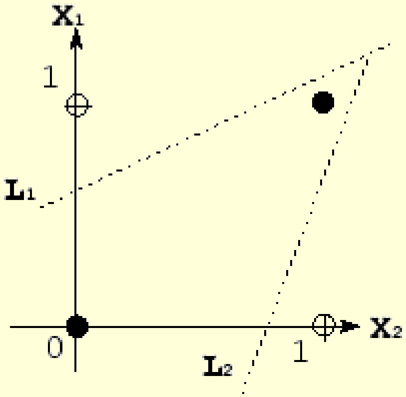


Figure : Exclusive-OR Function