Topic 9

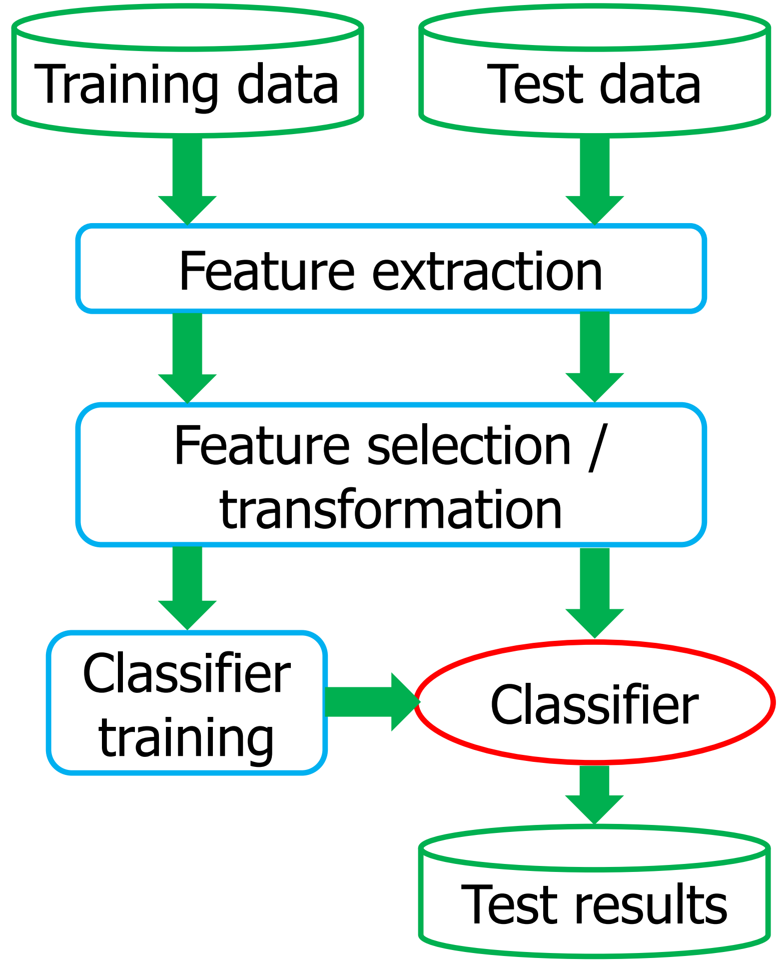
Deep Learning for Audio Applications

(some slides are adapted from Kevin Duh's slides on Deep Learning and Neural Networks)

Audio Classification Tasks

- Music genre, mood, artist, composer, instrument classification
- Auto tagging, i.e., labeling music with words
- Chord recognition
- Acoustic event detection
- Speech/speaker recognition

- General flowchart

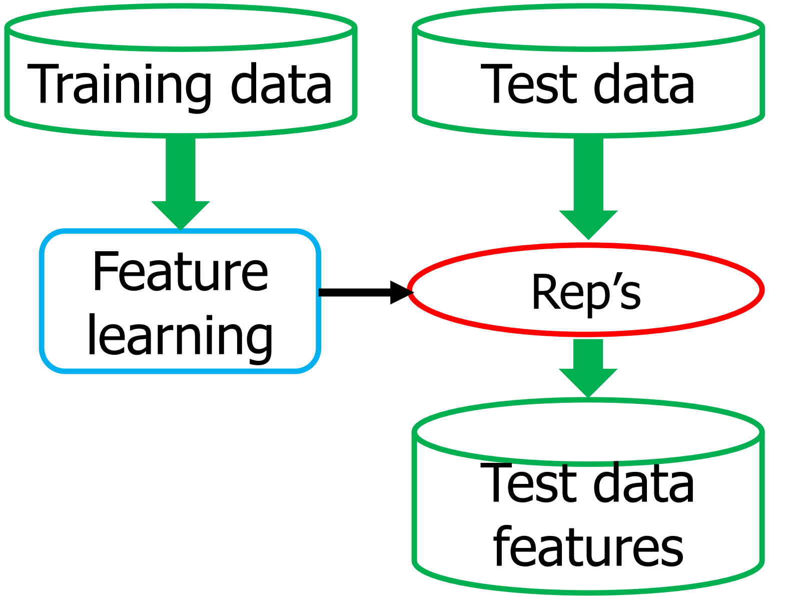


Features that we have studied

- Raw input: audio waveform or spectrogram
- Feature output:
  - RMS, Zero Crossing Rate
  - Spectral centroid, spread, skewness, kurtosis, flatness, irregularity, roll-off, flux, etc.
  - Harmonic features
  - MFCC, LPC, PLP, etc.
- Hand-crafted / engineered / pre-defined
- Hard to decide what features to use for a task
- Question: can computers learn features directly from data?

Feature / Representation Learning

- Learn a transformation of "raw" inputs to a representation that can be effectively exploited in a task
- Automatic / does not rely on human knowledge
- Target for a specific task



Methods Viewed as Feature Learning

- Principal Component Analysis (PCA)
  - Learns a linear transformation, where rows of $\boldsymbol{W}$ are the orthogonal directions of greatest variance in the training data

$$f(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

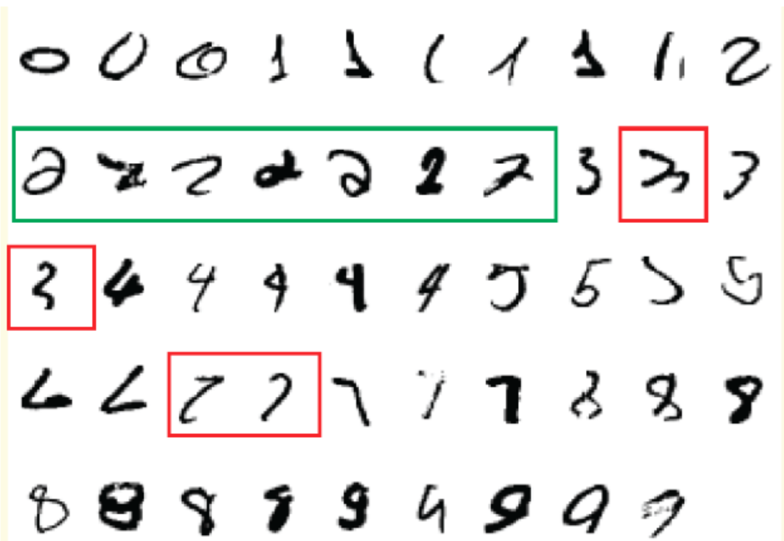
- Dictionary Learning (e.g., NMF)
  - Learns a linear transformation, where the input, transformation matrix, and activation matrix (i.e., features), are all non-negative

$$\boldsymbol{x} = \boldsymbol{W}\boldsymbol{h}$$

$$(\boldsymbol{X} = \boldsymbol{W}\boldsymbol{H})$$

Are linear features good enough?

- Probably not…
- The world is complex and often nonlinear.



Can you define a linear transformation on the images to discriminate "2"s from non-"2"s?

$$f(\boldsymbol{x}) = \sum_i w_i x_i + b$$

where $\boldsymbol{x}$ is a vector representing all pixel values of a image.

Are these features highly nonlinear…

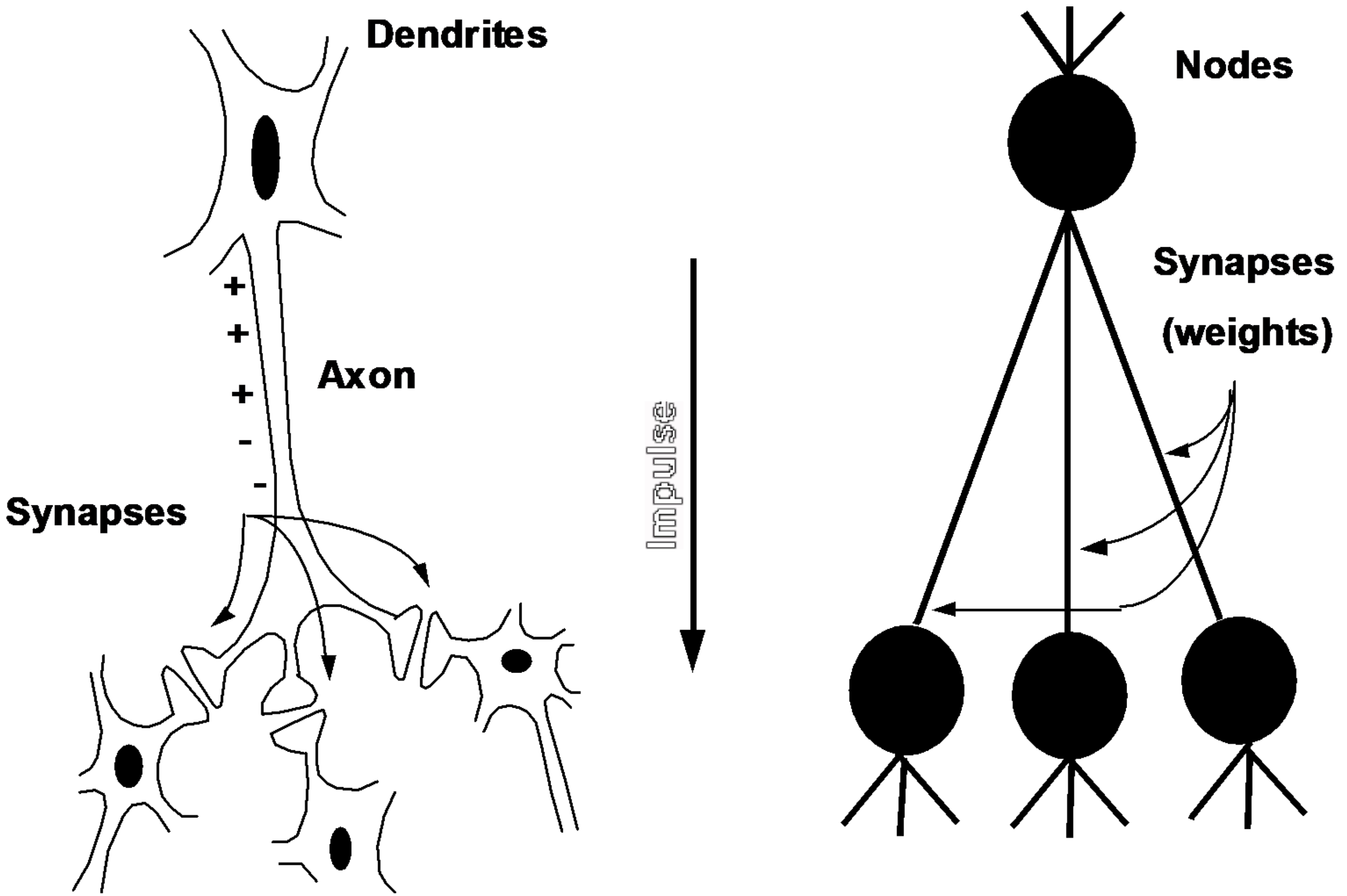
…to the waveform or spectrogram?

- RMS, ZCR
- Spectral centroid, spread, skewness, kurtosis, flatness, flux
- Harmonic features
- Cepstrum: $|\mathcal{F}^{-1}\{\log|\mathcal{F}\{x(t)\}|^2\}|^2$
- MFCC
- LPC

Can we learn highly non-linear features?

Deep neural networks!

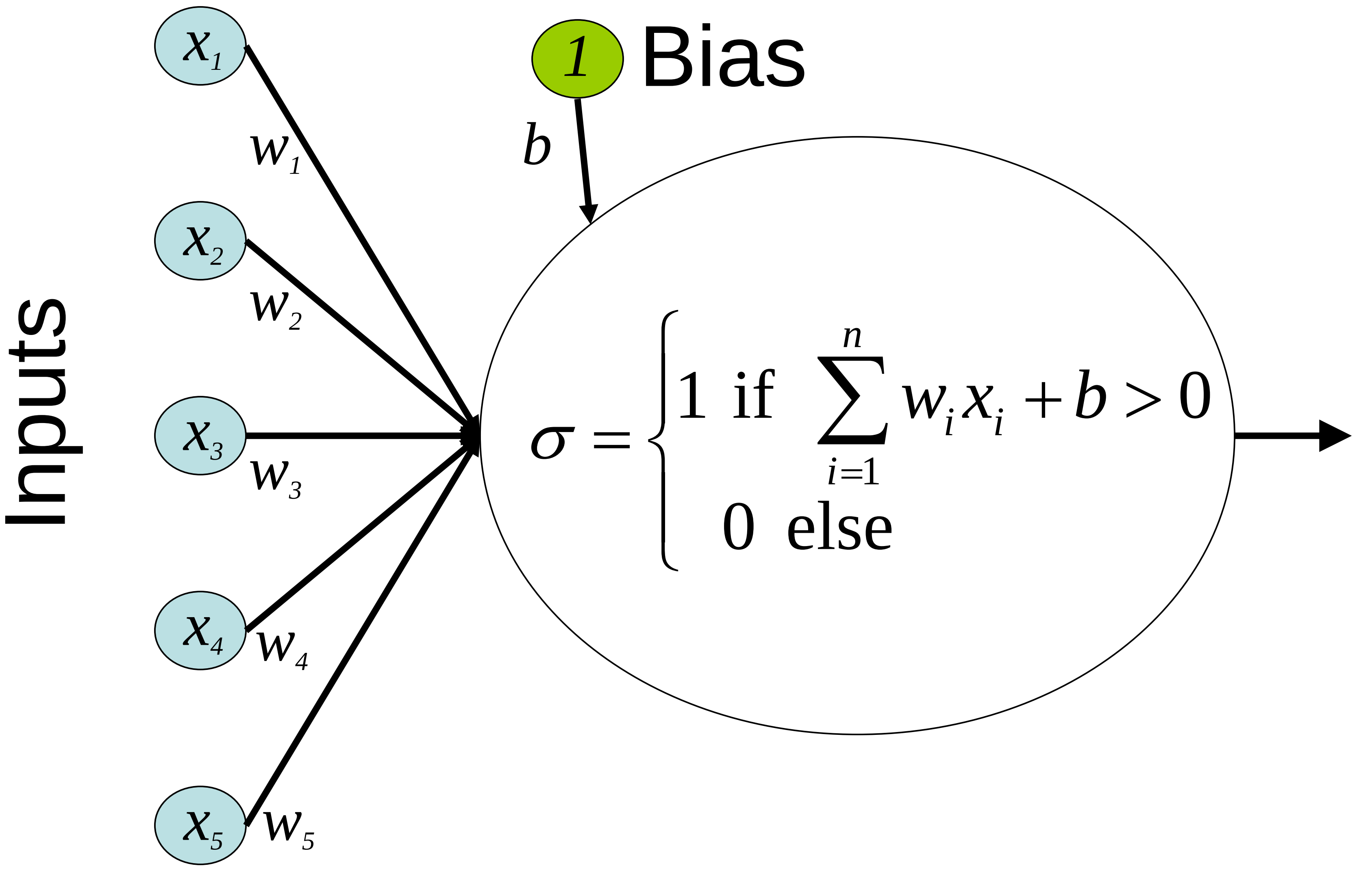
Biological Analogy



History of Neural Networks

- 1943 – first neural network computing model by McCulloch and Pitts
- 1958 – Perceptron by Rosenblatt
- 1960’s – a big wave
- 1969 – Minsky & Papert’s book “Perceptrons”
- 1970’s – “winter” of neural networks
- 1975 – Backpropagation algorithm by Werbos
- 1980’s – another big wave
- 1990’s – overtaken by SVM proposed in 1993 by Vapnik
- 2006 – a fast learning algorithm for training deep belief networks by Hinton
- 2010’s – another big wave
- 2018 Turing Award – Hinton, Bengio & LeCun
- ????????

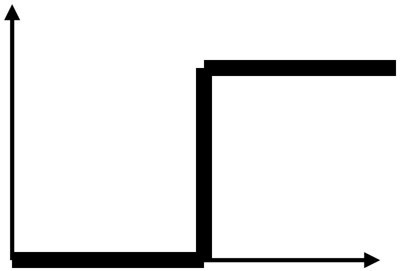
Perceptron



Inputs
x_1
x_2
x_3
x_4
x_5
w_1
w_2
w_3
w_4
w_5
1
Bias
b
$\sigma = \begin{cases} 1 \text{ if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 \text{ else} \end{cases}$

A compromise function

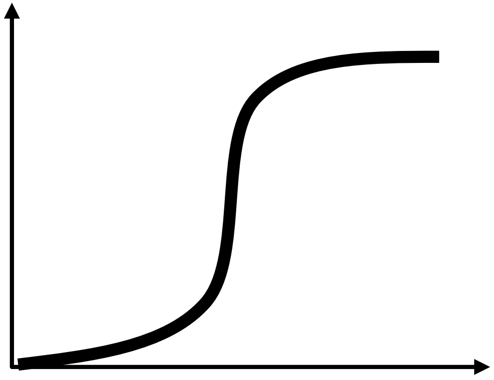
- Perceptron



$$output = sign(\mathbf{w}^T\mathbf{x} + b)$$

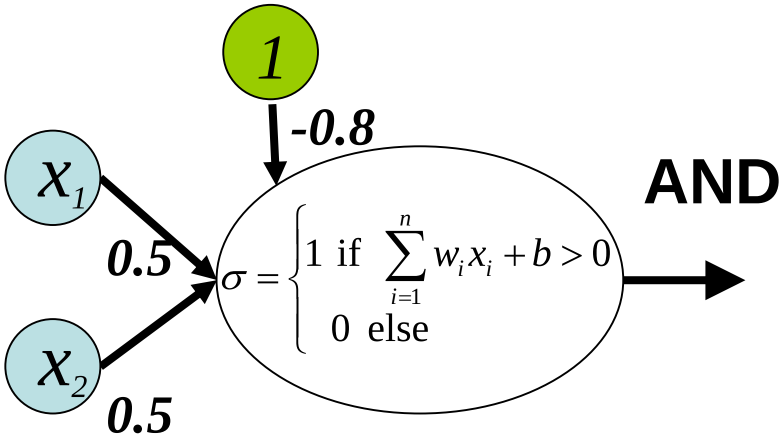
- Sigmoid (Logistic)

$$output = \sigma(\mathbf{w}^T\mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x} + b)}}$$

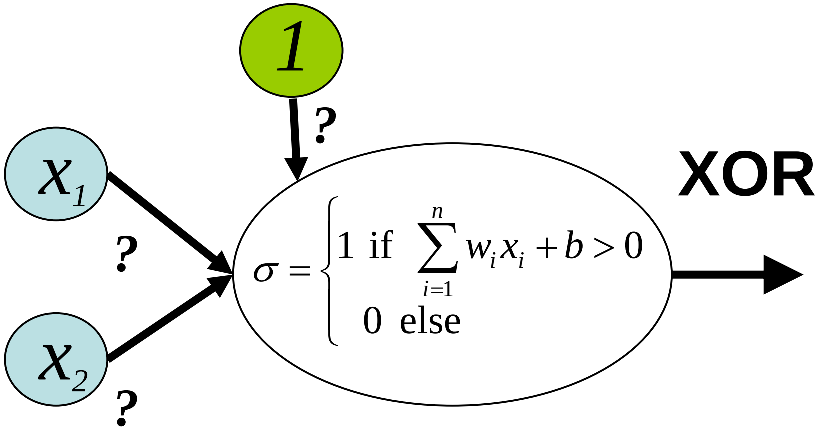


Limitations of 1-layer Nets

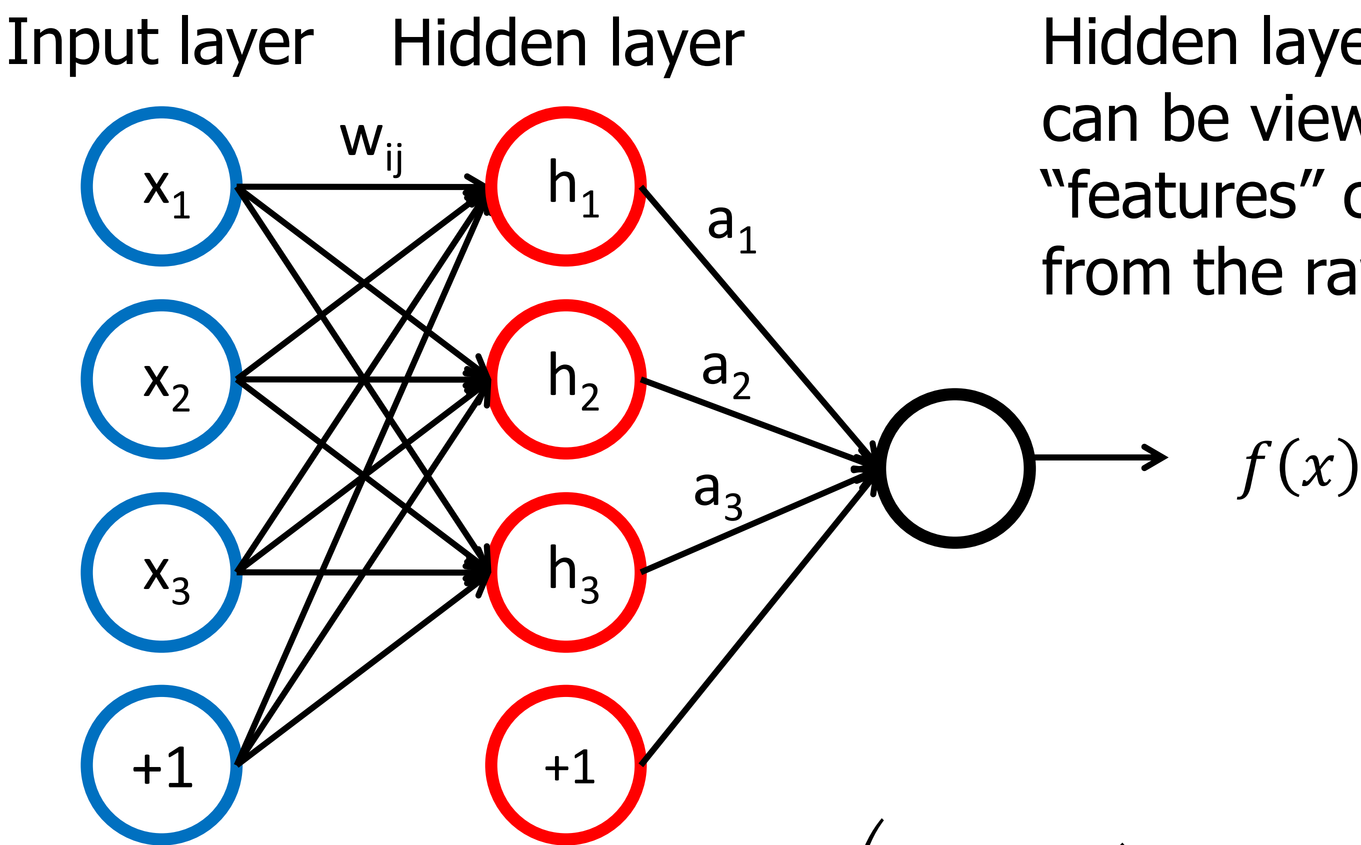
- Only express linearly separable cases
  - For example, they are good as logic operators "AND", "NOT", and "OR"



- How about "XOR", which is not a linearly separable case?



2-layer Nets

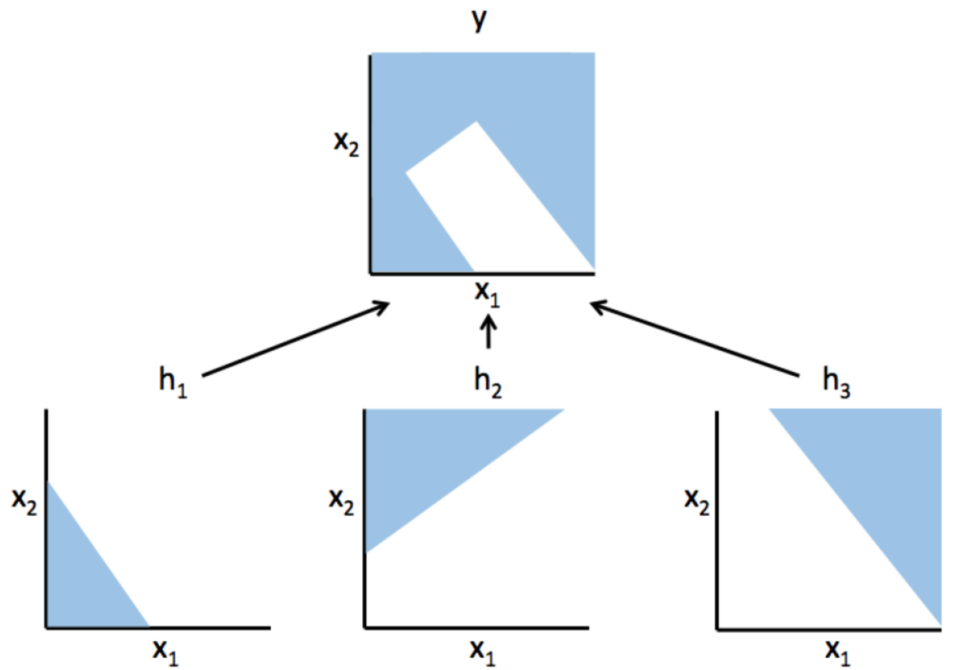


Hidden layer output can be viewed as "features" calculated from the raw input

$$f(x) = \sigma\left(\sum_j a_j h_j\right) = \sigma\left(\sum_j a_j \sigma\left(\sum_i w_{ij} x_i\right)\right)$$

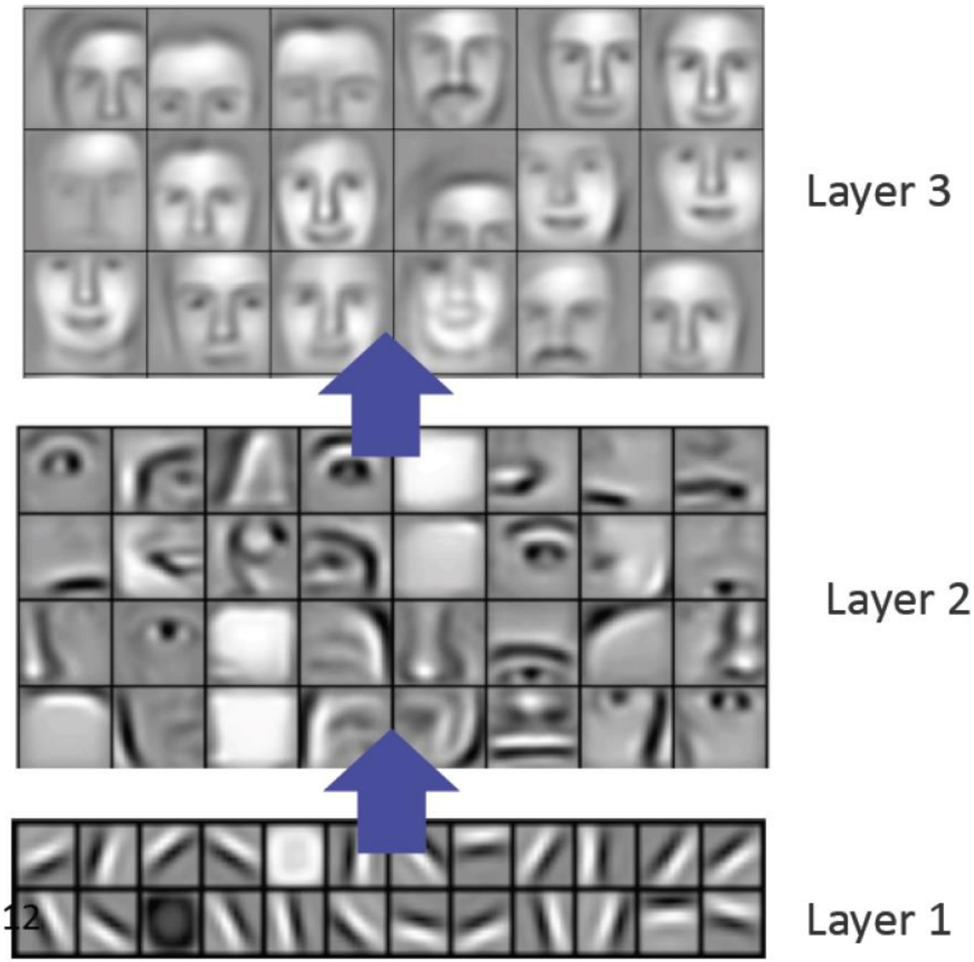
Richer Representation with More Layers

- 1-layer nets only model linear hyperplanes
- 2-layer nets can approximate any continuous function, given enough hidden nodes
- >3-layer nets can do so with fewer nodes and weights



Richer Representations

(Lee et al., 2009)



How to learn the weights?

- Given training data - input and label pairs $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$
- Adjust network weights to minimize difference (error) between $f\left(\boldsymbol{x}^{(i)}\right)$ and $y^{(i)}$
  - Calculate derivative of error w.r.t. weights
  - Back Propagation algorithm

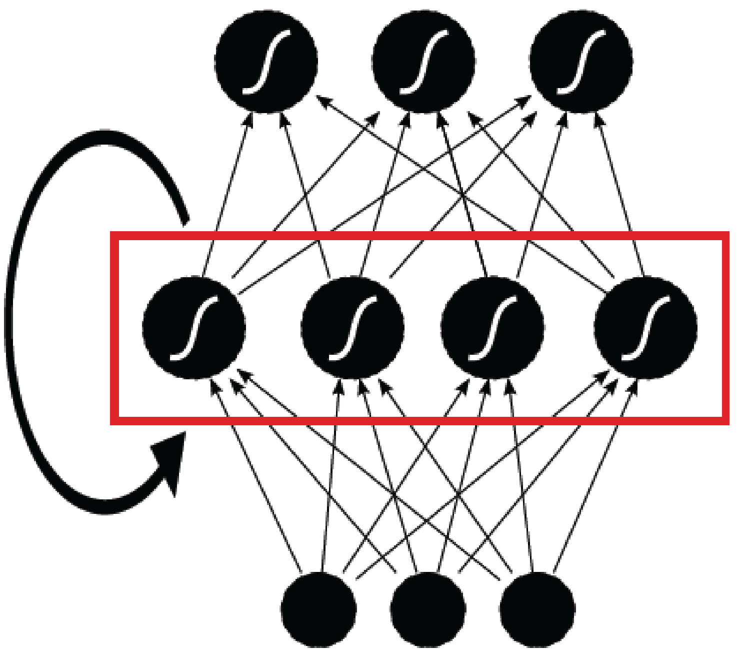
- See derivation on white board

Problems of BP for deep networks

- Vanishing gradient problem
  – Gradients vanishes when they are propagated back to early layers, hence their weights are hard to adjust
- Many local minima
  – Which will trap gradient decent methods

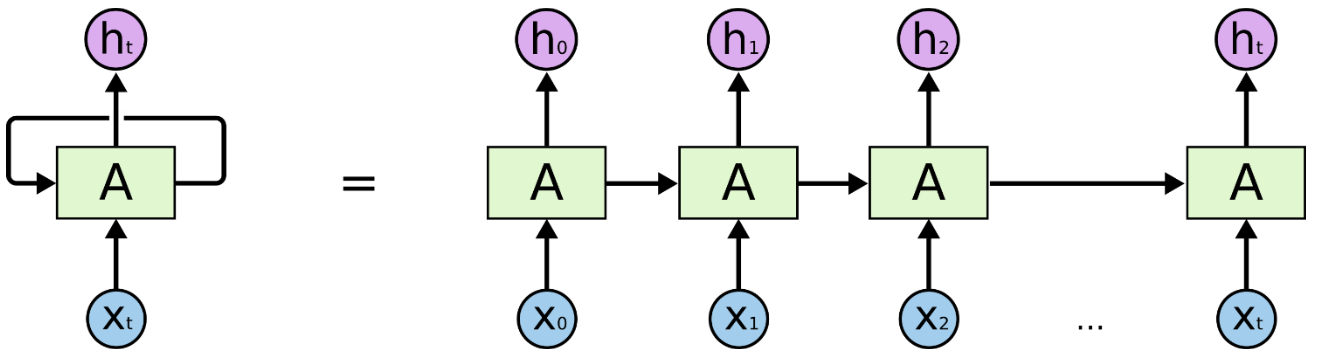
Recurrent Neural Network (RNN)

- Network with loops
- Good for sequential data



Another Look of RNN

- Unfolding in time



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

- Training: Back Propagation Through Time (BPTT)

Bidirectional RNN

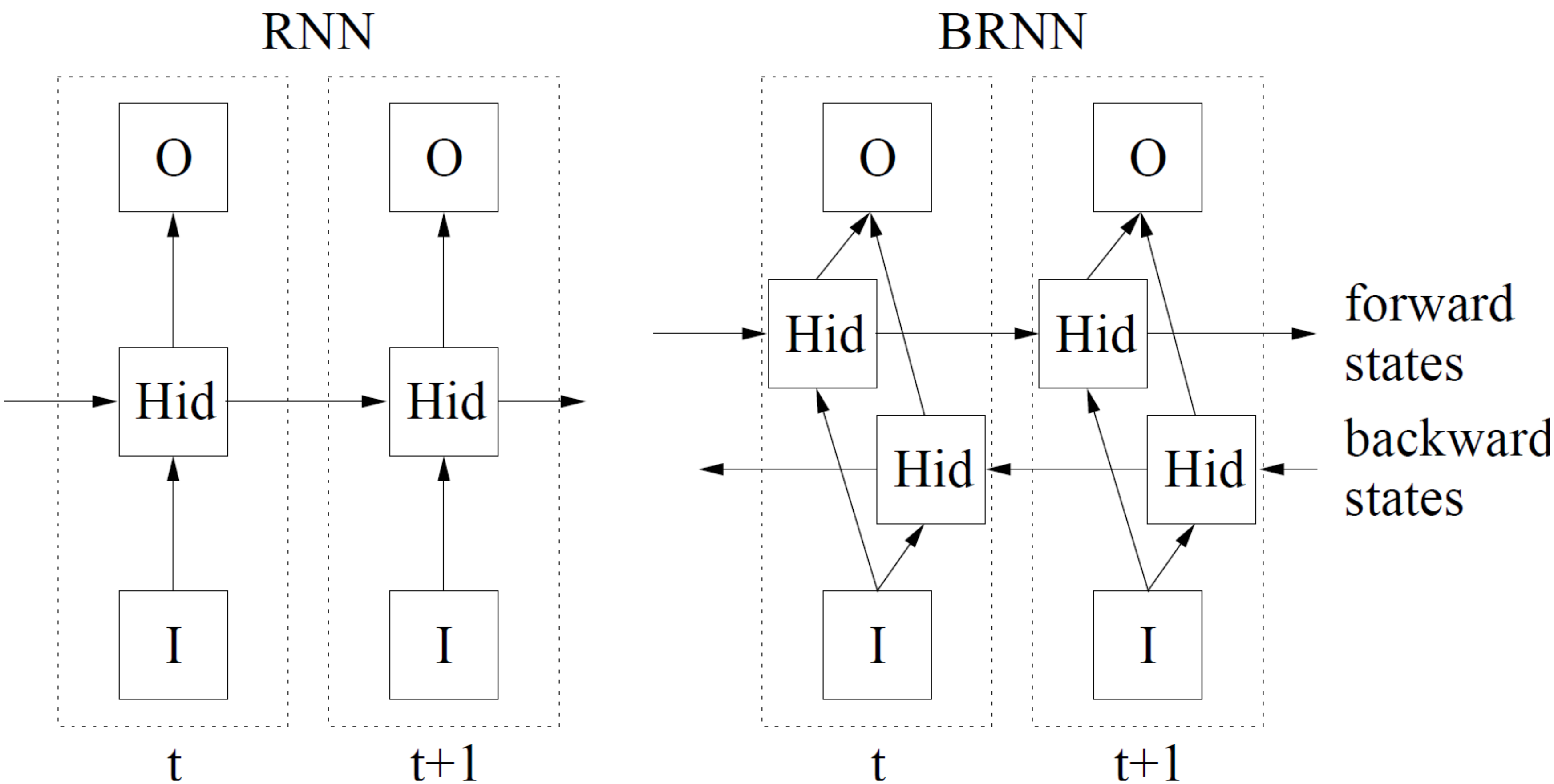
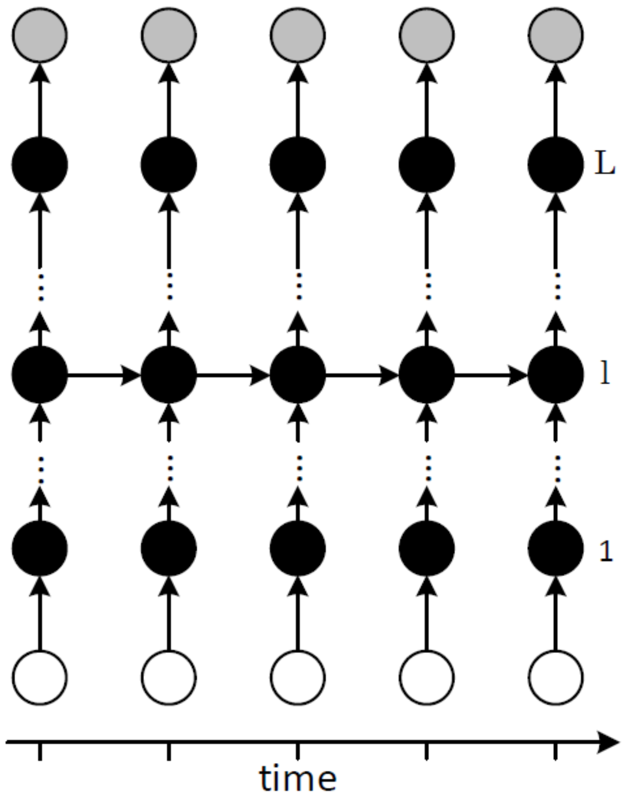


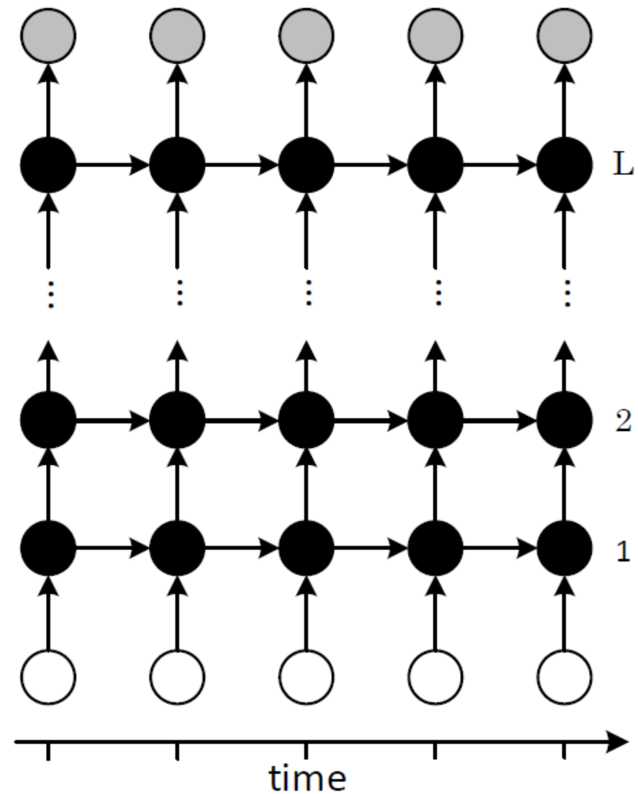
Figure from [Graves, 2008]

Deep RNN (DRNN)

- RNN lacks hierarchical processing of inputs



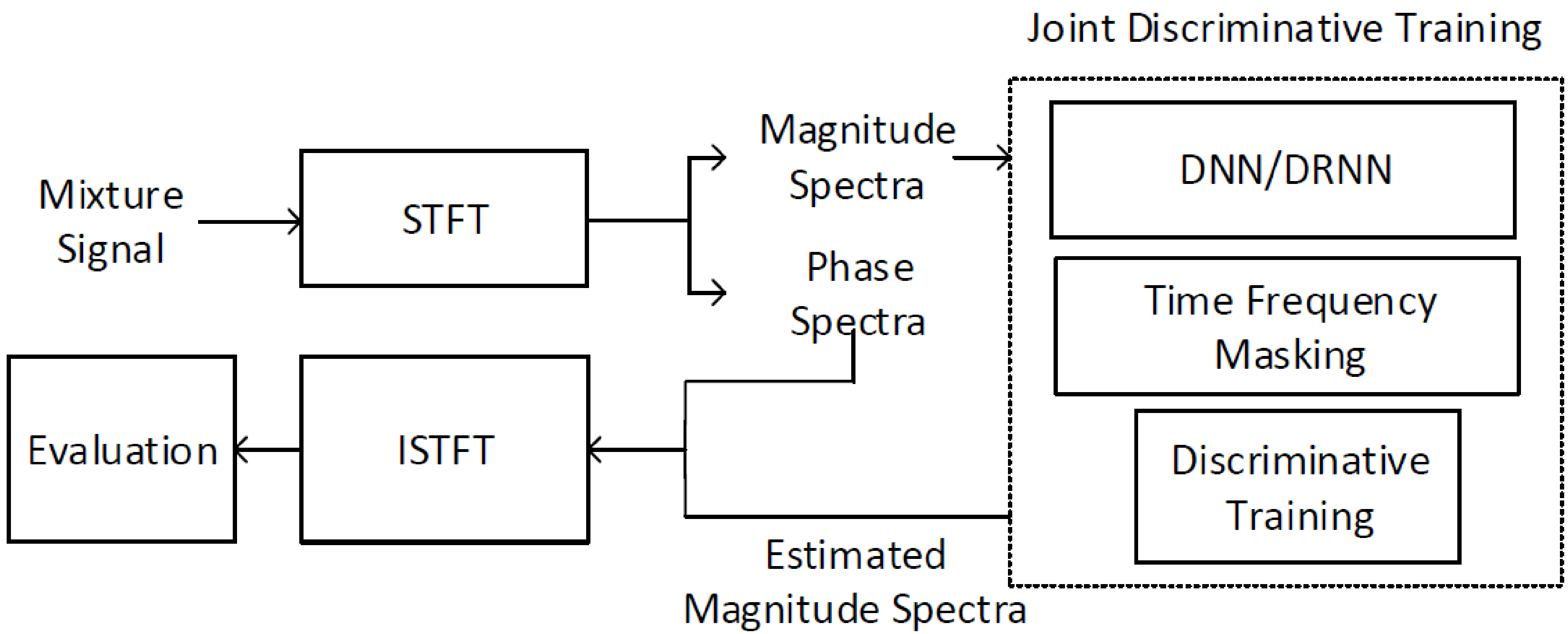
Recurrent at the l-th layer



Recurrent at all layers
(stacked RNN)

DRNN for Melody/Background Separation

- [Huang et al., 2014]



DRNN for Melody/Background Separation

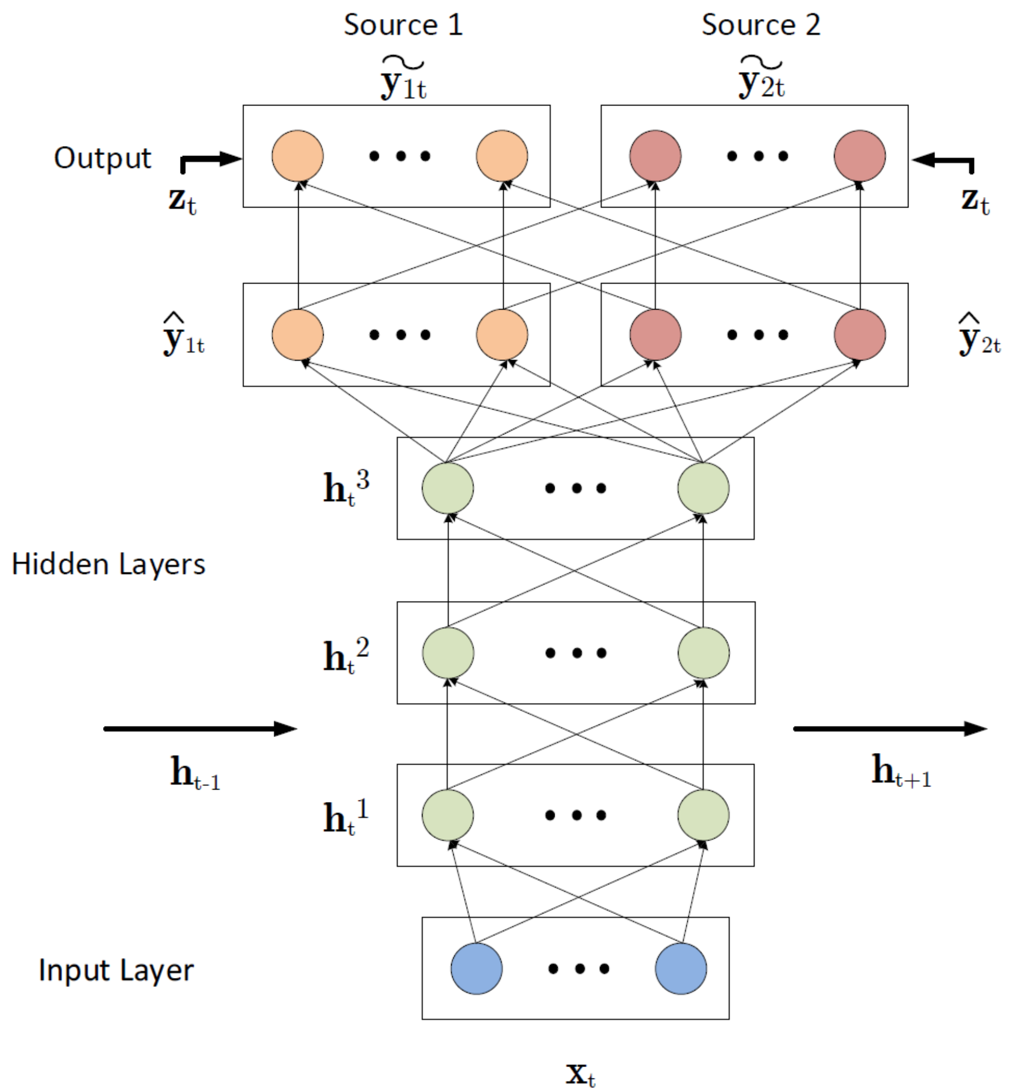
$$\tilde{\mathbf{y}}_{\mathbf{1}_t} = \frac{|\hat{\mathbf{y}}_{\mathbf{1}_t}|}{|\hat{\mathbf{y}}_{\mathbf{1}_t}| + |\hat{\mathbf{y}}_{\mathbf{2}_t}|} \odot \mathbf{z}_t$$

$$\tilde{\mathbf{y}}_{\mathbf{2}_t} = \frac{|\hat{\mathbf{y}}_{\mathbf{2}_t}|}{|\hat{\mathbf{y}}_{\mathbf{1}_t}| + |\hat{\mathbf{y}}_{\mathbf{2}_t}|} \odot \mathbf{z}_t$$

$$\mathbf{y}_t = f_o(\mathbf{h}_t^l)$$

$$\mathbf{h}_t^l = f_h(\mathbf{x}_t, \mathbf{h}_{t-1}^l)$$

Magnitude spectrum
of mixture signal



DRNN for Melody/Background Separation

- Training objectives
  - Reconstruction

$$J_{MSE} = ||\hat{\mathbf{y}}_{\mathbf{1}_t} - \mathbf{y}_{\mathbf{1}_t}||_2^2 + ||\hat{\mathbf{y}}_{\mathbf{2}_t} - \mathbf{y}_{\mathbf{2}_t}||_2^2$$

$$J_{KL} = D(\mathbf{y}_{\mathbf{1_t}}||\hat{\mathbf{y}}_{\mathbf{1}_t}) + D(\mathbf{y}_{\mathbf{2}_t}||\hat{\mathbf{y}}_{\mathbf{2}_t})$$

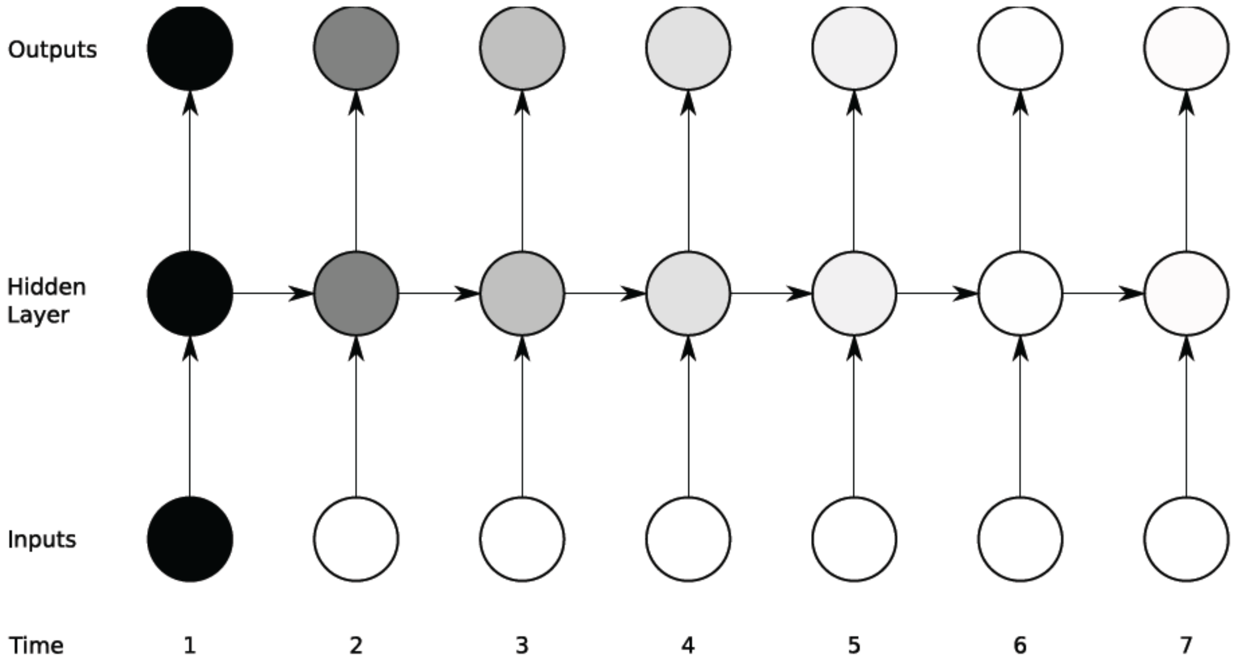
  - Reconstruction & Discrimination

$$||\hat{\mathbf{y}}_{\mathbf{1}_t} - \mathbf{y}_{\mathbf{1}_t}||_2^2 - \gamma||\hat{\mathbf{y}}_{\mathbf{1}_t} - \mathbf{y}_{\mathbf{2}_t}||_2^2 + ||\hat{\mathbf{y}}_{\mathbf{2}_t} - \mathbf{y}_{\mathbf{2}_t}||_2^2 - \gamma||\hat{\mathbf{y}}_{\mathbf{2}_t} - \mathbf{y}_{\mathbf{1}_t}||_2^2$$

$$D(\mathbf{y}_{\mathbf{1_t}}||\hat{\mathbf{y}}_{\mathbf{1}_t}) - \gamma D(\mathbf{y}_{\mathbf{1_t}}||\hat{\mathbf{y}}_{\mathbf{2}_t}) + D(\mathbf{y}_{\mathbf{2}_t}||\hat{\mathbf{y}}_{\mathbf{2}_t}) - \gamma D(\mathbf{y}_{\mathbf{2_t}}||\hat{\mathbf{y}}_{\mathbf{1}_t})$$

Vanishing Gradient Problem of RNN

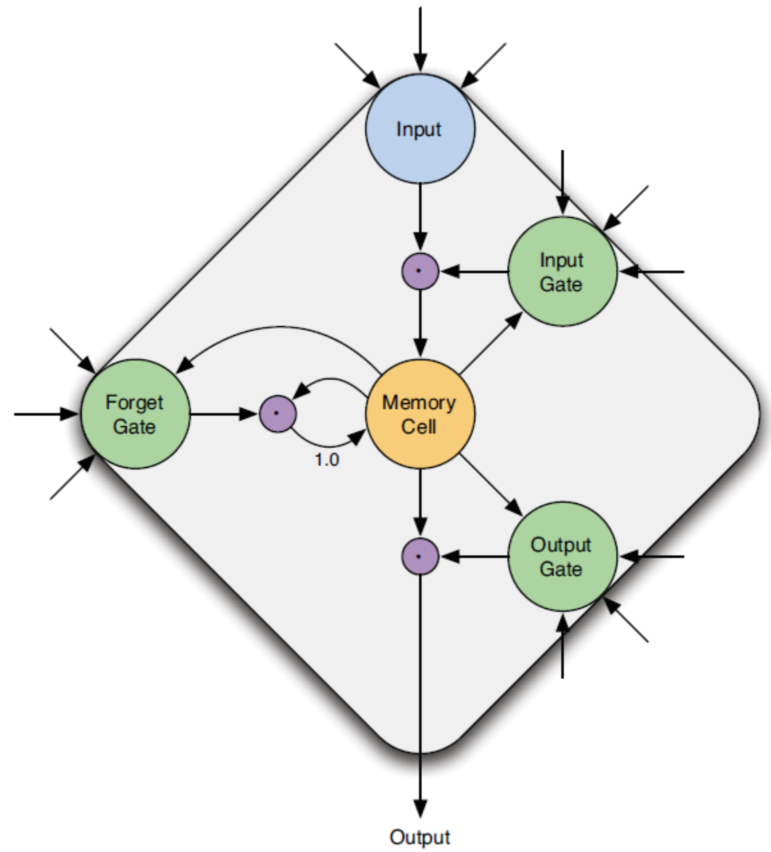
- Memory is very short



Darkness indicates the influence of input at time 1
Figure from [Graves, 2008]

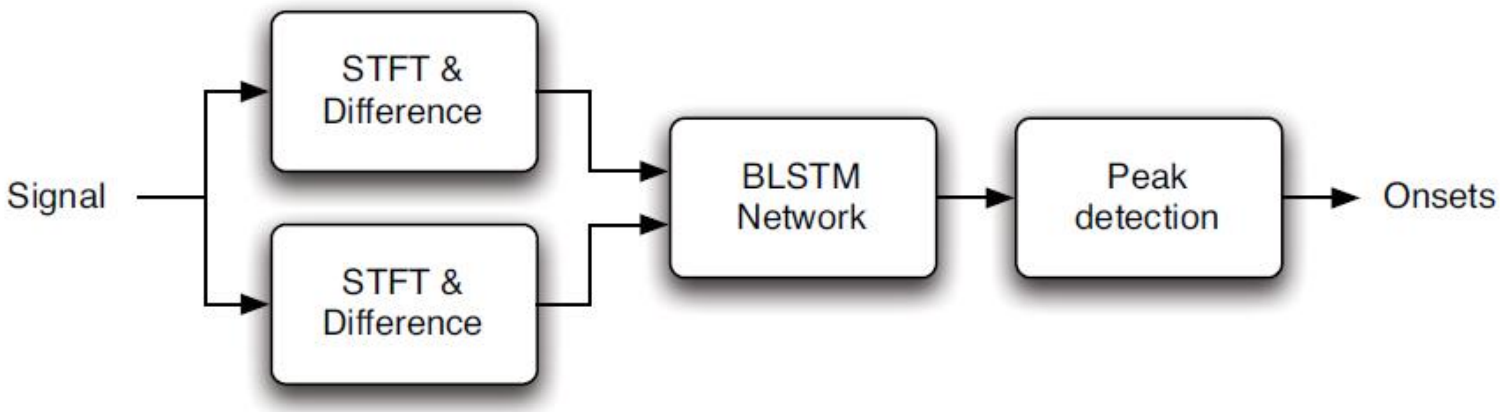
Long Short-Term Memory (LSTM)

- A memory cell
  - It can remember things for a long time



LSTM for Onset Detection

- [Eyben, et al., 2010]



LSTM for Onset Detection

Network input (Spectrogram)

Network output (onset strength curve)