Neural Networks in Signal Enhancement

Bhiksha Raj

Carnegie Mellon University

About me

- **Bhiksha Raj**
  - School of Computer Science
    Courtesy: Electrical and Computer Engineering

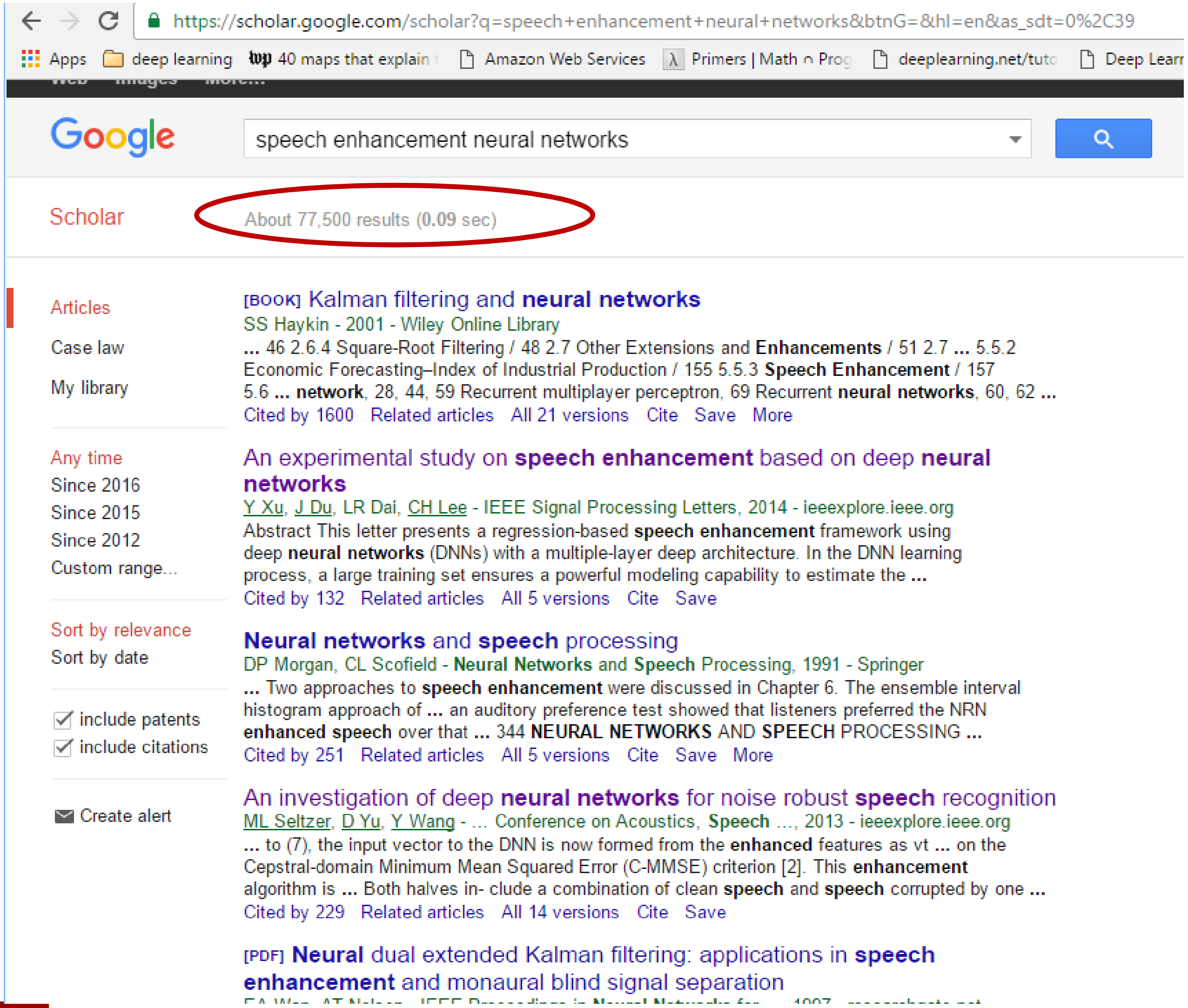
    Carnegie Mellon University

- I have worked extensively on speech recognition, speech enhancement and audio processing
  - And, of course, on neural networks
    - I teach the subject at CMU
    - Investigations on the basic principles of NNets
    - And how they may be applied to signal processing..

Neural Networks have Taken Over

- Neural networks are increasingly providing the state of the art in many pattern classification, regression, planning, and prediction tasks
  - Speech recognition
  - Image classification
  - Machine translation
  - Robot planning
  - Games
  - …
  - …

Neural Networks have Taken Over



https://scholar.google.com/scholar?q=speech+enhancement+neural+networks&btnG=&hl=en&as_sdt=0%2C39

speech enhancement neural networks

About 77,500 results (**0.09** sec)

[BOOK] Kalman filtering and **neural networks**
SS Haykin - 2001 - Wiley Online Library
... 46 2.6.4 Square-Root Filtering / 48 2.7 Other Extensions and **Enhancements** / 51 2.7 ... 5.5.2 Economic Forecasting–Index of Industrial Production / 155 5.5.3 **Speech Enhancement** / 157 5.6 ... **network**, 28, 44, 59 Recurrent multiplayer perceptron, 69 Recurrent **neural networks**, 60, 62 ...
Cited by 1600 Related articles All 21 versions Cite Save More

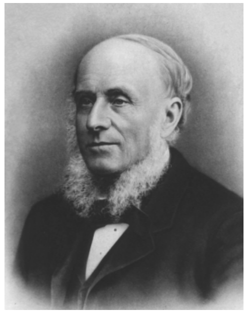
An experimental study on **speech enhancement** based on deep **neural networks**
Y Xu, J Du, LR Dai, CH Lee - IEEE Signal Processing Letters, 2014 - ieeexplore.ieee.org
Abstract This letter presents a regression-based **speech enhancement** framework using deep **neural networks** (DNNs) with a multiple-layer deep architecture. In the DNN learning process, a large training set ensures a powerful modeling capability to estimate the ...
Cited by 132 Related articles All 5 versions Cite Save

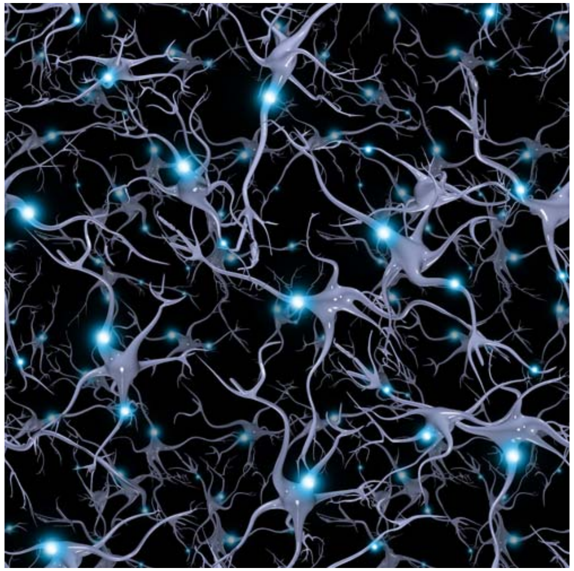
Neural networks and **speech** processing
DP Morgan, CL Scofield - **Neural Networks** and **Speech** Processing, 1991 - Springer
... Two approaches to **speech enhancement** were discussed in Chapter 6. The ensemble interval histogram approach of ... an auditory preference test showed that listeners preferred the NRN **enhanced speech** over that ... 344 **NEURAL NETWORKS** AND **SPEECH** PROCESSING ...
Cited by 251 Related articles All 5 versions Cite Save More

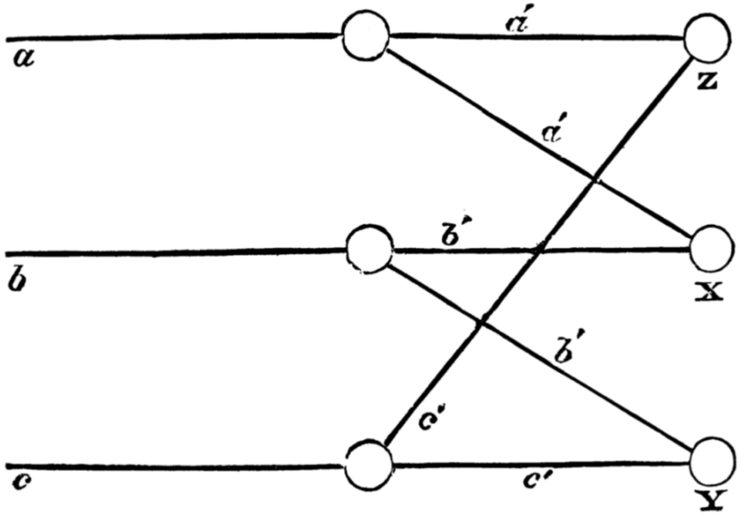
An investigation of deep **neural networks** for noise robust **speech** recognition
ML Seltzer, D Yu, Y Wang - ... Conference on Acoustics, **Speech** ..., 2013 - ieeexplore.ieee.org
... to (7), the input vector to the DNN is now formed from the **enhanced** features as vt ... on the Cepstral-domain Minimum Mean Squared Error (C-MMSE) criterion [2]. This **enhancement** algorithm is ... Both halves in- clude a combination of clean **speech** and **speech** corrupted by one ...
Cited by 229 Related articles All 14 versions Cite Save

[PDF] **Neural** dual extended Kalman filtering: applications in **speech enhancement** and monaural blind signal separation

Connectionism

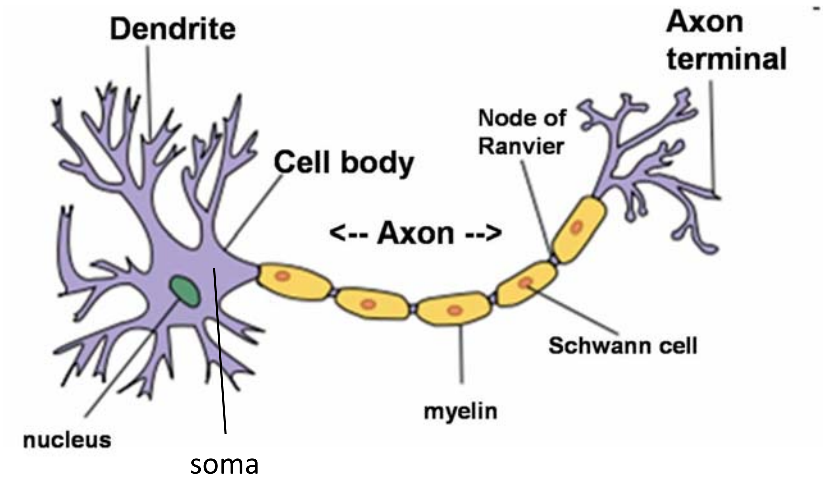


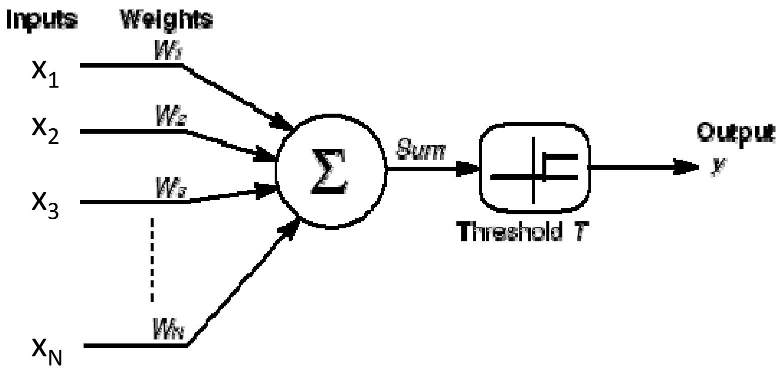




- Alexander Bain: 1873
  - The magic is in the connections!
  - An early computational neural network model

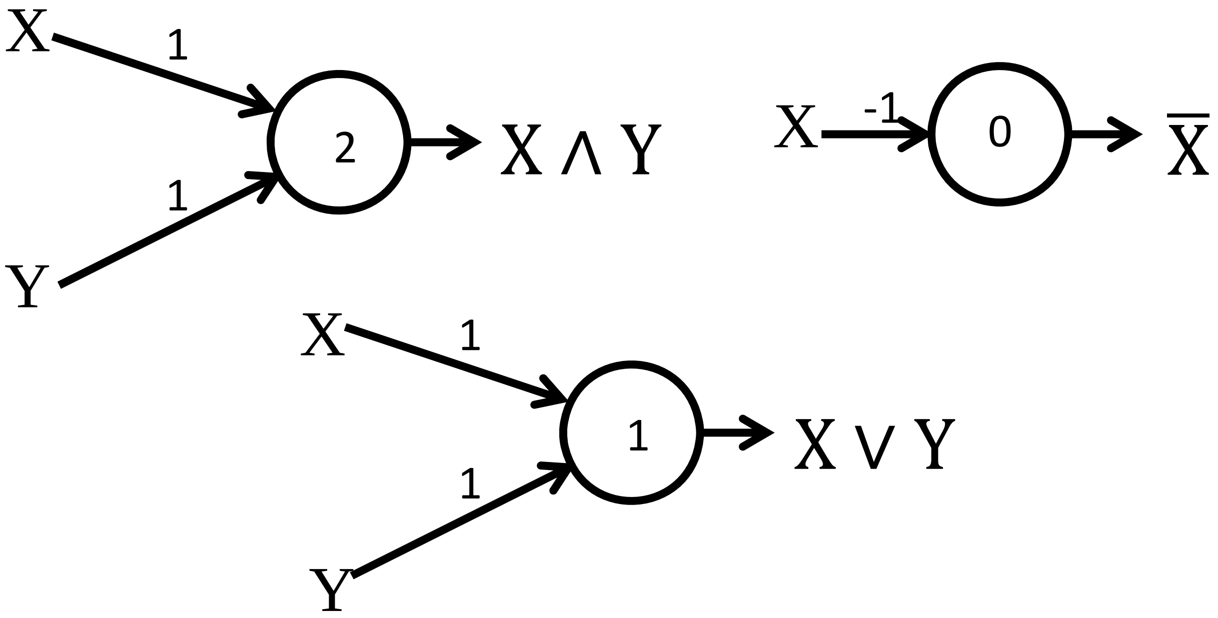
The Computational Model of the Neuron





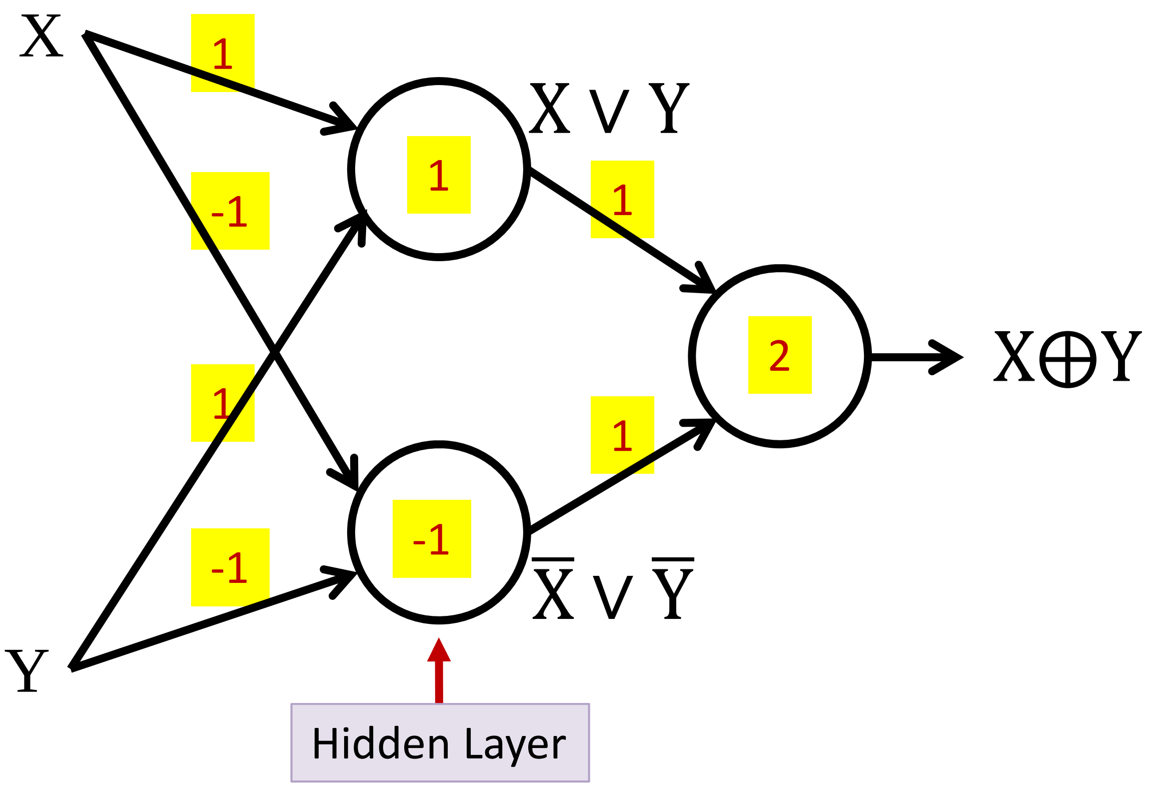
- Left: Biological Neuron
- Right: The computational model

Perceptron as a Boolean gate



- The basic Perceptron
  - Simple Boolean unit
    - **The gates can combine *any* number of inputs**
    - Including *negated* inputs (just flip the sign of the weight)
  - Cannot represent an XOR

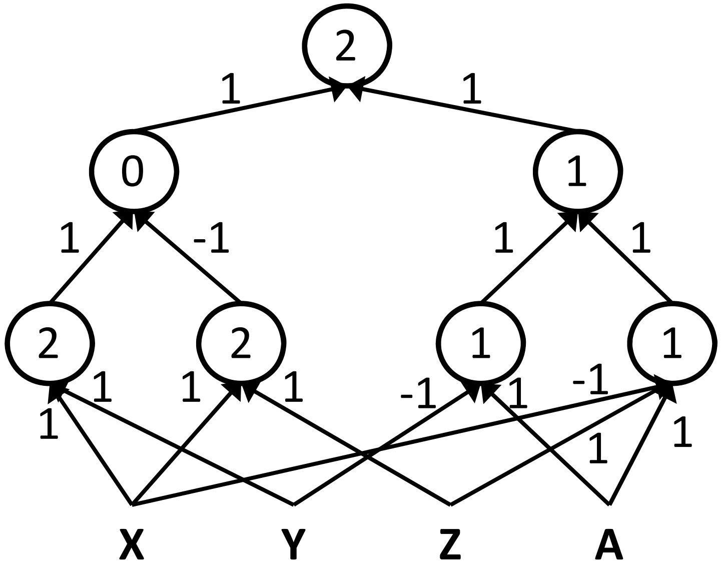
MLP as a Boolean function



- Multi layer perceptron
  - The first layer is a “hidden” layer

Constructing a Boolean Function

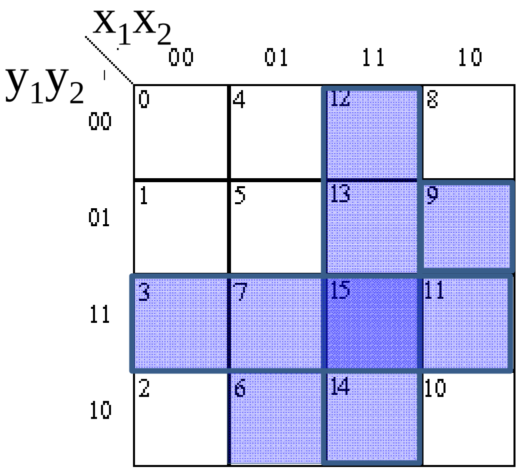
$((A\&\bar{X}\&Z)|(A\&\bar{Y}))\&((X\ \&\ Y)|\overline{(X\&Z)})$



- A more complex Boolean function
- Has *two* hidden layers
- *Any* Boolean function can be composed using a multi-layer perceptron

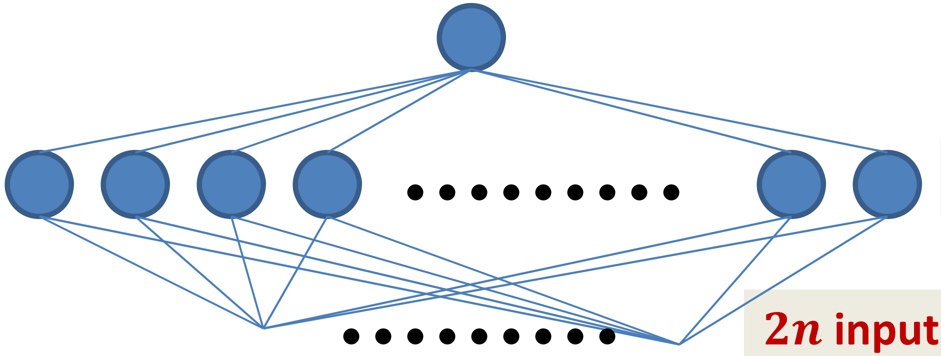
Carnegie Mellon

Constructing Boolean functions with only one hidden layer



$x_1x_2\bar{y}_1\bar{y}_2 + x_1x_2\bar{y}_1y_2 + x_1\bar{x}_2\bar{y}_1y_2 + \ \dots$

9 terms



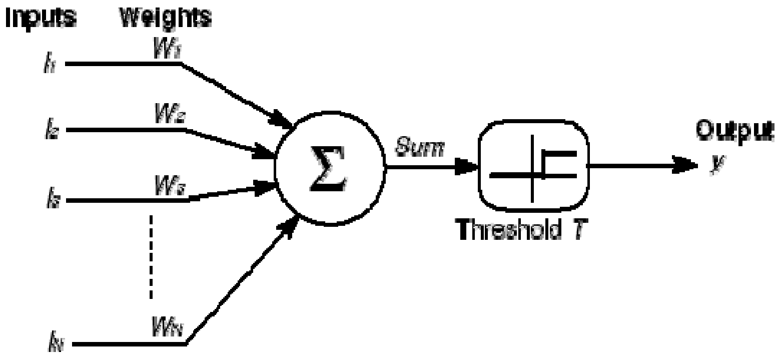
2^n units

$2n$ inputs

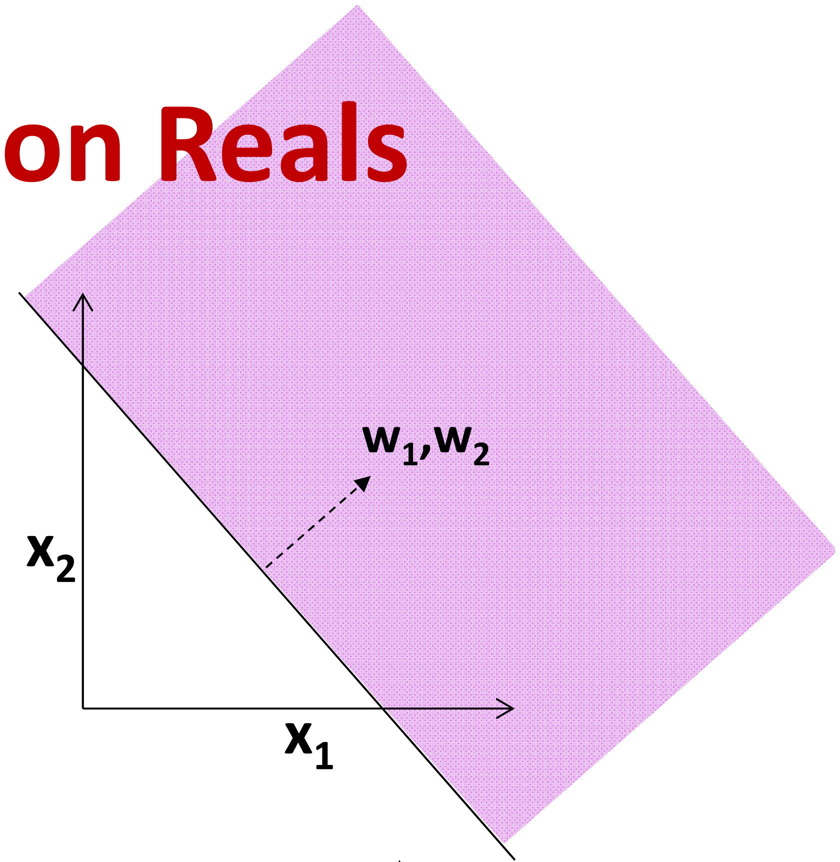
- *Any* Boolean formula can be expressed by an MLP with **one hidden layer**
  - Any Boolean formula can be expressed in Conjunctive Normal Form
- The one hidden layer can be exponentially wide
  - But the same formula can be obtained with a much smaller network if we have *multiple* hidden layers

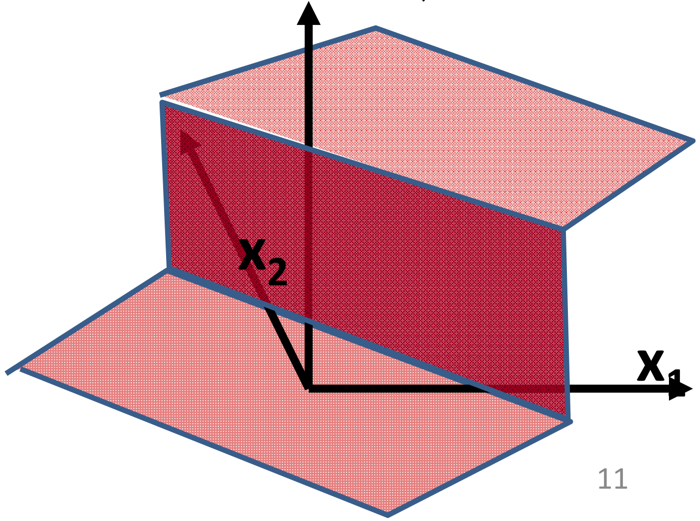
Carnegie Mellon

A Perceptron on Reals



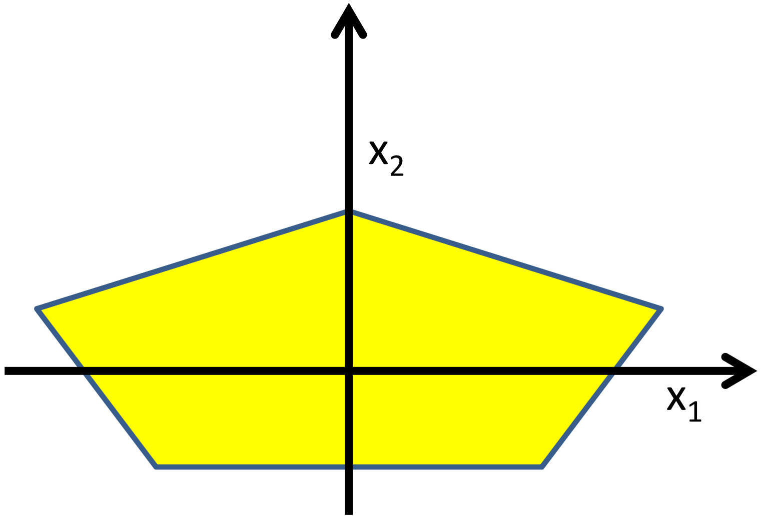
$$y = \begin{cases} 1 \ if \ \sum_i w_i \mathrm{x}_i \geq T \\ 0 \ else \end{cases}$$





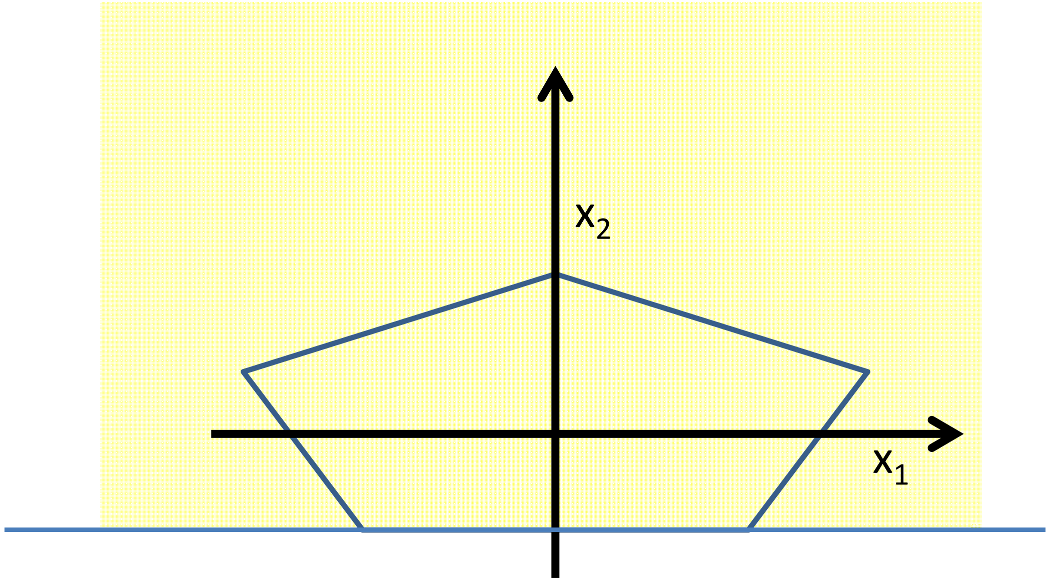
- A perceptron operates on *real*-valued vectors
  - This is just a linear classifier

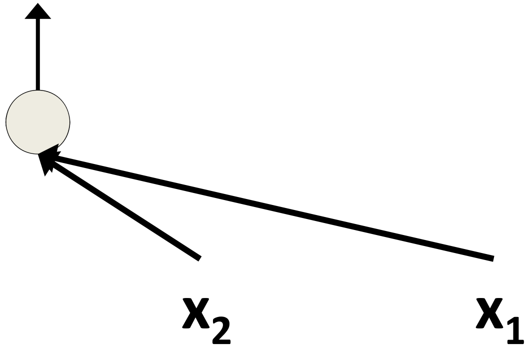
Booleans over the reals



- The network must fire if the input is in the coloured area

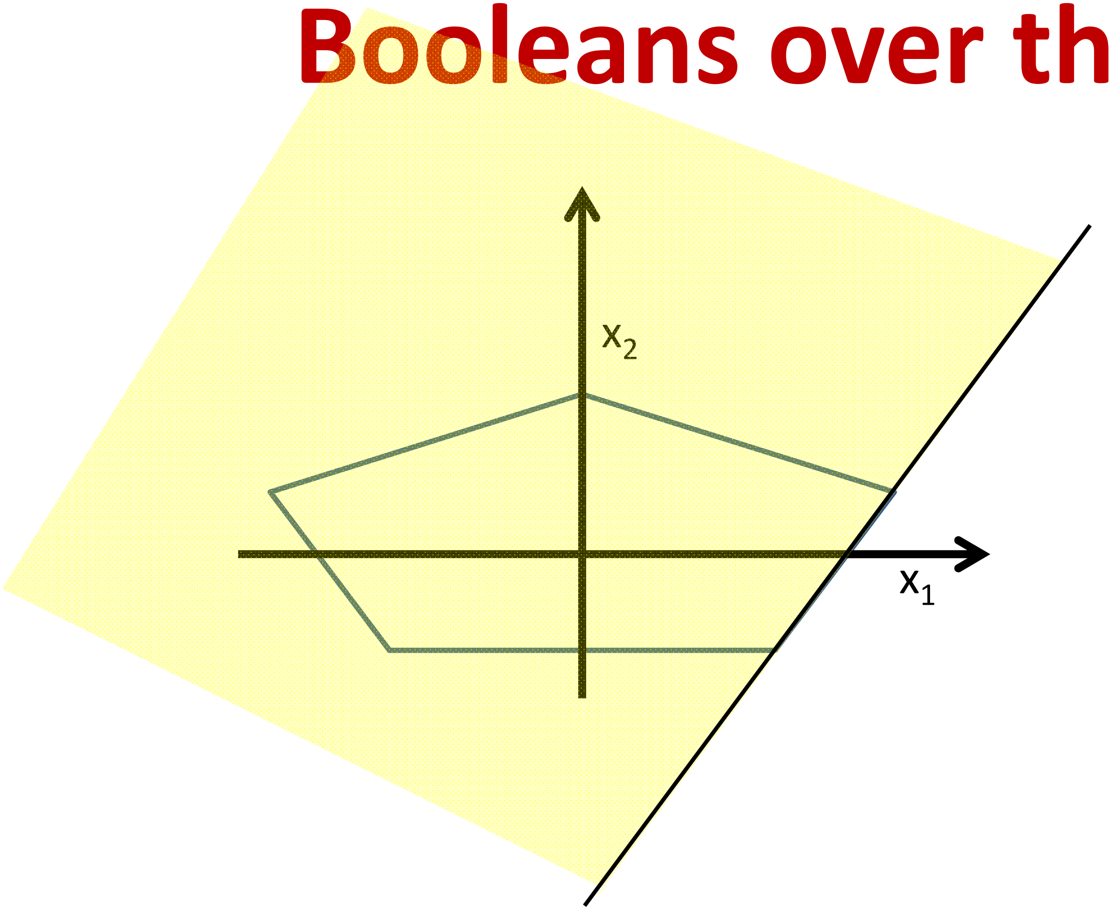
Booleans over the reals

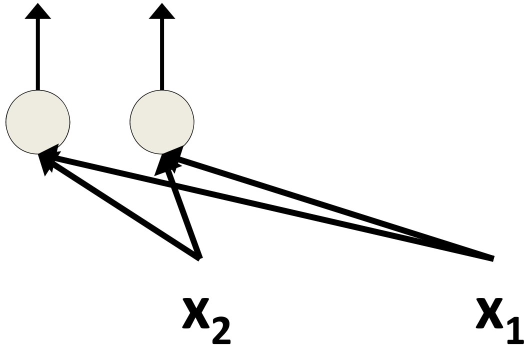




- The network must fire if the input is in the coloured area

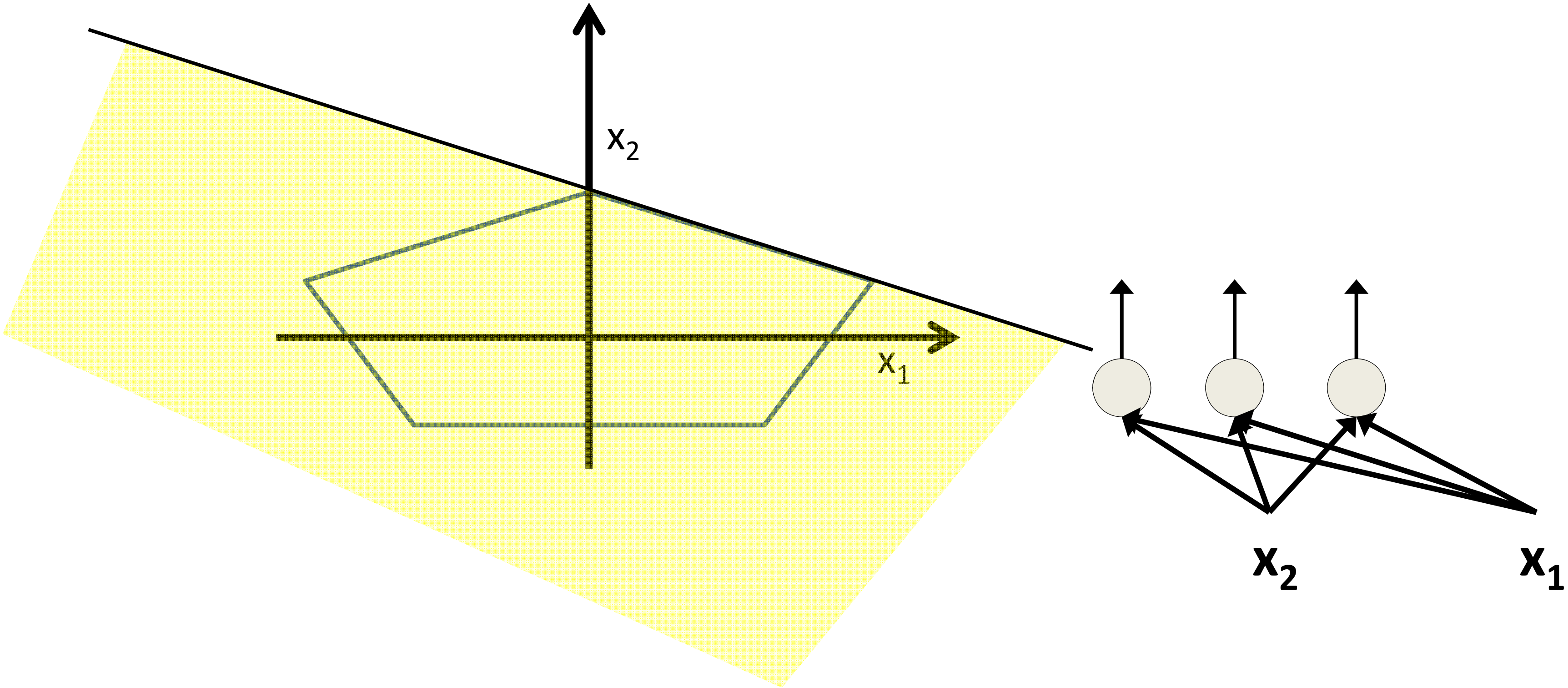
Booleans over the reals





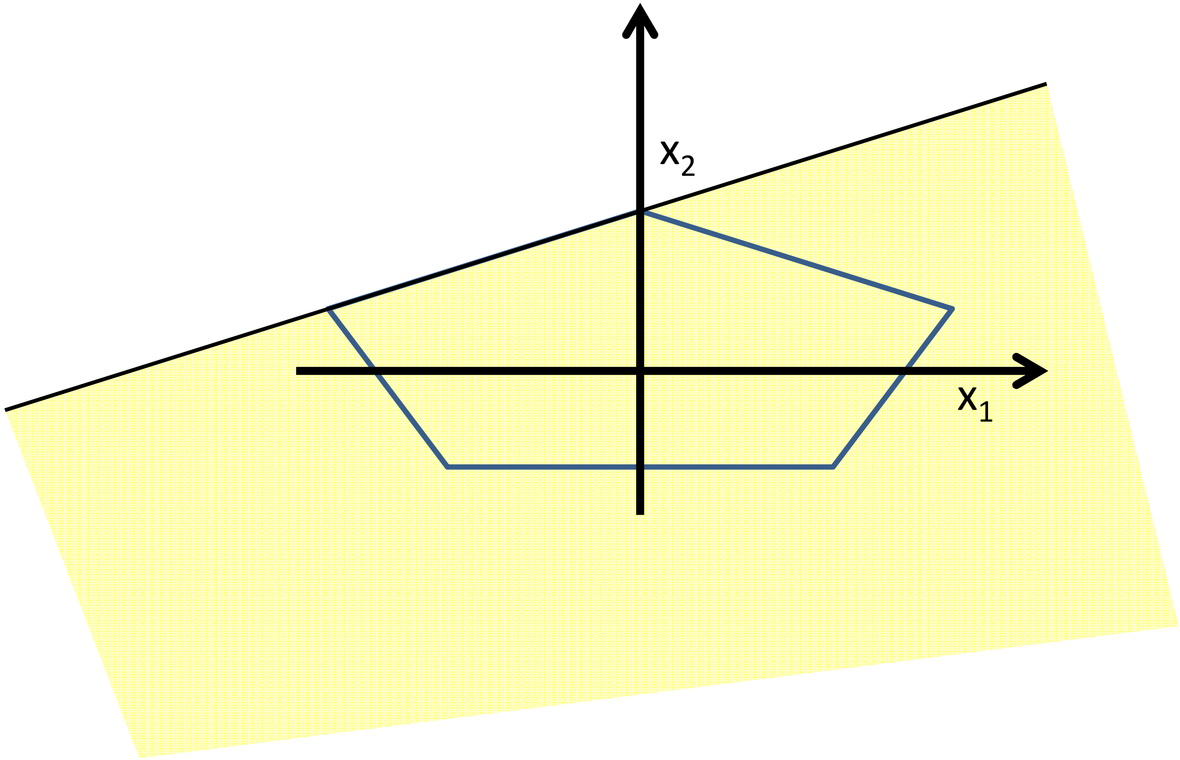
- The network must fire if the input is in the coloured area

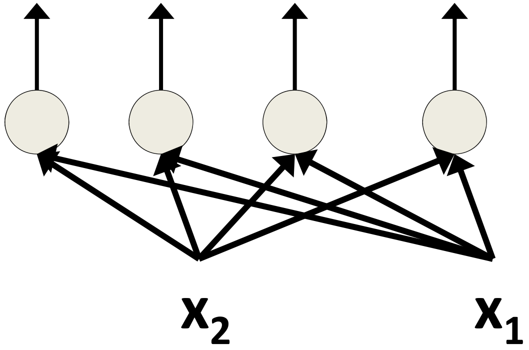
Booleans over the reals



- The network must fire if the input is in the coloured area

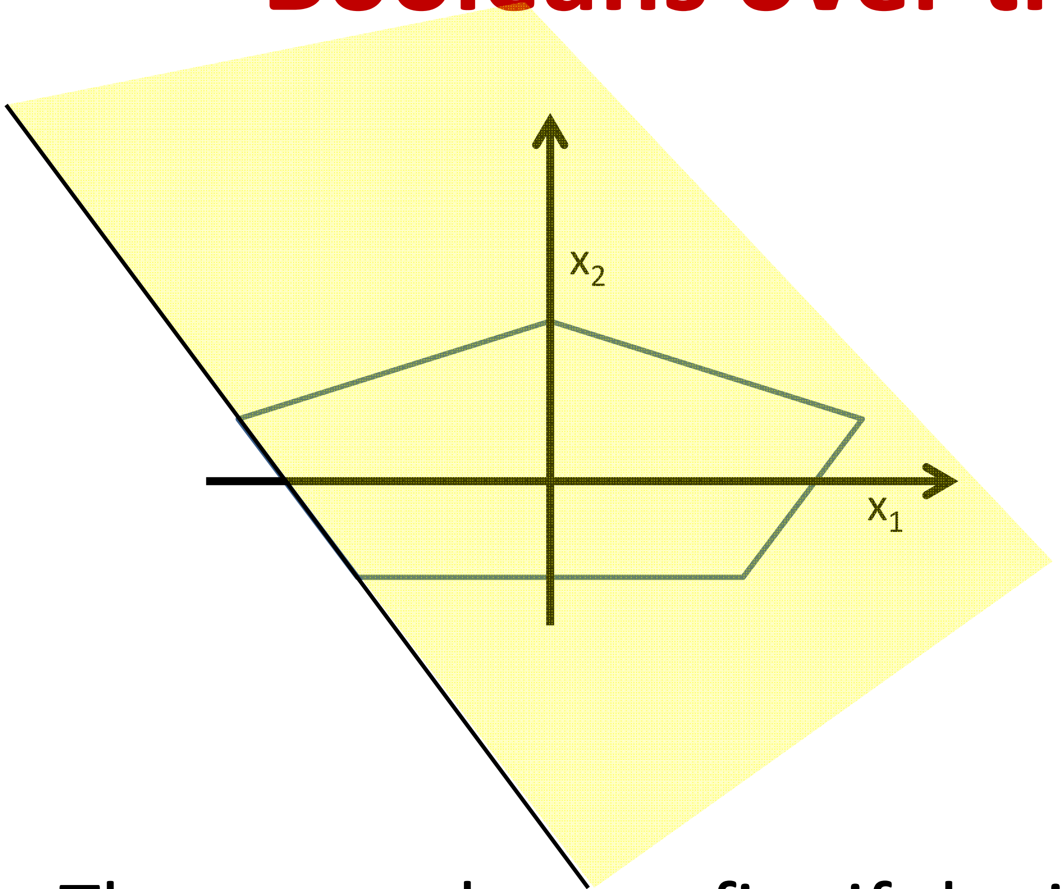
Booleans over the reals

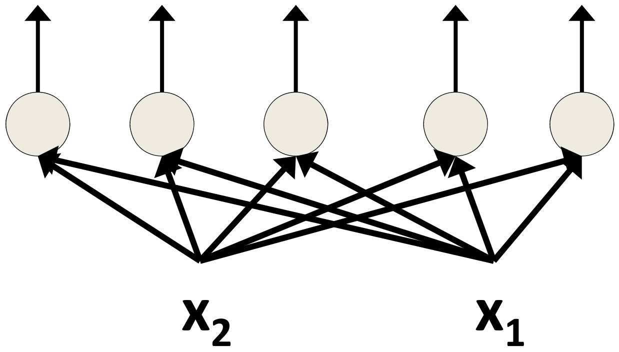




- The network must fire if the input is in the coloured area

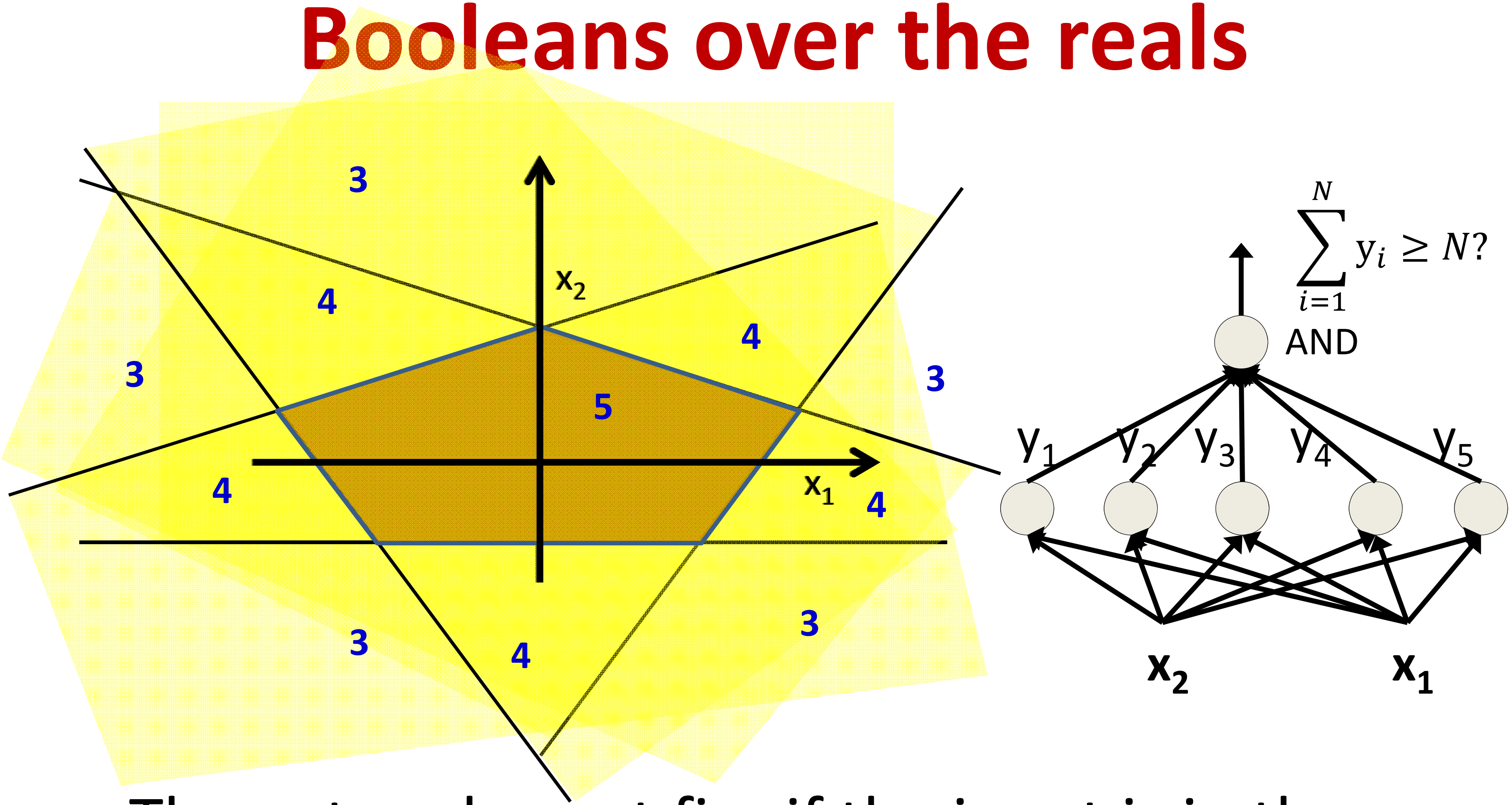
Booleans over the reals





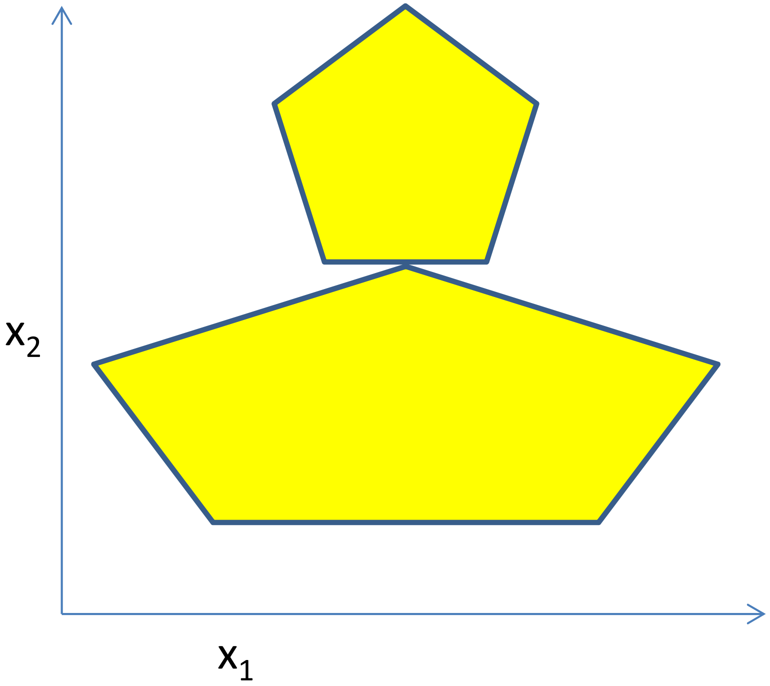
- The network must fire if the input is in the coloured area

Booleans over the reals



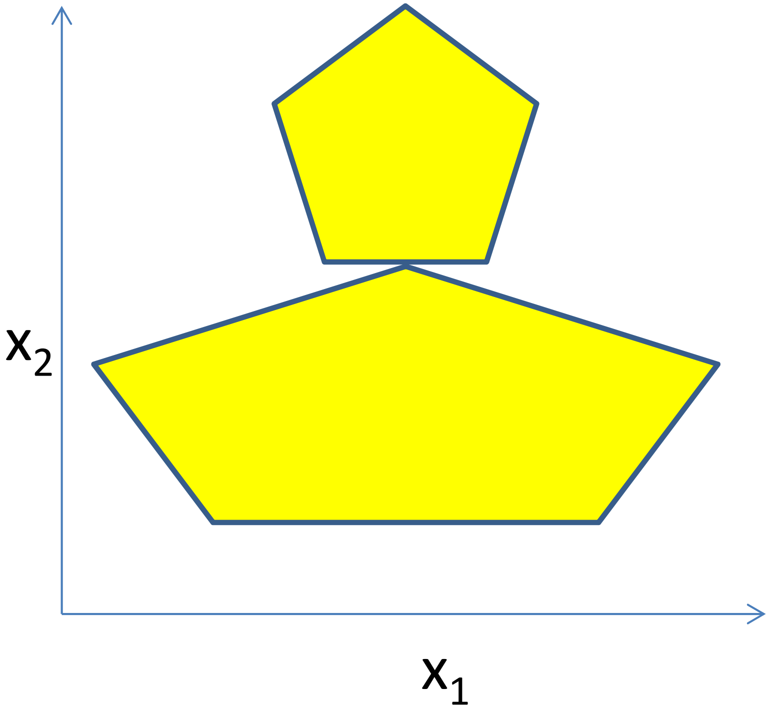
- The network must fire if the input is in the coloured area

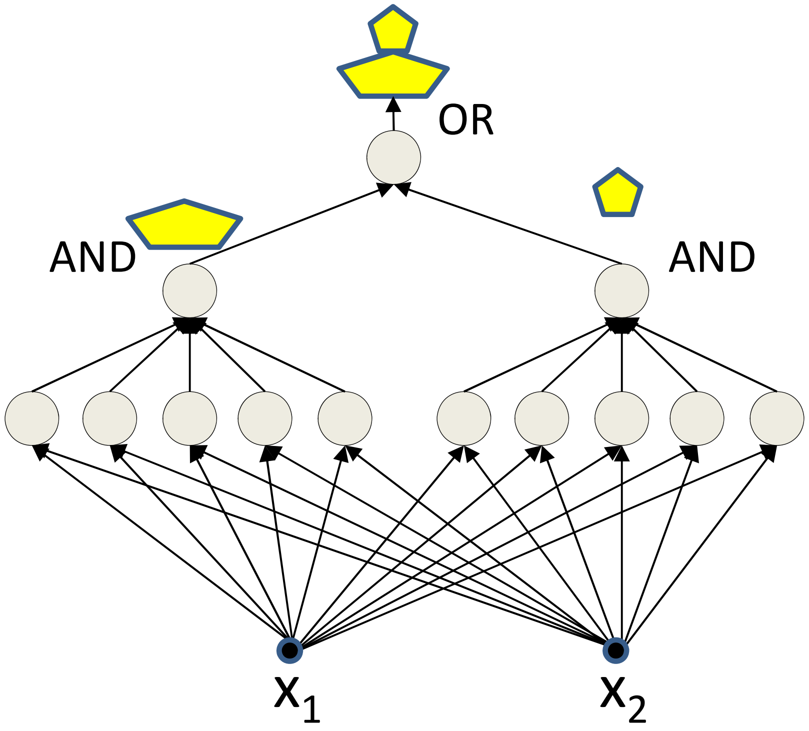
Booleans over the reals



- The network must fire if the input is in the coloured area

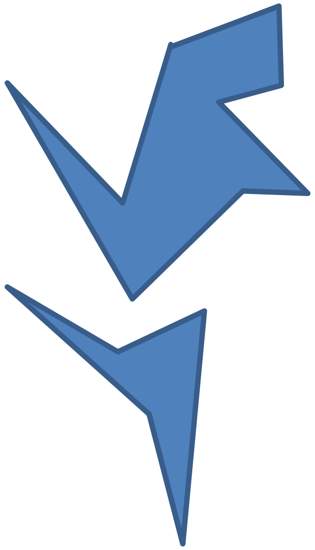
Booleans over the reals

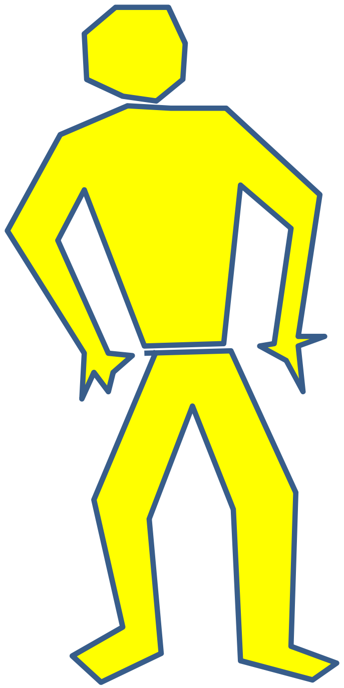


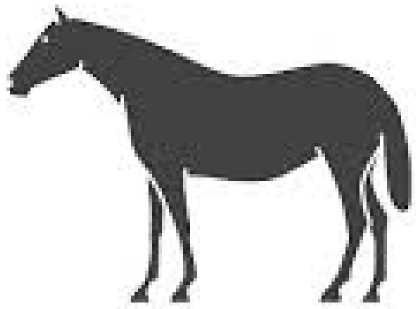


- “OR” two polygons
- A third layer is required

How Complex Can it Get

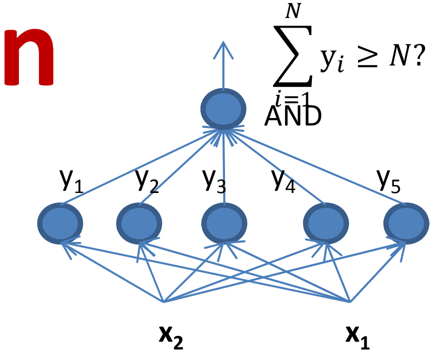


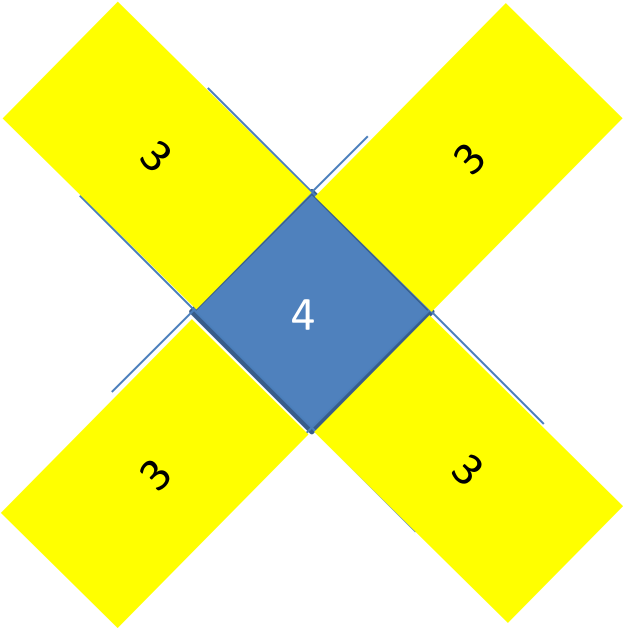


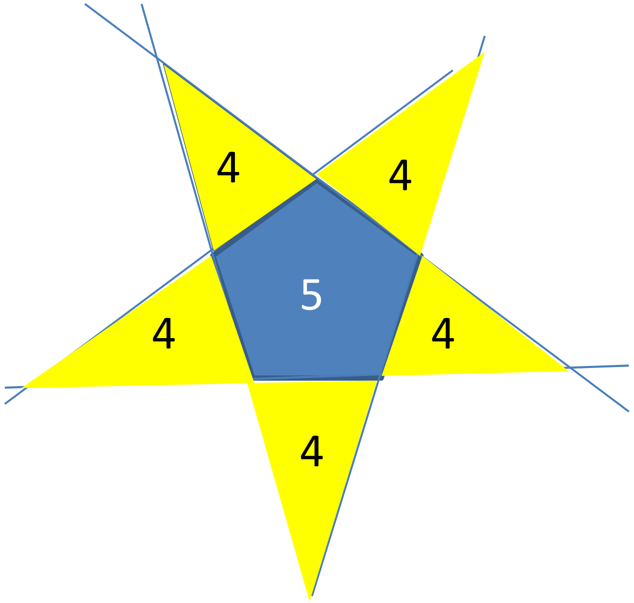


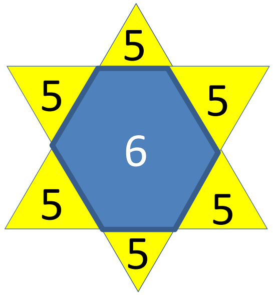
- An arbitrarily complex figure
- Basically any Boolean function over the basic linear boundaries

Composing a polygon



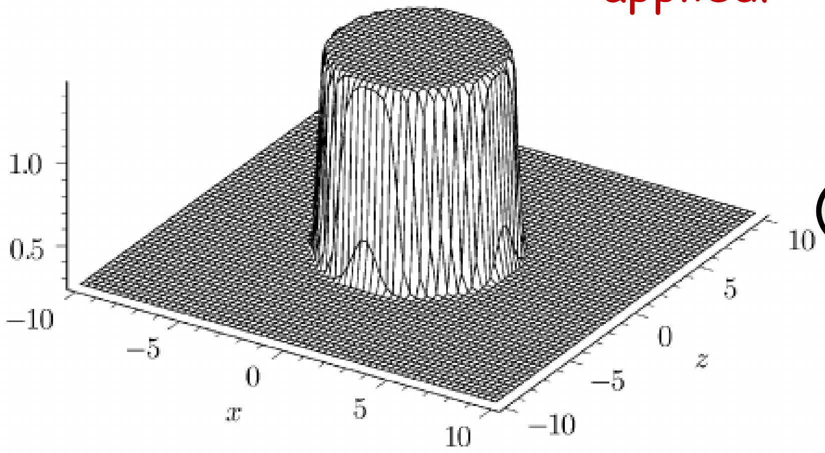


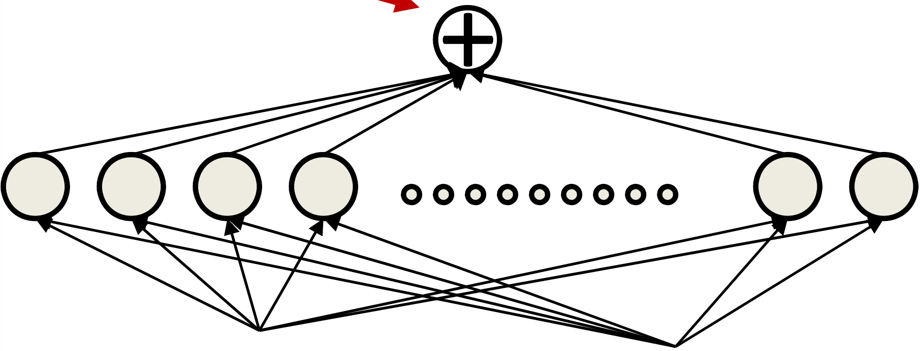




- The polygon net
- Increasing the number of sides shrinks the area outside the polygon that have sum close to N

Composing a circle

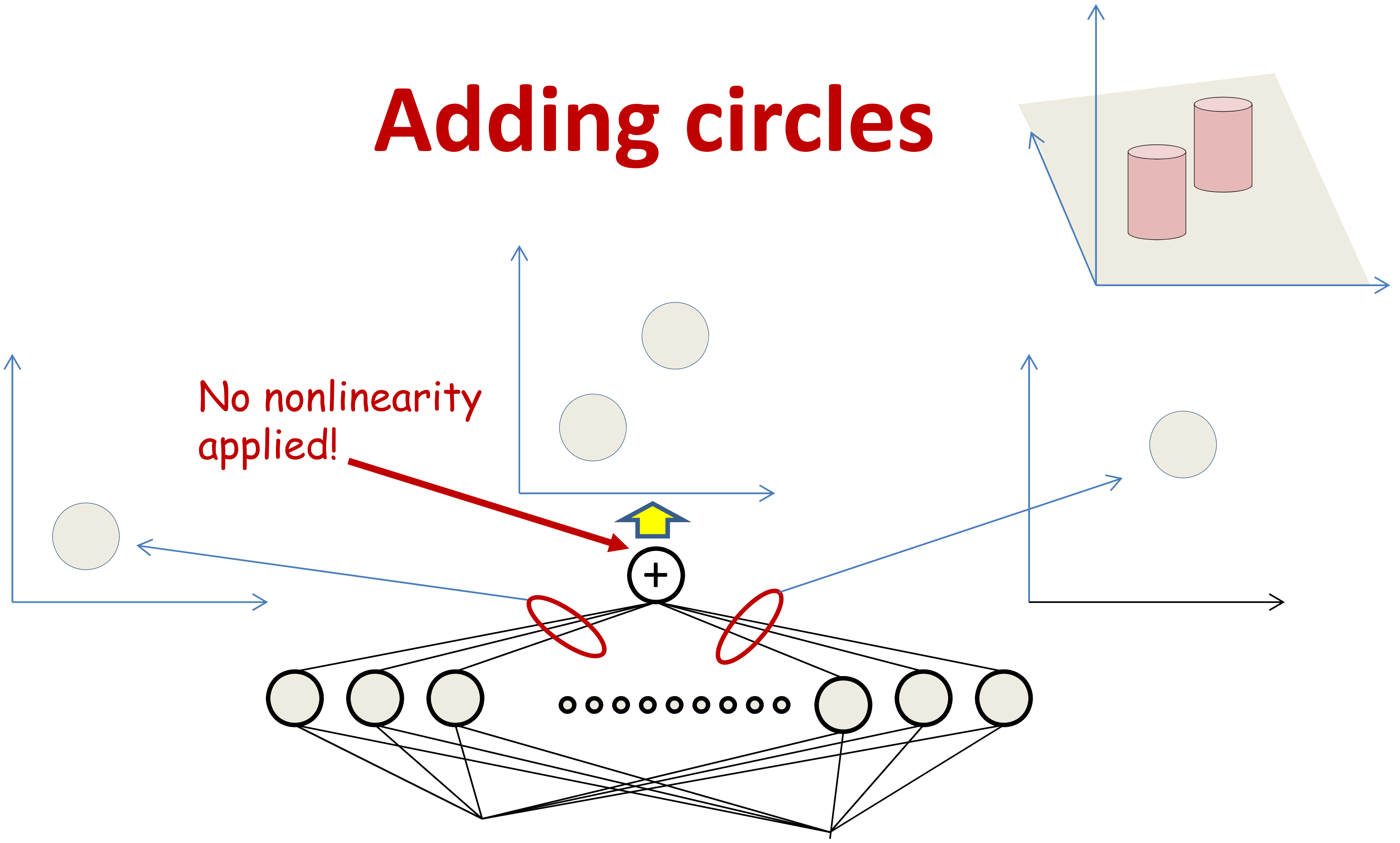




No nonlinearity applied!

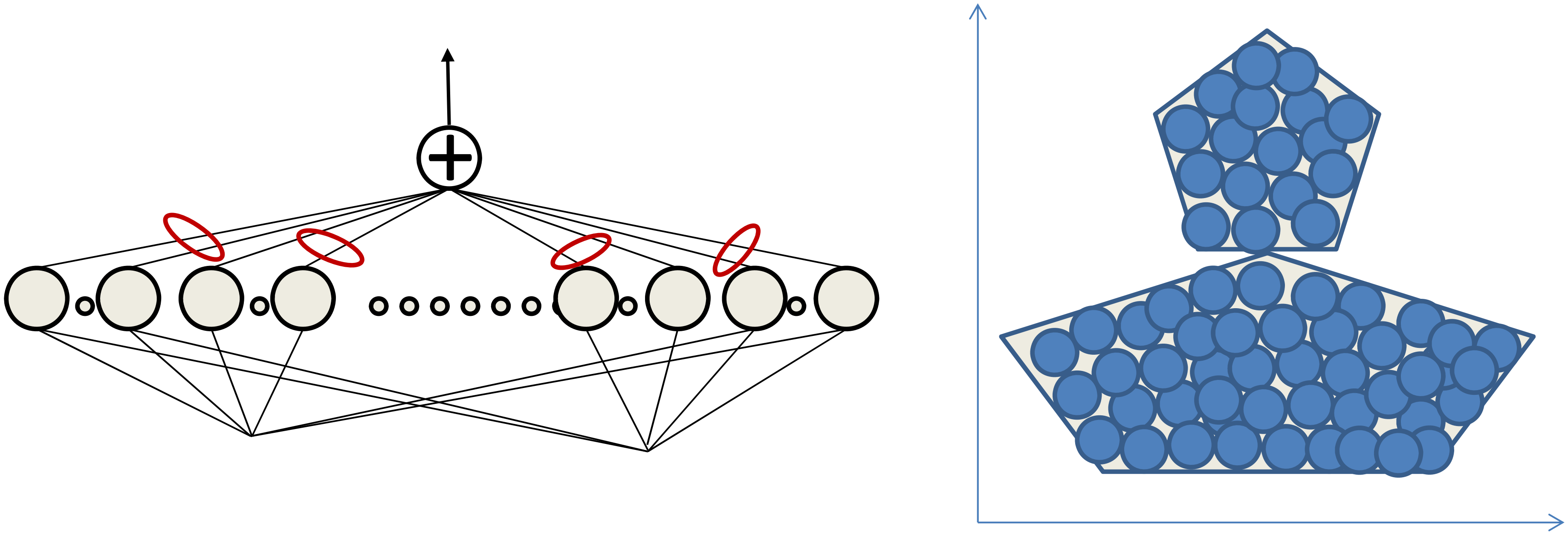
- The circle net
  - Very large number of neurons
  - Circle can be of arbitrary diameter, at any location
  - **Achieved *without using a thresholding function!!***

Adding circles



- The “sum” of two circles sub nets is exactly a net with output 1 if the input falls within either circle

Composing an arbitrary figure

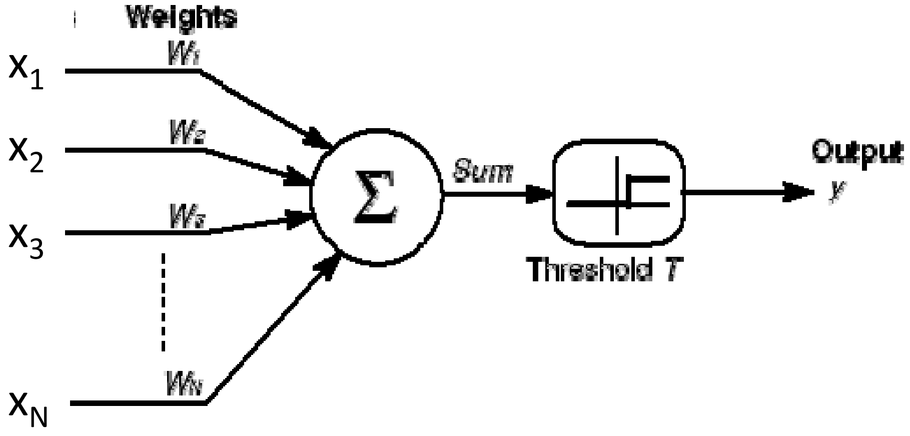


- Just fit in an arbitrary number of circles
  - More accurate approximation with greater number of smaller circles
  - A lesson here that we will refer to again shortly..

Story so far..

- **Multi-layer perceptrons are *Boolean* networks**
  - They represent Boolean functions over linear boundaries
  - They can approximate *any* boundary
    - Using a sufficiently large number of linear units
  - Complex Boolean functions are better modeled with more layers
  - Complex boundaries are more compactly represented using more layers

Lets look at the weights



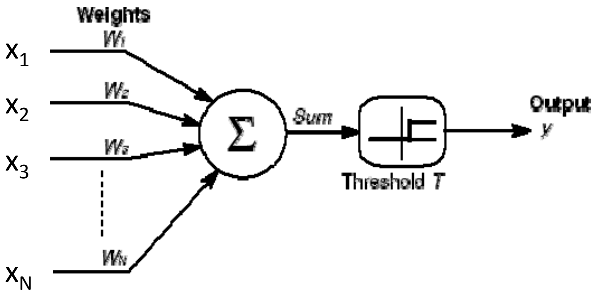
$$y = \begin{cases} 1 \; if \; \sum_i w_i \mathrm{x}_i \geq T \\ 0 \; else \end{cases}$$

$$y = \begin{cases} 1 \; if \; \mathbf{x}^T \mathbf{w} \geq T \\ 0 \; else \end{cases}$$

- What do the *weights* tell us?
  - The neuron fires if the inner product between the weights and the inputs exceeds a threshold

Carnegie Mellon

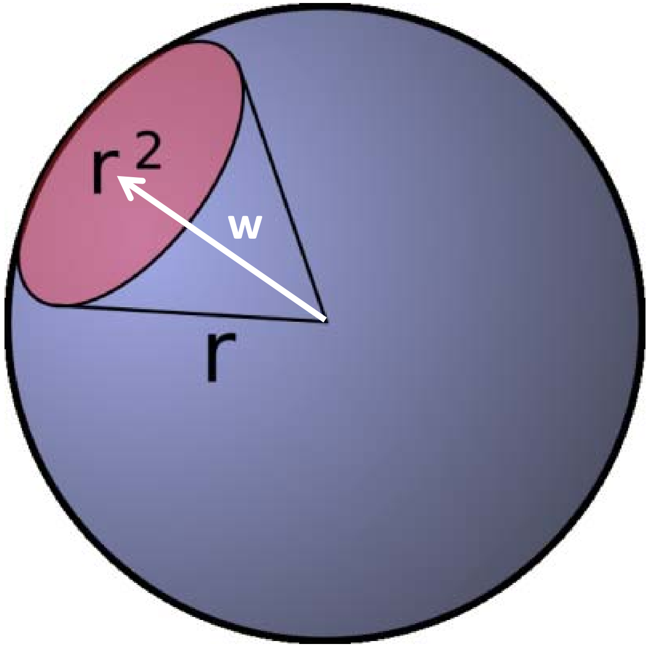
The weight as a "template"



$$X^T W > T$$

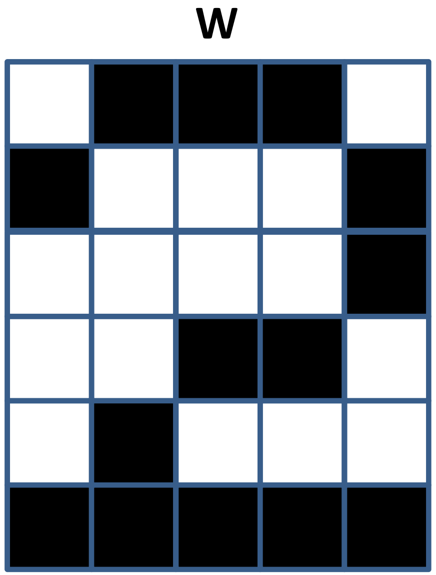
$$\cos\theta > \frac{T}{|X|}$$

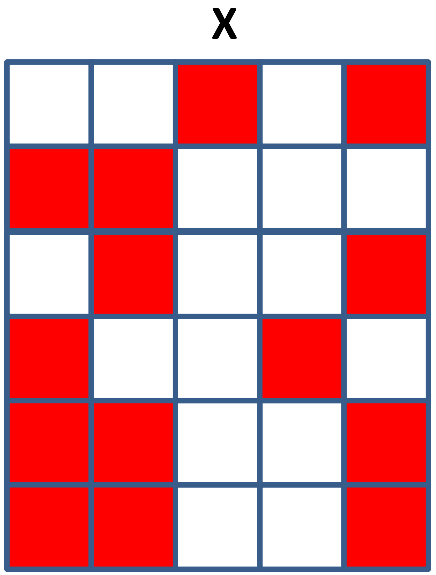
$$\theta < \cos^{-1}\left(\frac{T}{|X|}\right)$$



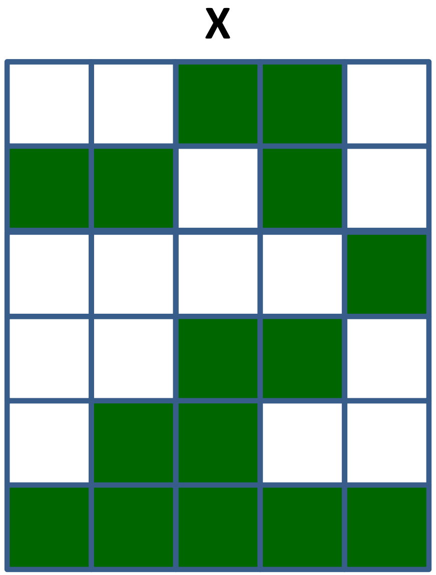
- The perceptron fires if the input is within a specified angle of the weight
  - Represents a convex region on the surface of the sphere!
  - The network is a Boolean function over these regions.
    - The overall decision region can be arbitrarily nonconvex
- Neuron fires if the input vector is close enough to the weight vector.
  - If the input pattern matches the weight pattern closely enough

The weight as a template





Correlation = 0.57

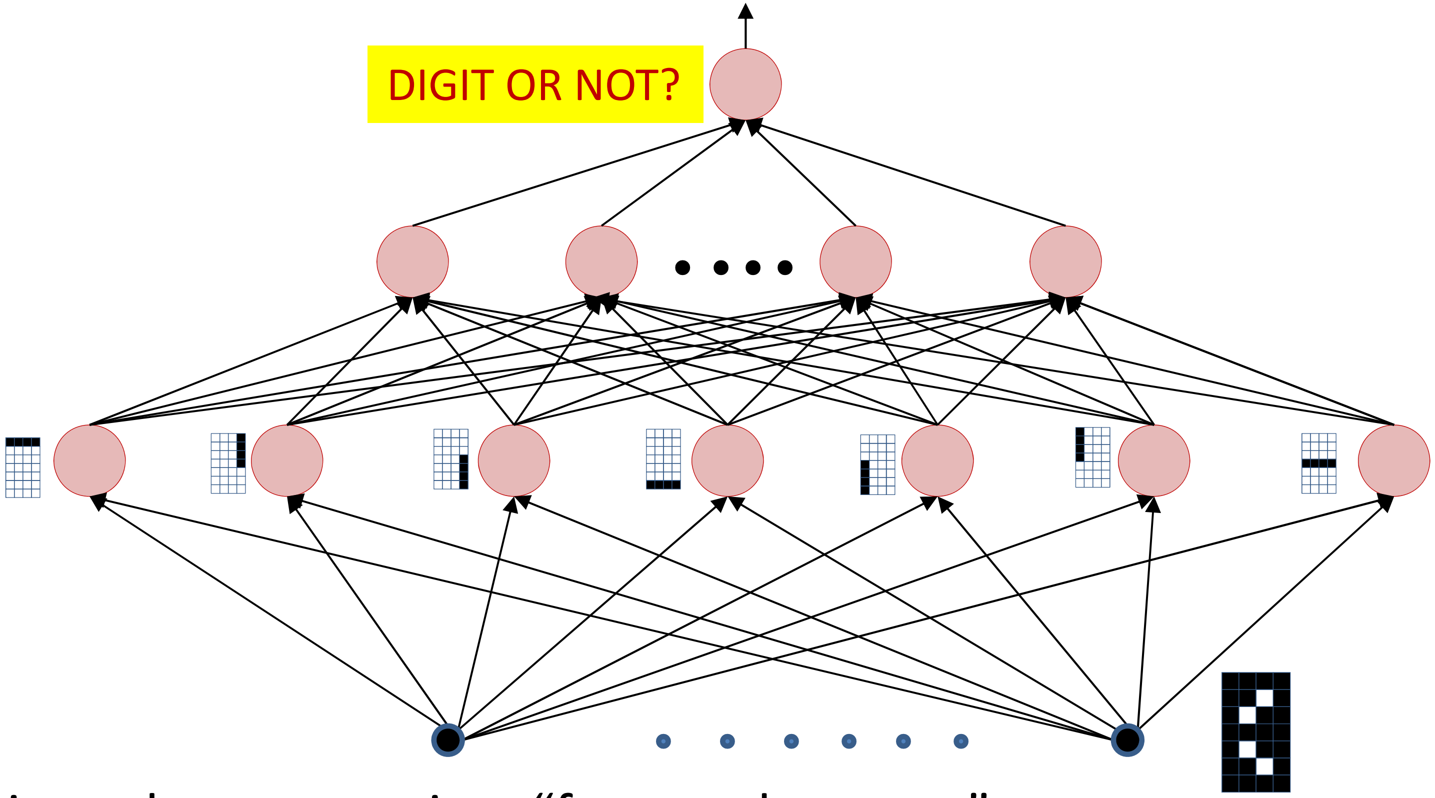


Correlation = 0.82

$$y = \begin{cases} 1\ if \sum_i w_i \mathrm{x}_i \geq T \\ 0\ else \end{cases}$$

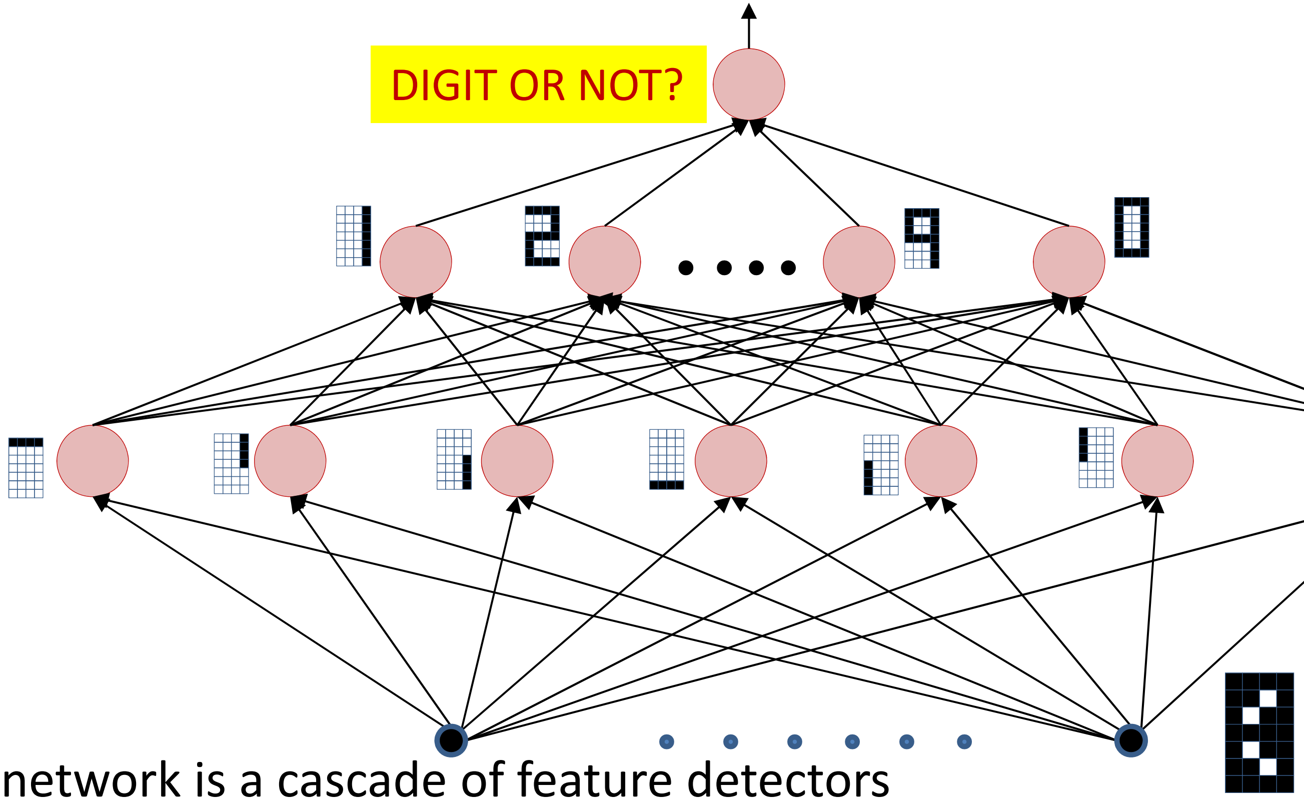
- If the *correlation* between the weight pattern and the inputs exceeds a threshold, fire
- The perceptron is a *correlation filter!*

The MLP as a Boolean function over feature detectors



- The input layer comprises “feature detectors”
  - Detect if certain patterns have occurred in the input
- The network is a Boolean function over the feature detectors
- I.e. it is important for the *first* layer to capture relevant patterns

The MLP as a cascade of feature detectors

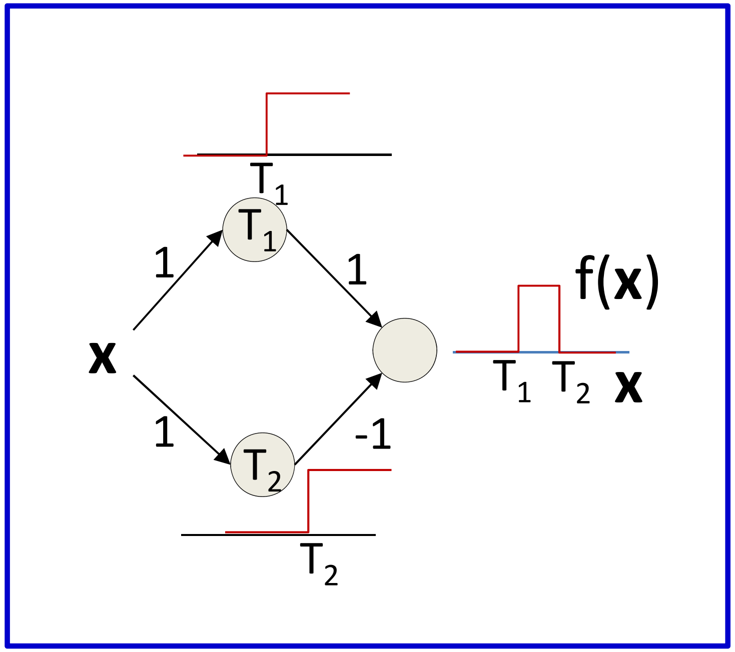


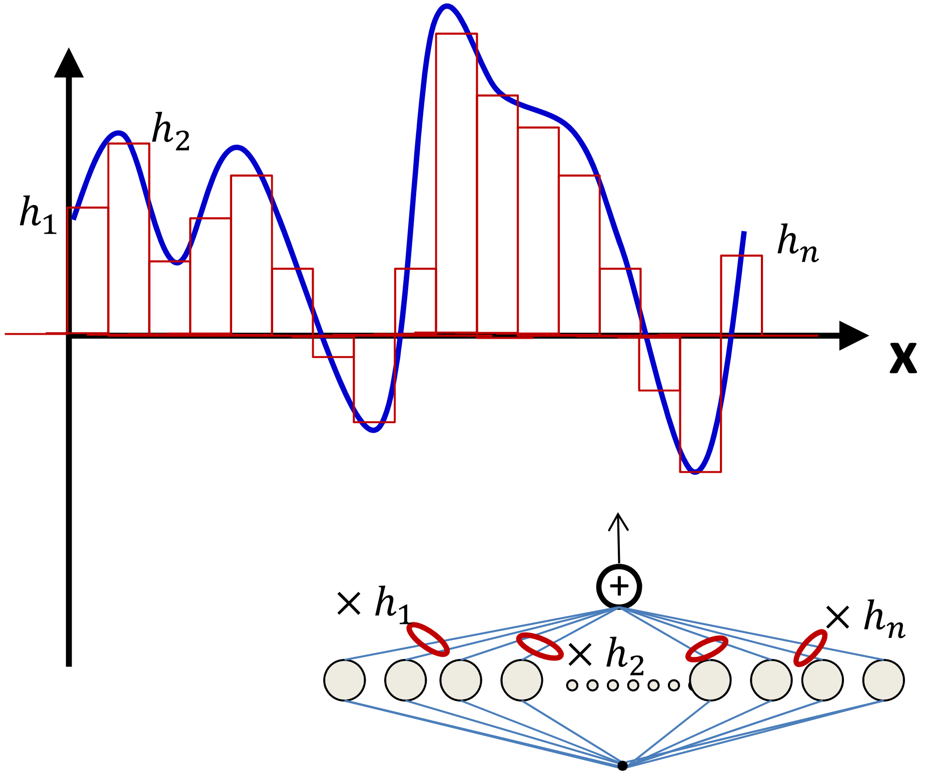
- The network is a cascade of feature detectors
  - **Higher level neurons compose complex templates from features represented by lower-level neurons**
- **Risk in this perspective**: Upper level neurons may be performing "OR"
  - Looking for a *choice* of compound patterns

Story so far

- **MLPs are Boolean machines**
  - They represent Boolean functions over linear boundaries
  - They can represent arbitrary boundaries
- **Perceptrons are correlation filters**
  - They detect patterns in the input
- MLPs are Boolean formulae over patterns detected by perceptrons
  - Higher-level perceptrons may also be viewed as feature detectors
- **Extra: MLP in classification**
  - The network will fire if the combination of the detected basic features matches an "acceptable" pattern for a desired class of signal
    - E.g. Appropriate combinations of (Nose, Eyes, Eyebrows, Cheek, Chin) → Face

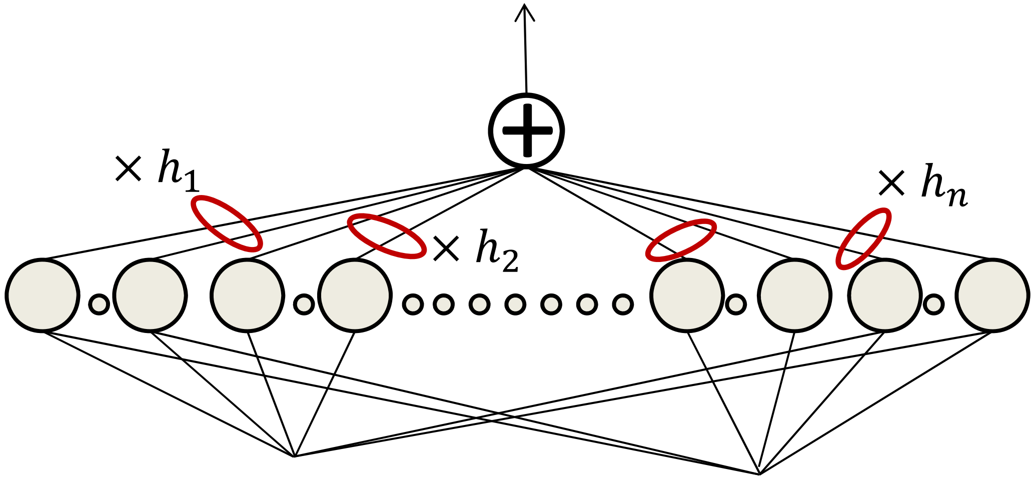
MLP as a continuous-valued regression

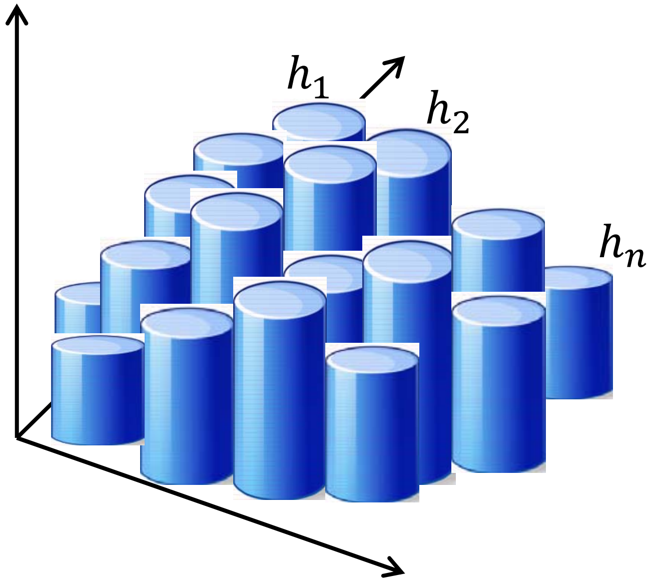




- MLPs can actually compose arbitrary functions to arbitrary precision
- 1D example
  - Left: A simple net with one pair of units can create a single square pulse of any width at any location
  - Right: A network of N such pairs approximates the function with N scaled pulses

MLP as a continuous-valued regression





- MLPs can actually compose arbitrary functions
  - Even with only one layer
  - To arbitrary precision
  - **The MLP is a universal approximator!**

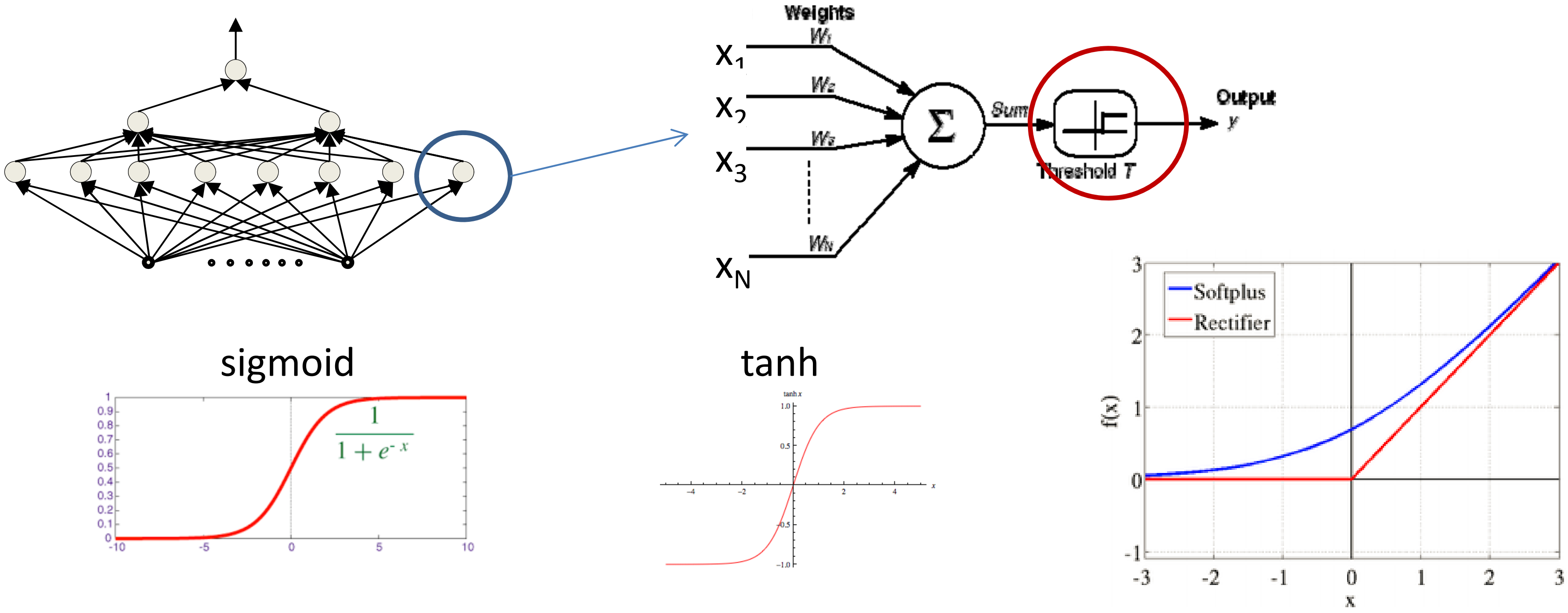
Story so far

- **MLPs are Boolean machines**
  - They represent arbitrary Boolean functions over arbitrary linear boundaries
  - MLPs perform classification

- **Perceptrons are pattern detectors**
  - MLPs are Boolean formulae over patterns detected by perceptrons

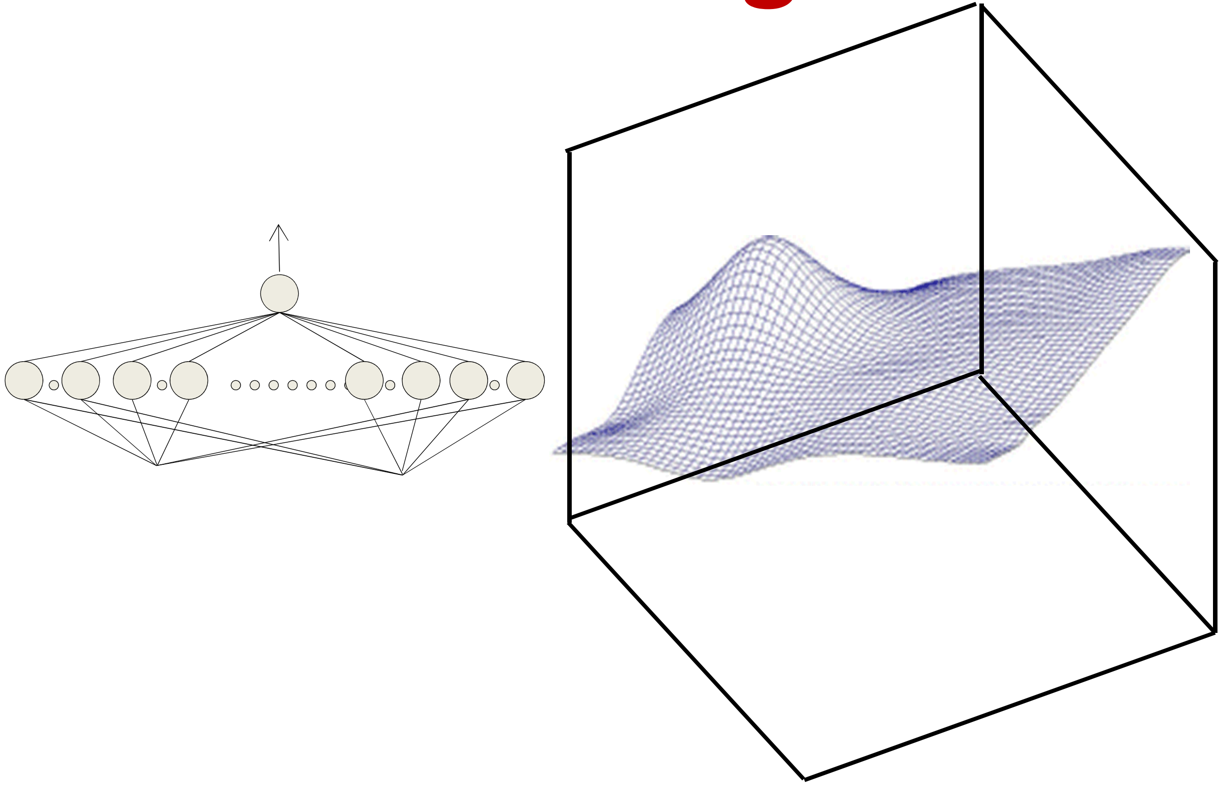
- **MLPs can compute arbitrary real-valued functions of arbitrary real-valued inputs**
  - To arbitrary precision
  - **They are *universal approximators***

A note on activations



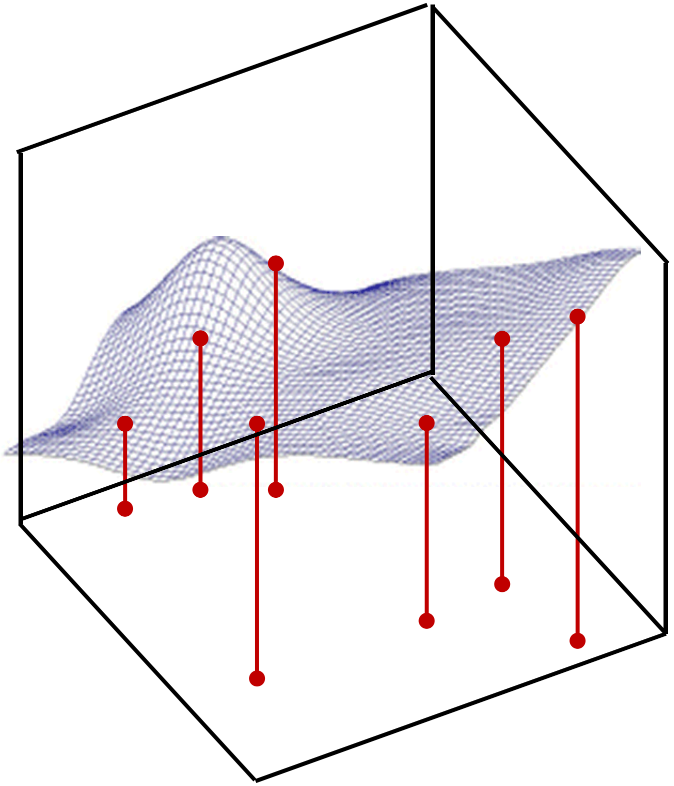
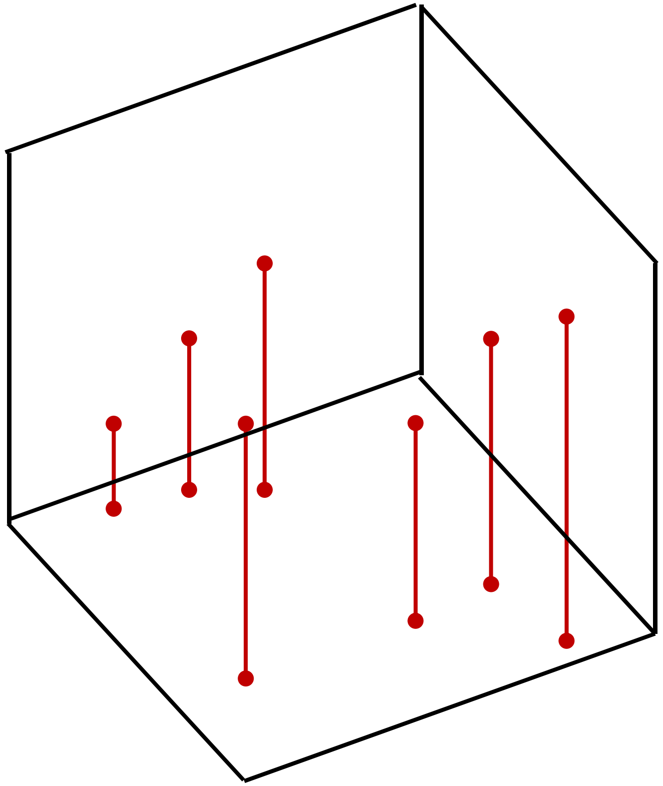
- Explanations have been in terms of a thresholding “step” function applied to the weighted sum of inputs
- In reality, we use a number of other functions
  - Mostly, but not always, “squashing” functions
  - Differentiable, unlike the step function
- Does not substantially change any of our interpretations

Learning the network



- The neural network can approximate *any* function
- But only if the function is known *a priori*

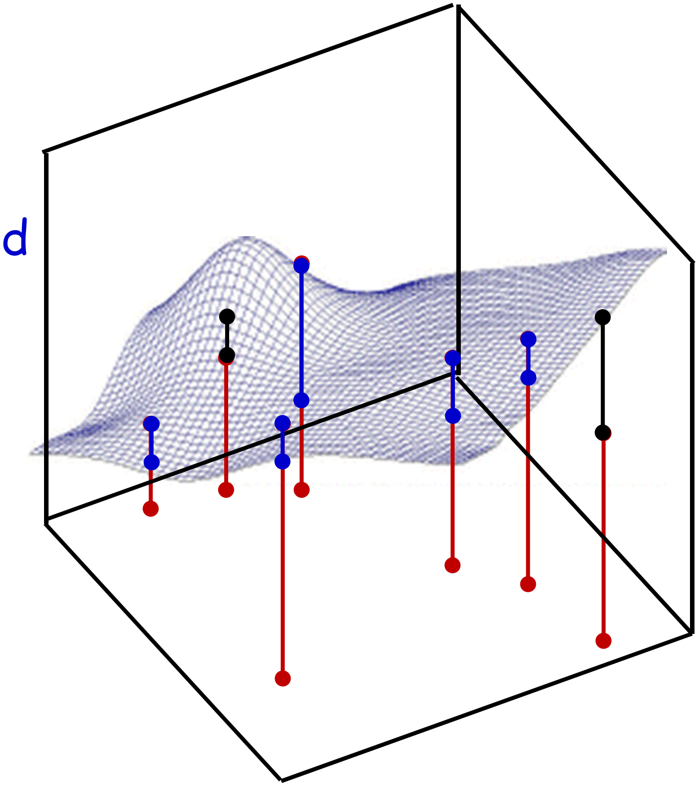
Learning the network

- In reality, we will only get a few *snapshots* of the function to learn it from
- We must learn the entire function from these “training” snapshots

General approach to training

Blue lines: error when function is *below* desired output

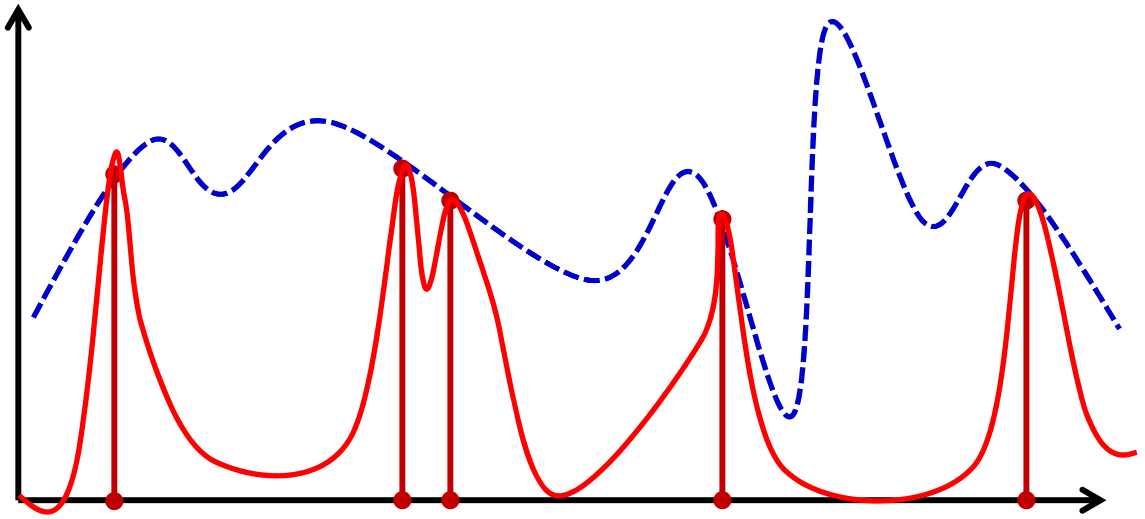


Black lines: error when function is *above* desired output

$$E = \sum_i (y_i - f(\mathbf{x}_i, \mathbf{W}))^2$$

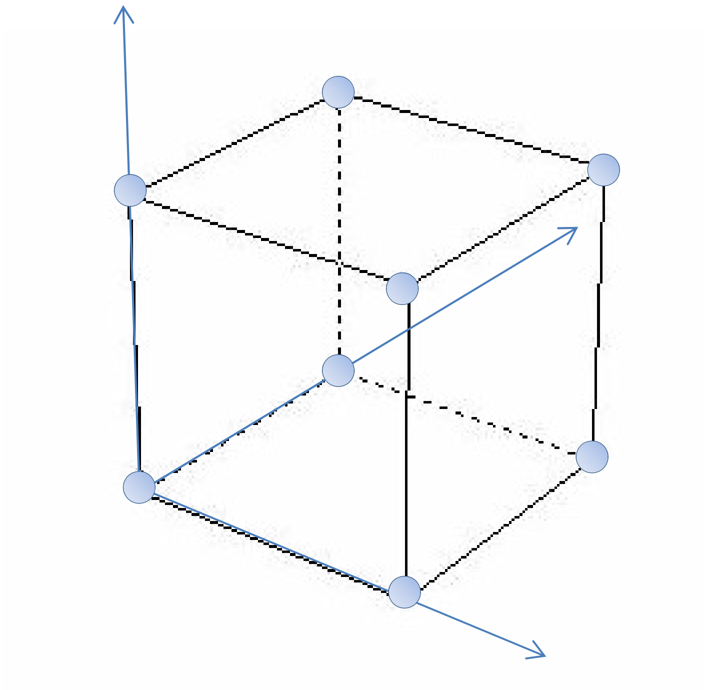
- Define an *error* between the ***actual*** network output for any parameter value and the *desired* output
  - Error typically defined as the *sum* of the squared error over individual training instances

General approach to training



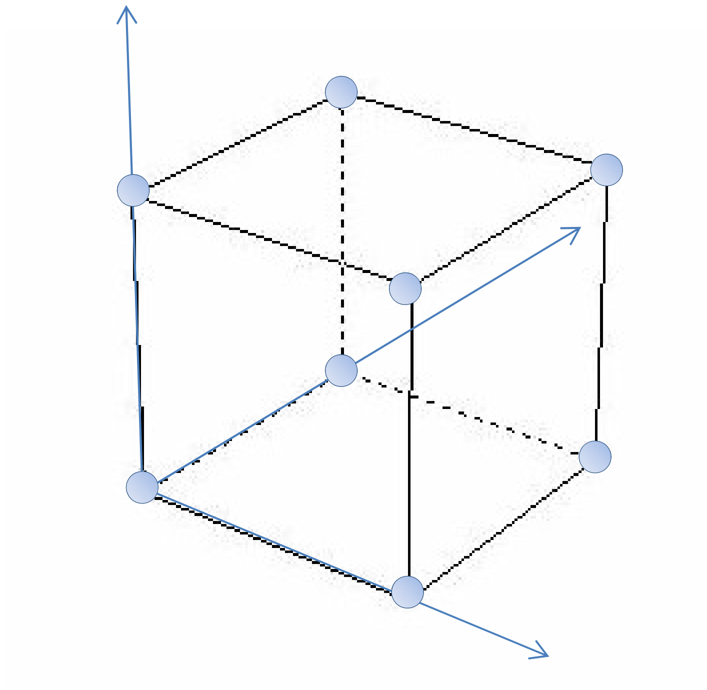
- Problem: Network may just learn the values at the inputs
  - Learn the red curve instead of the dotted blue one
    - Given only the red vertical bars as inputs
  - Need “smoothness” constraints

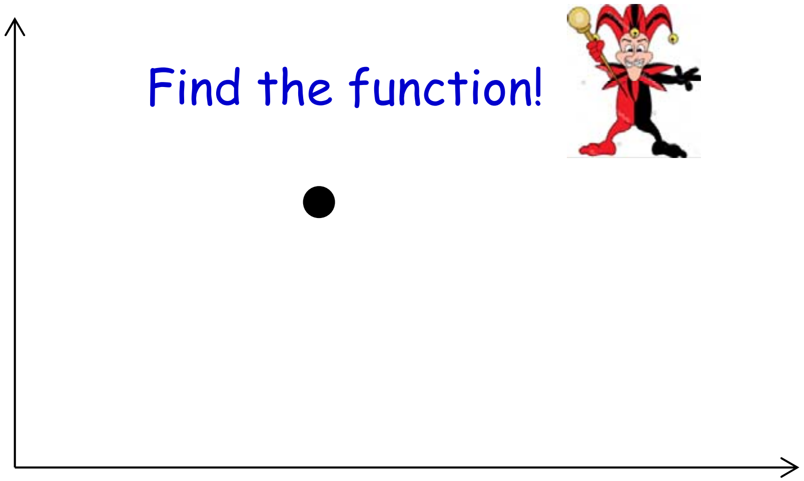
Data under-specification in learning



- Consider a binary 100-dimensional input
- There are $2^{100}=10^{30}$ possible inputs
- Complete specification of the function will require specification of 10^{30} output values
- A training set with only 10^{15} training instances will be off by a factor of 10^{15}

Data under-specification in learning



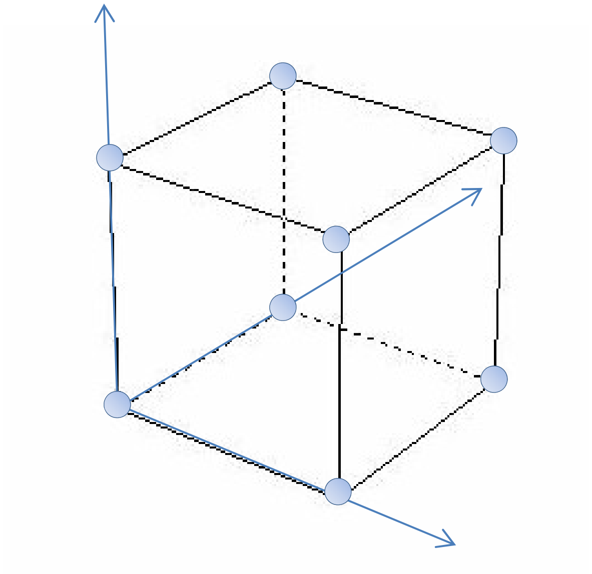


- Consider a binary 100-dimensional input
- There are $2^{100}=10^{30}$ possible inputs
- Complete specification of the function will require specification of 10^{30} output values
- A training set with only 10^{15} training instances will be off by a factor of 10^{15}

Data under-specification in learning

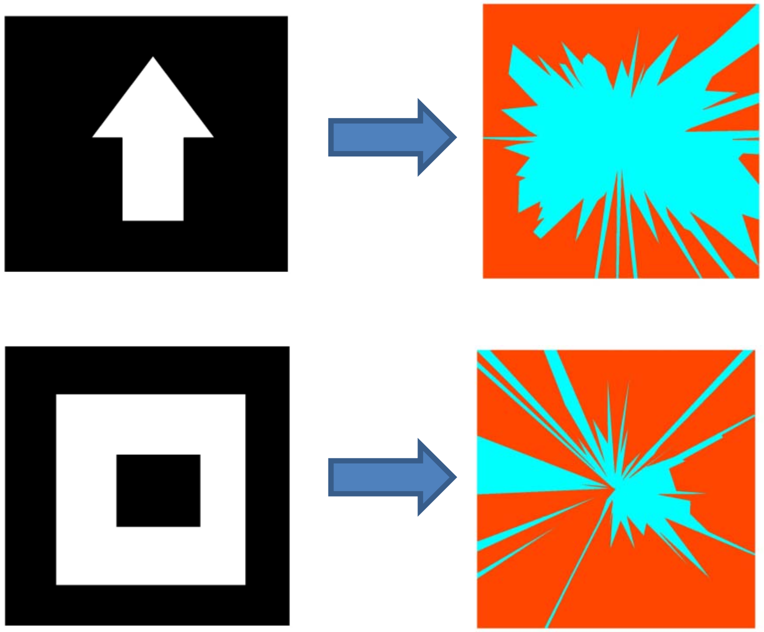
- MLPs naturally impose constraints

- MLPs are universal approximators
  - Arbitrarily increasing size can give you arbitrarily wiggly functions
  - The function will remain ill-defined on the majority of the space



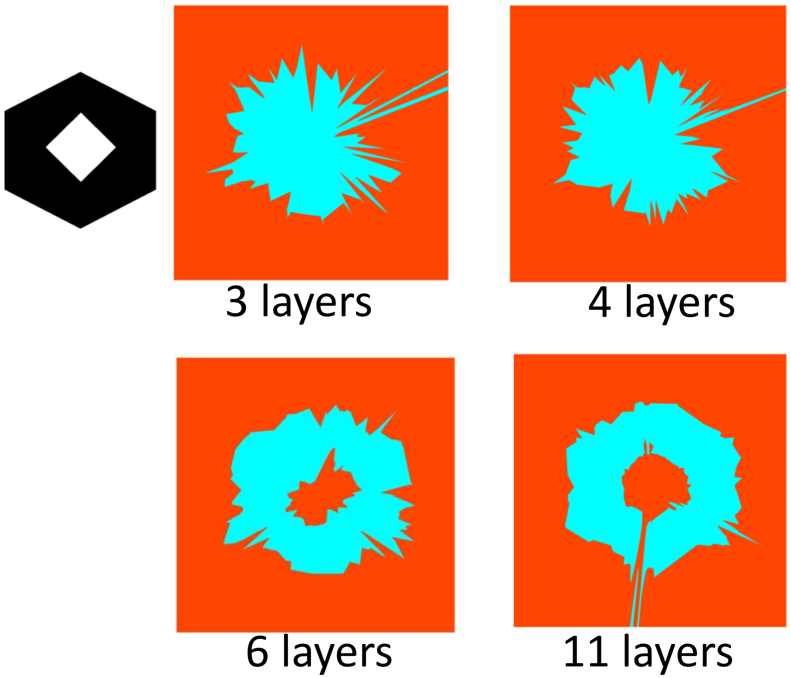
- *For a given number of parameters deeper networks impose more smoothness than shallow ones*
  - Each layer works on the already smooth surface output by the previous layer

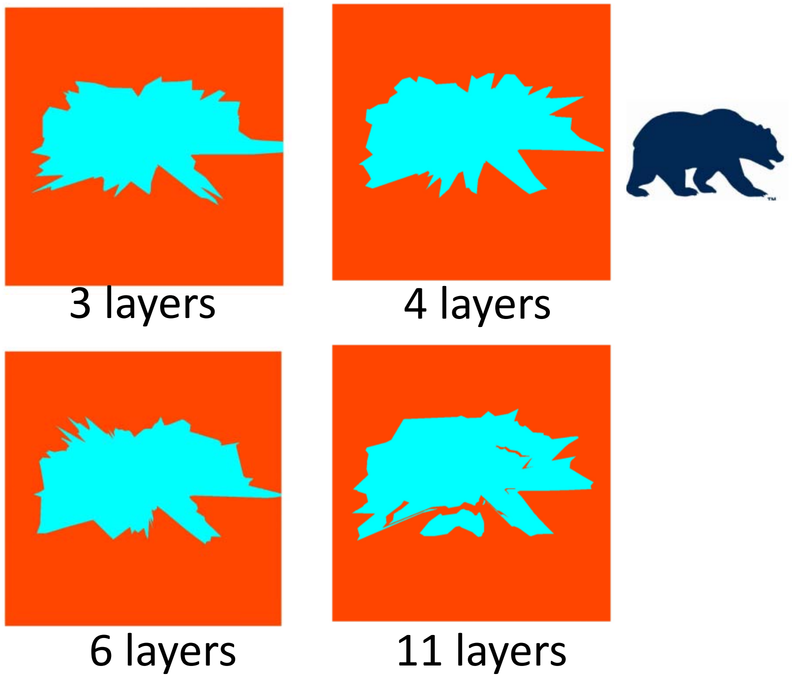
Even when we get it all right



- Typical results (varies with initialization)
- 1000 training points Many orders of magnitude more than you usually get
- All the training tricks known to mankind

But depth and training data help

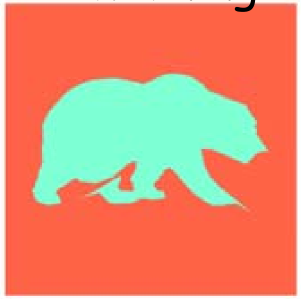




- Deeper networks seem to learn better, for the same number of total neurons
  - *Implicit smoothness constraints*
    - *As opposed to explicit constraints from more conventional classification models*

- **Similar functions not learnable using more usual pattern-recognition models!!**

10000 training instances



Story so far

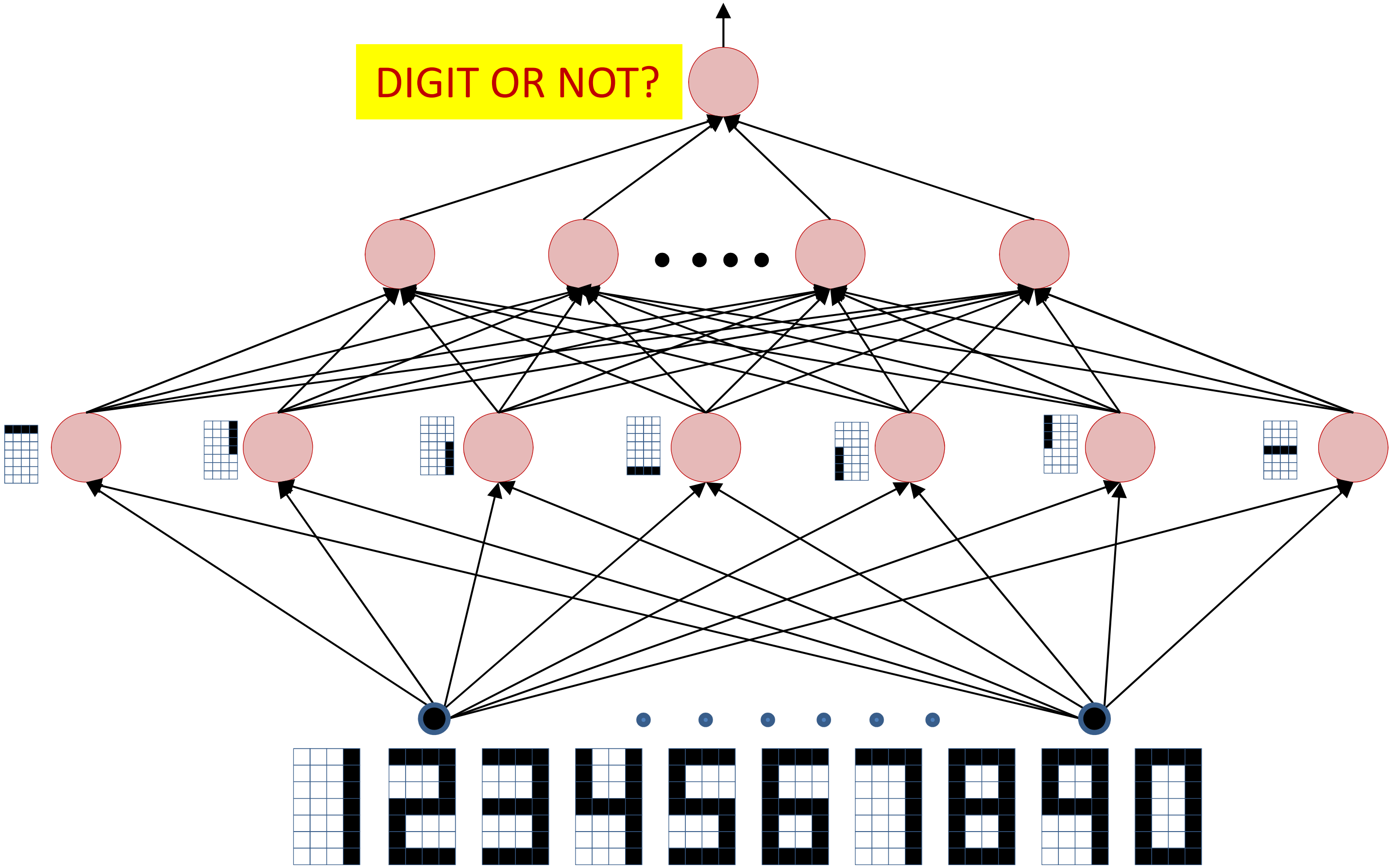
- **MLPs are Boolean machines**
  - They represent arbitrary Boolean functions over arbitrary linear boundaries

- **Perceptrons are pattern detectors**
  - MLPs are Boolean formulae over these patterns

- **MLPs are universal approximators**
  - Can model any function to arbitrary precision

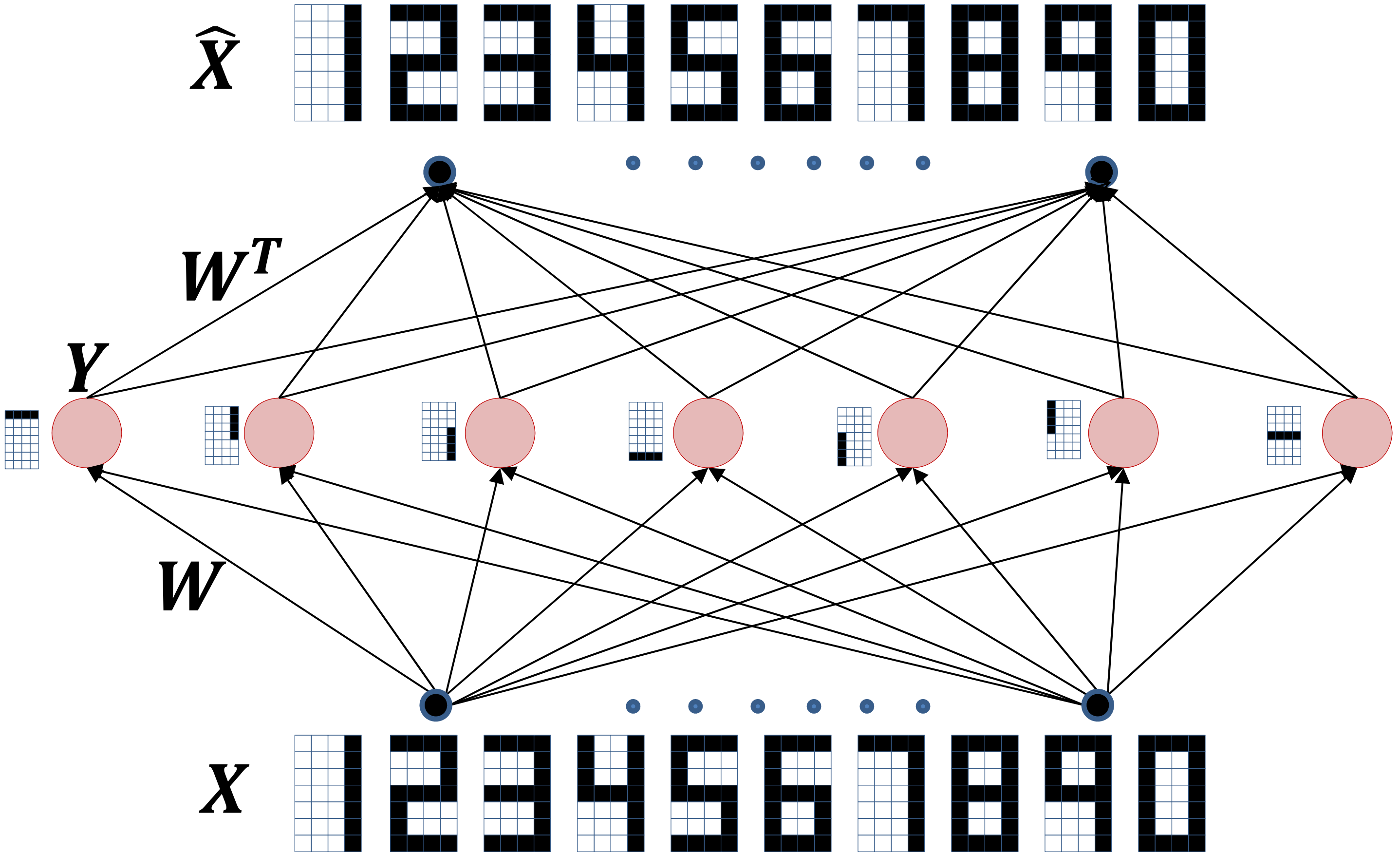
- **MLPs are very hard to train**
  - Training data are generally many orders of magnitude too few
  - Even with optimal architectures, we could get rubbish
  - Depth helps greatly!
  - Can learn functions that regular classifiers cannot

MLP features



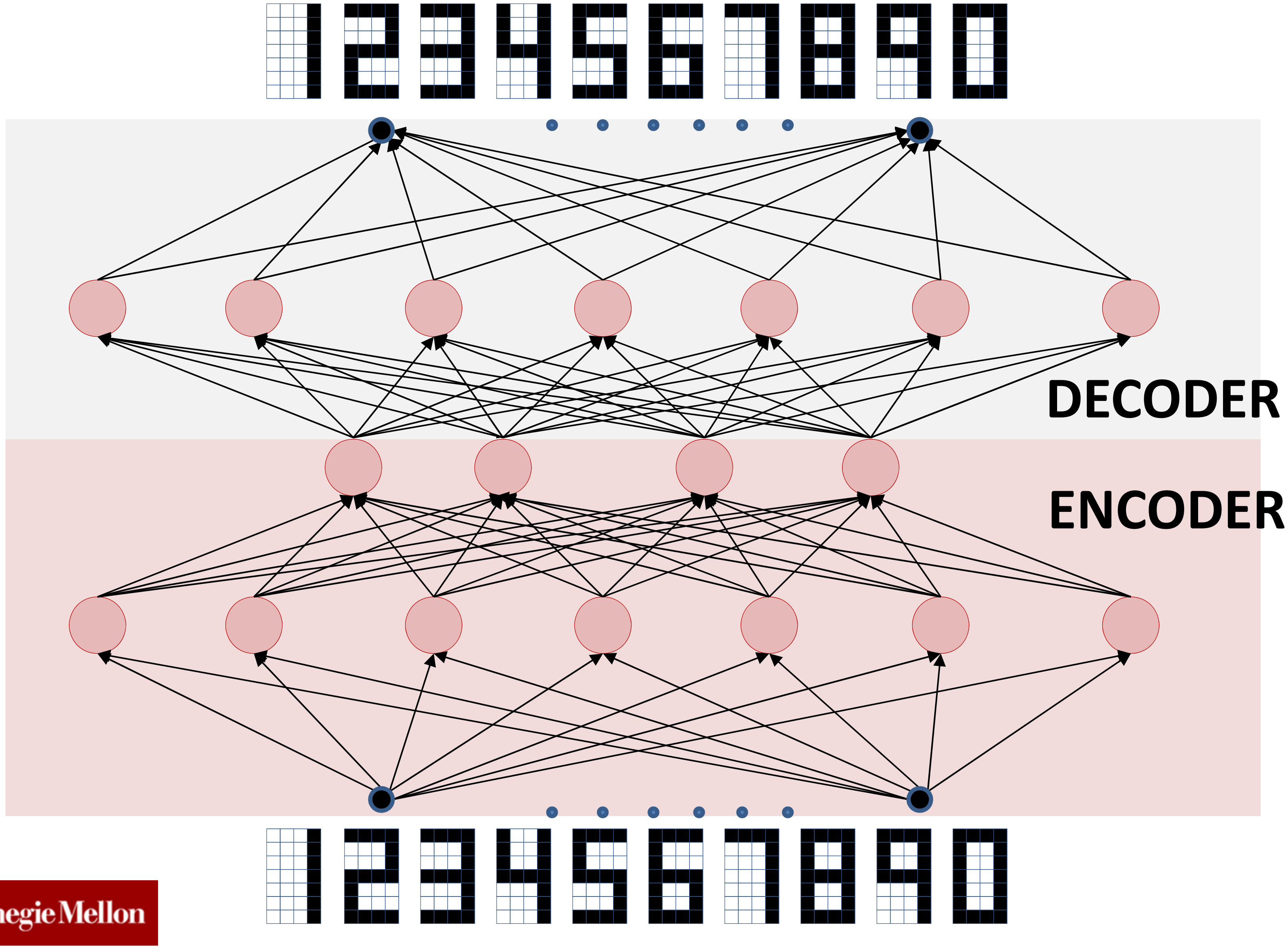
- The lowest layers of a network detect significant features in the signal
- The signal could be reconstructed using these features
  - Will retain all the significant components of the signal

Making it explicit: an autoencoder

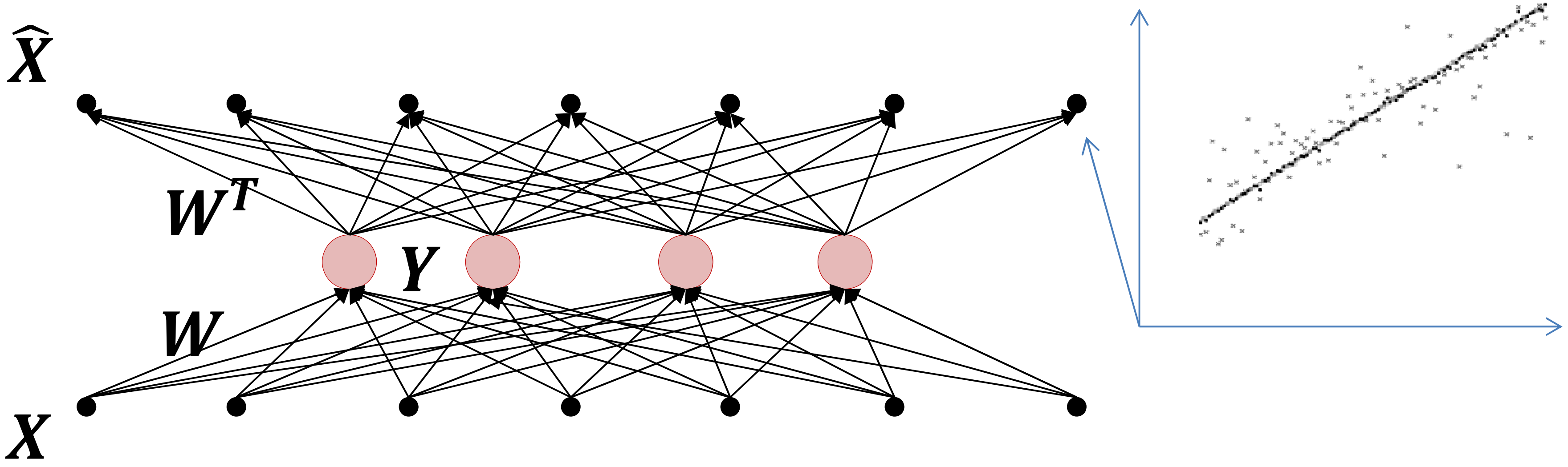


- A neural network can be trained to predict the input itself
- This is an *autoencoder*
- An *encoder* learns to detect all the most significant patterns in the signals
- A *decoder* recomposes the signal from the patterns

Deep Autoencoder



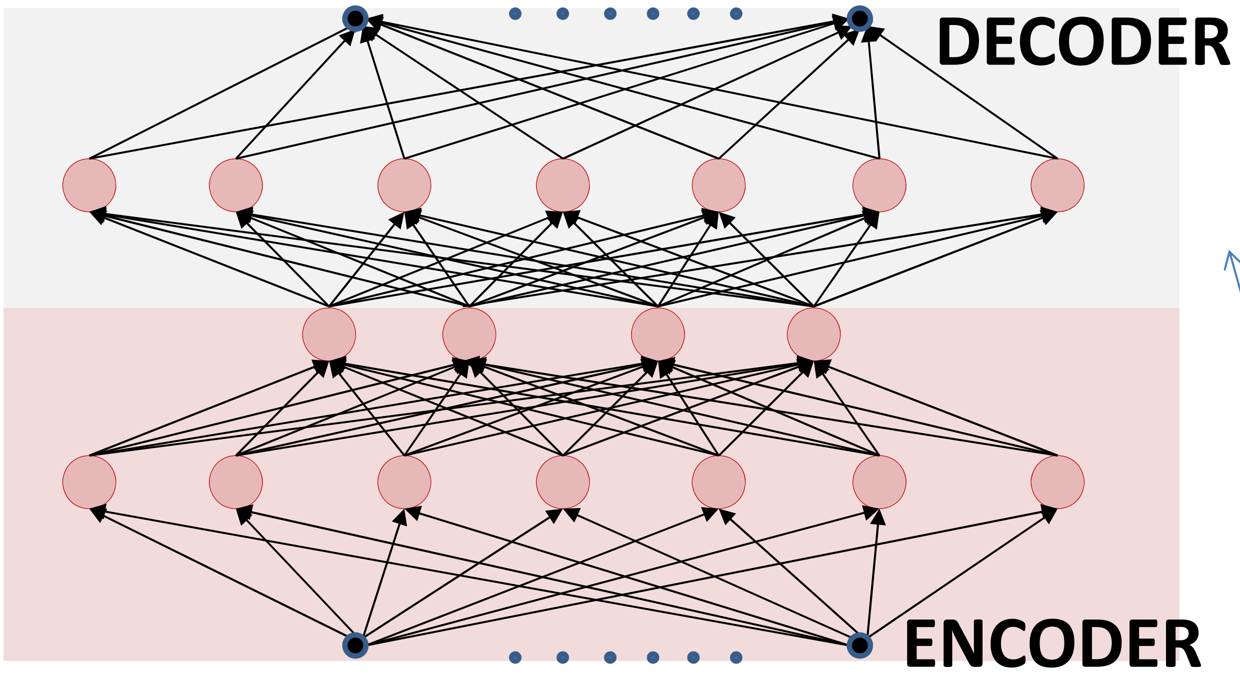
What does the AE learn

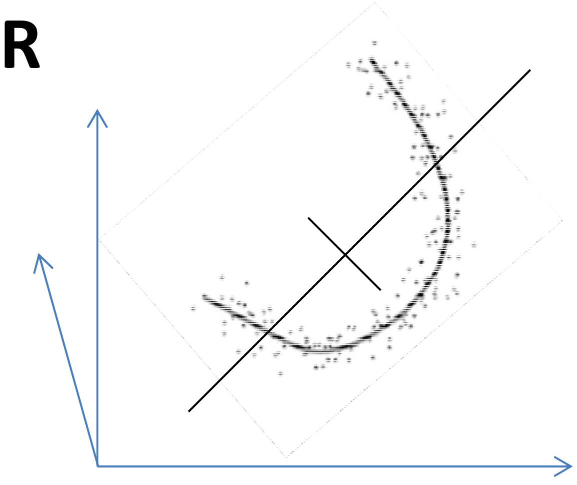


$\boxed{\mathbf{Y} = \mathbf{WX}}$ $\boxed{\hat{\mathbf{X}} = \mathbf{W}^T\mathbf{Y}}$ $E = \|\mathbf{X} - \mathbf{W}^T\mathbf{WX}\|^2$ Find W to minimize Avg[E]

- In the absence of an intermediate non-linearity
- This is just PCA

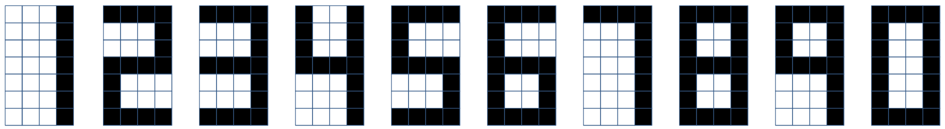
The AE

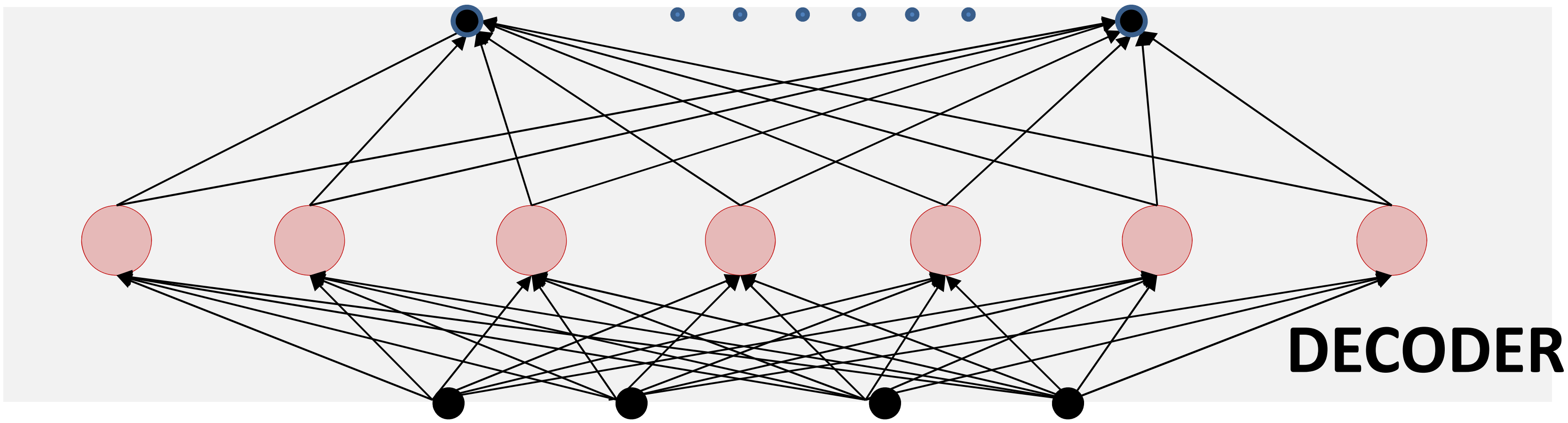




- With non-linearity
  - “Non linear” PCA
  - Deeper networks can capture more complicated manifolds

The Decoder:

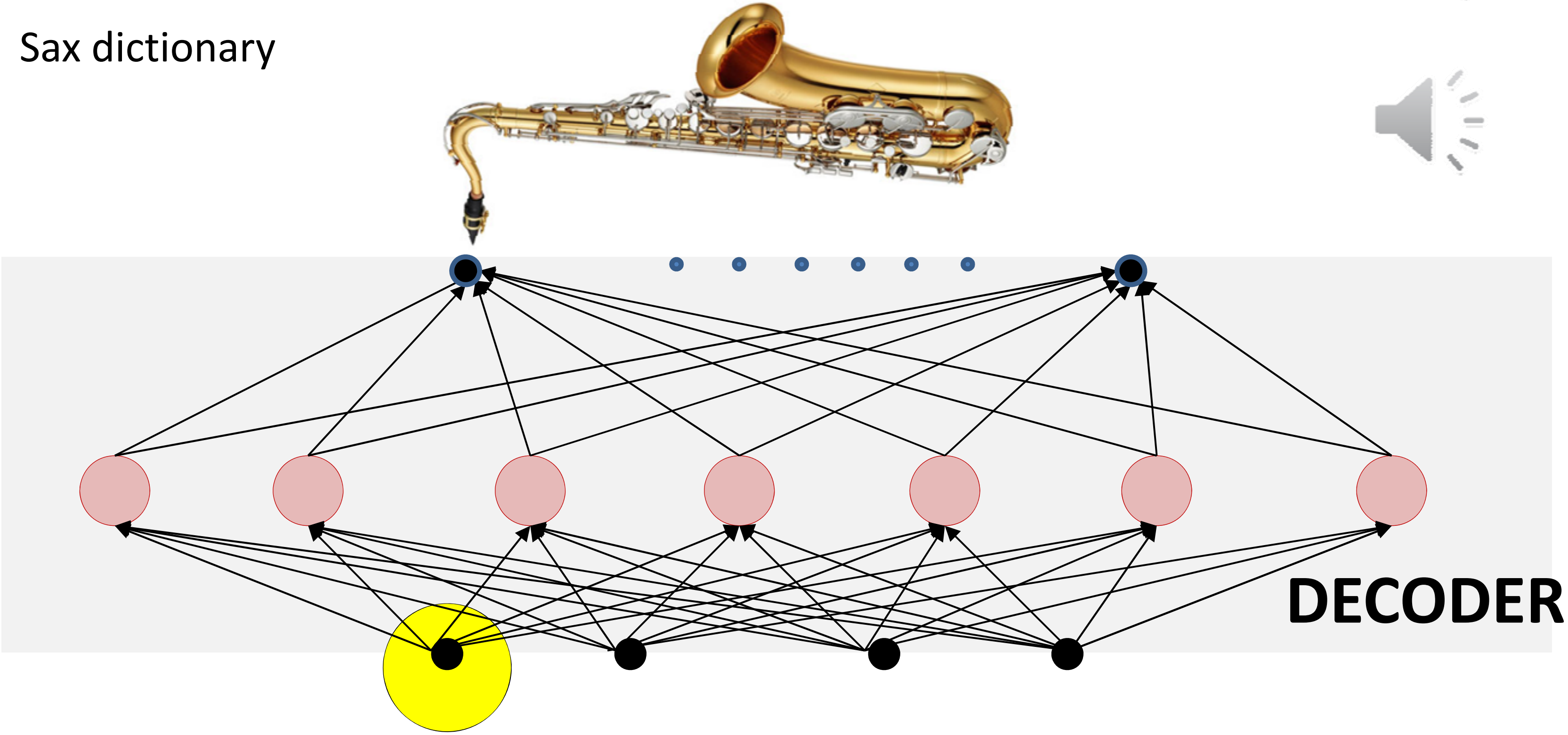




- The decoder represents a source-specific generative *dictionary*
- Exciting it will produce typical signals from the source!

The Decoder:

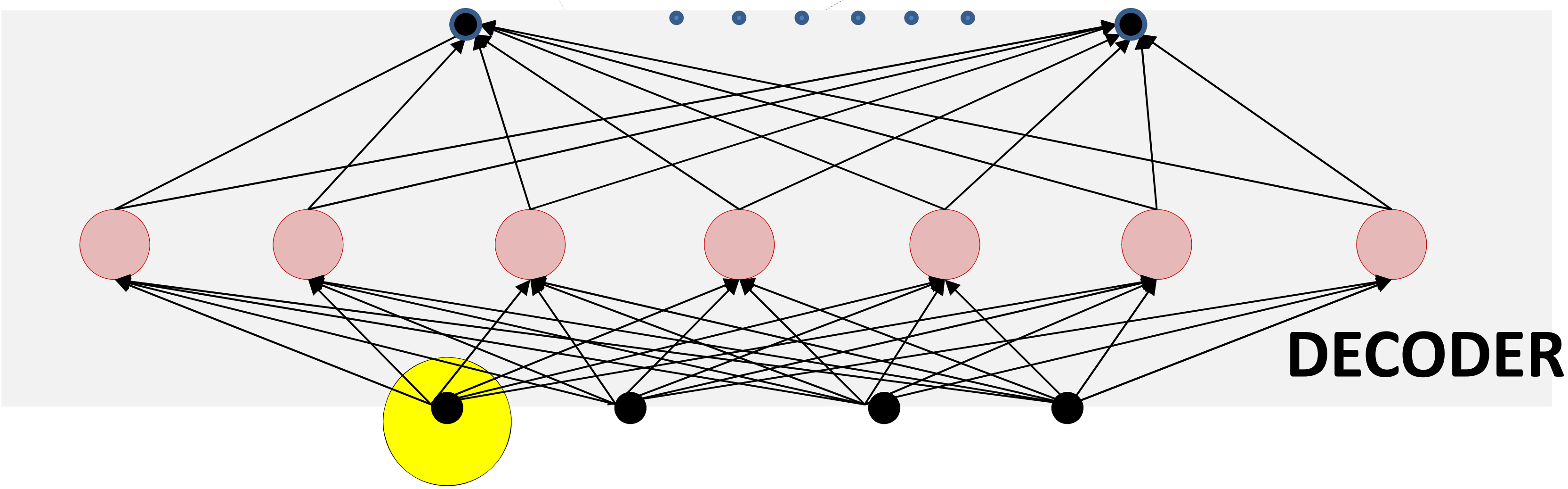
Sax dictionary



- The decoder represents a source-specific generative *dictionary*
- Exciting it will produce typical signals from the source!

The Decoder:

Clarinet dictionary



- The decoder represents a source-specific generative *dictionary*
- Exciting it will produce typical signals from the source!

Story so far

- **MLPs are universal classifiers**
  – They can model any decision boundary

- **Neural networks are universal approximators**
  – They can model any regression

- **The decoder of MLP autoencoders represent a non-linear constructive dictionary!**

NNets for speech enhancement

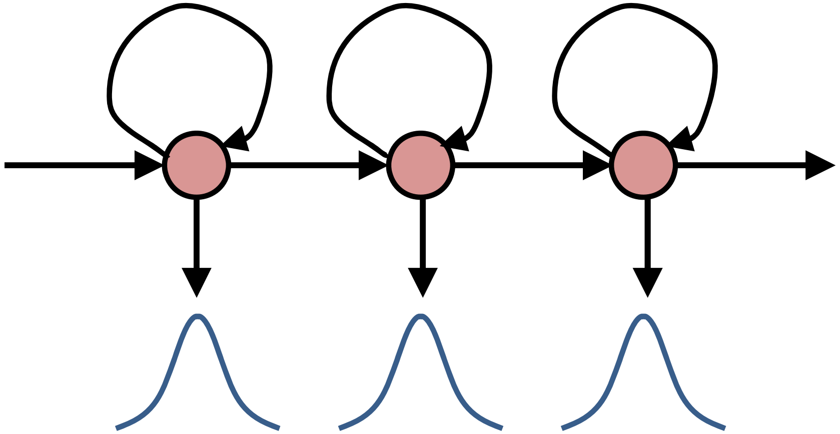
- NNets as a blackbox
- NNets for classification
- NNets for regression
- NNets as dictionaries

- Largely in the context of automatic speech recognition!

NN as a black box

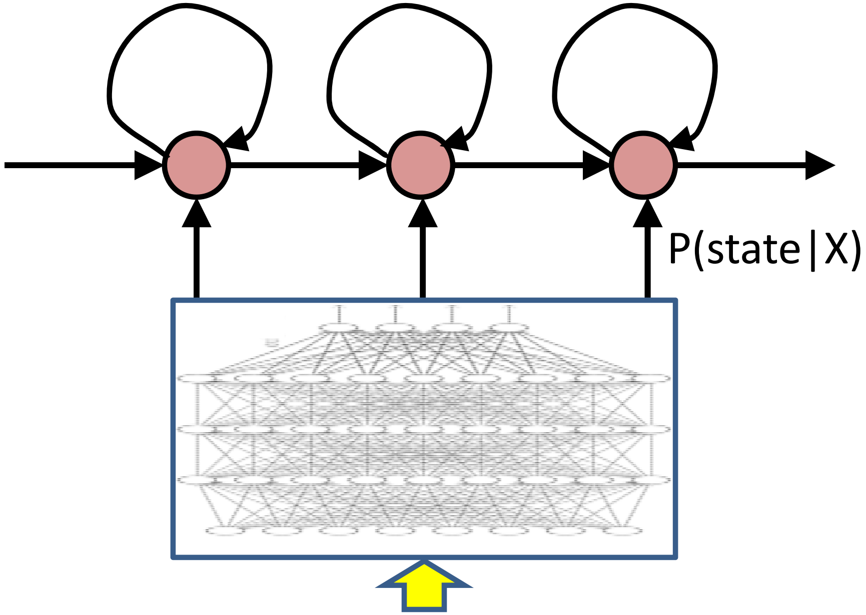
- In speech recognition tasks, simply providing the noise as additional input to the recognizer seems to provide large gains!

Old-fashioned Automatic Speech Recognition



- Traditional ASR system (antebellum, circa 2010)
  - Phonemes modelled by HMMs
  - Phoneme state output distributions modelled by Gaussian mixtures

Deep Neural Networks for Automatic Speech Recognition

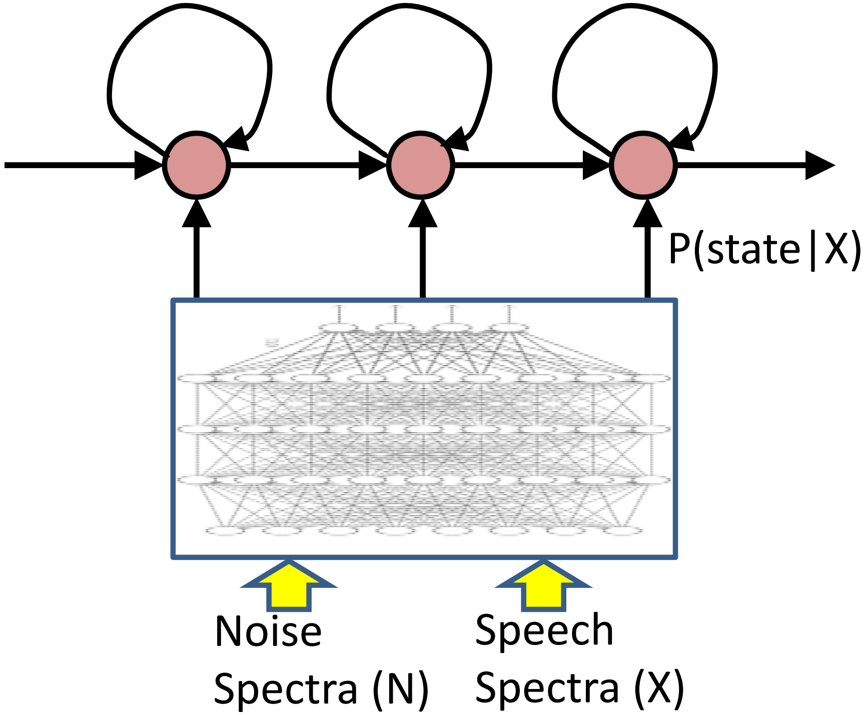


P(state|X)

Spectral Vectors of Speech (X)

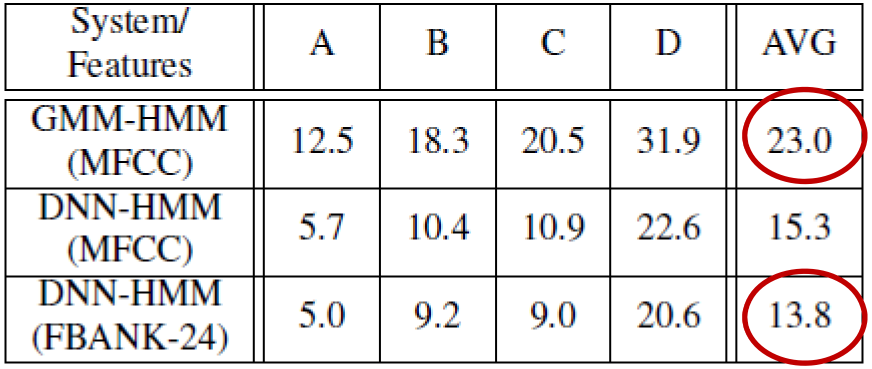
- Postbellum ASR
  - Gaussian mixtures replaced by a deep neural network

NN-BB: *Noise-Aware* speech recognition

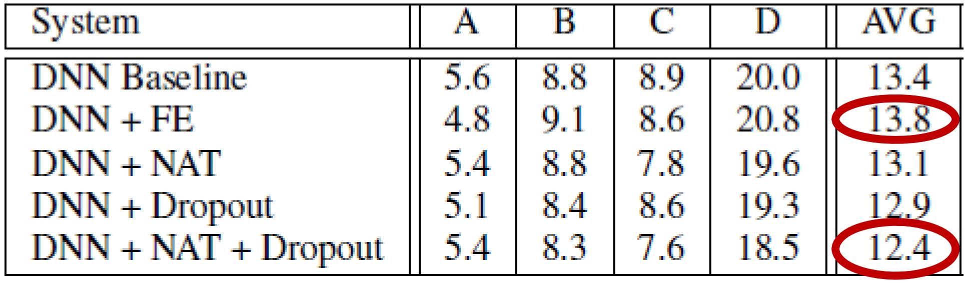


- Simply add an estimate of the noise as an additional input
  - The system is "noise aware"
- The noise estimate too may have been derived by another network..

NN-BB: *Noise-Aware* speech recognition



System/ Features	A	B	C	D	AVG
GMM-HMM (MFCC)	12.5	18.3	20.5	31.9	23.0
DNN-HMM (MFCC)	5.7	10.4	10.9	22.6	15.3
DNN-HMM (FBANK-24)	5.0	9.2	9.0	20.6	13.8

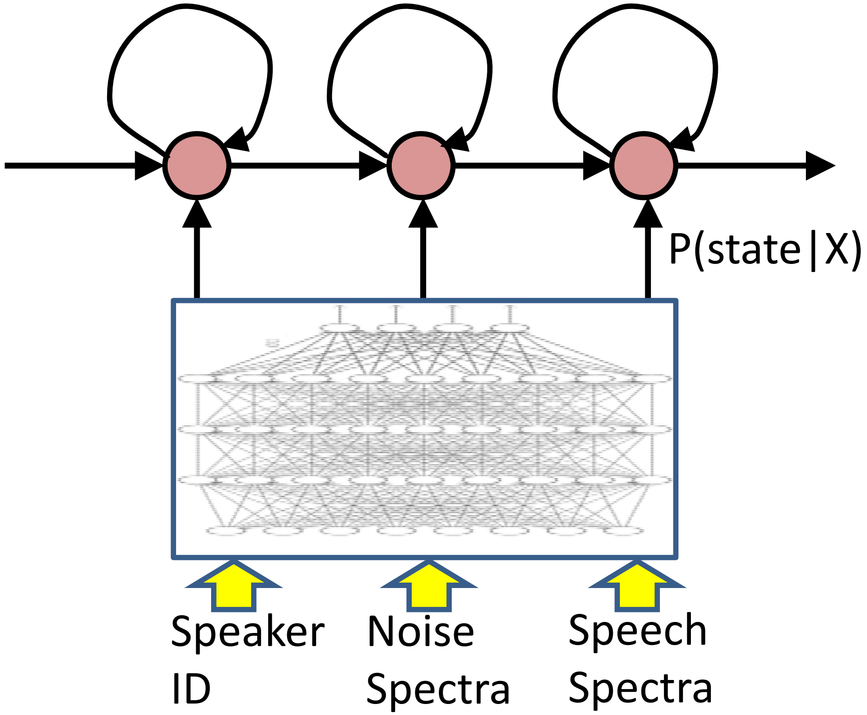


System	A	B	C	D	AVG
DNN Baseline	5.6	8.8	8.9	20.0	13.4
DNN + FE	4.8	9.1	8.6	20.8	13.8
DNN + NAT	5.4	8.8	7.8	19.6	13.1
DNN + Dropout	5.1	8.4	8.6	19.3	12.9
DNN + NAT + Dropout	5.4	8.3	7.6	18.5	12.4

From Seltzer, Yu, Wong, ICASSP 2013

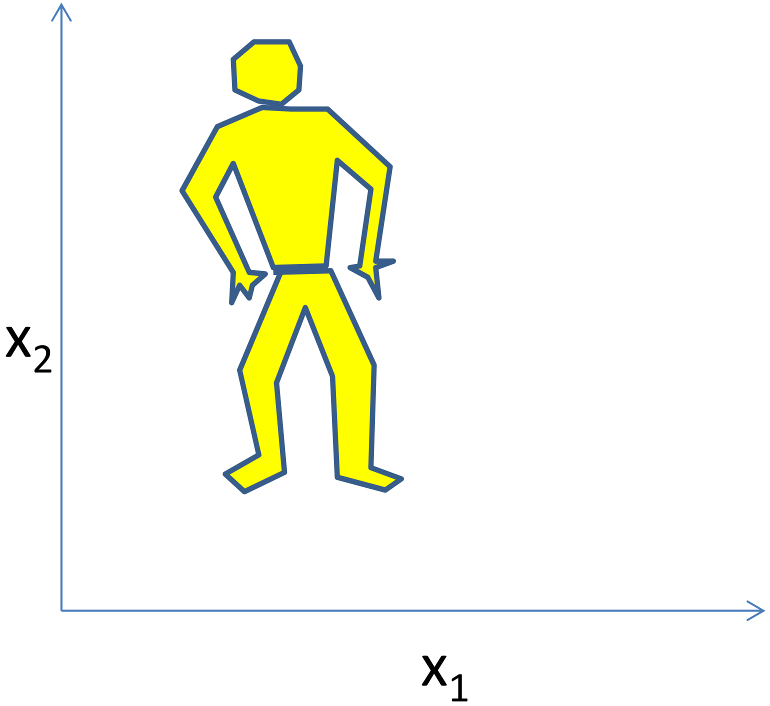
- DNN provides large improvements by itself
  - Results on Aurora 4 task, with four subtasks
- Adding “noise-awareness” improves matters
  - Seltzer, Yu, Wong, 2013, many others later
- Acutal noise spectrum not essential
  - Simply having a guess of noise *type* is beneficial (Kim, Lane, Raj, 2016)

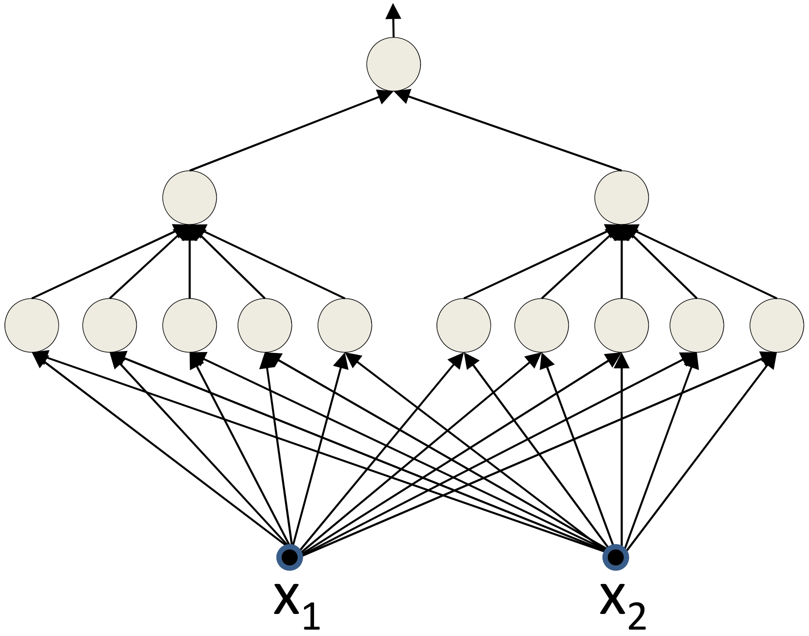
NN-BB: **-aware* speech recognition



- Adding extra input about *any* additional signal characteristic improves matters
  - Speaker
  - Environment
  - Channel..

Neural Networks as Classifiers





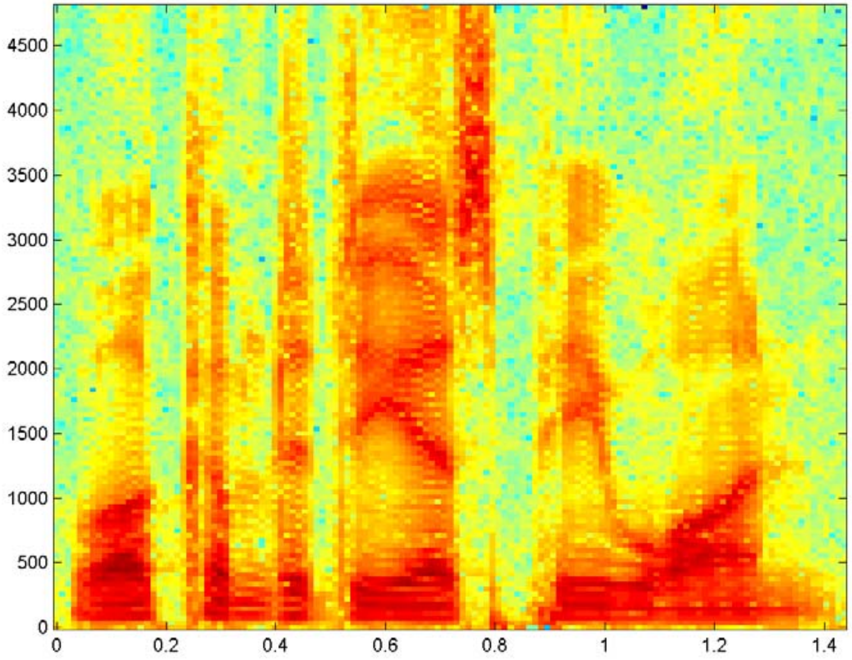
- Neural networks learn Boolean classification functions
- For a fixed network size, deeper networks learn better functions
- Can be superior to conventional classification functions

Recasting Signal Enhancement as Classification

- Noise attenuation can be viewed as the detection of spectrographic *masks*
  - A classification problem

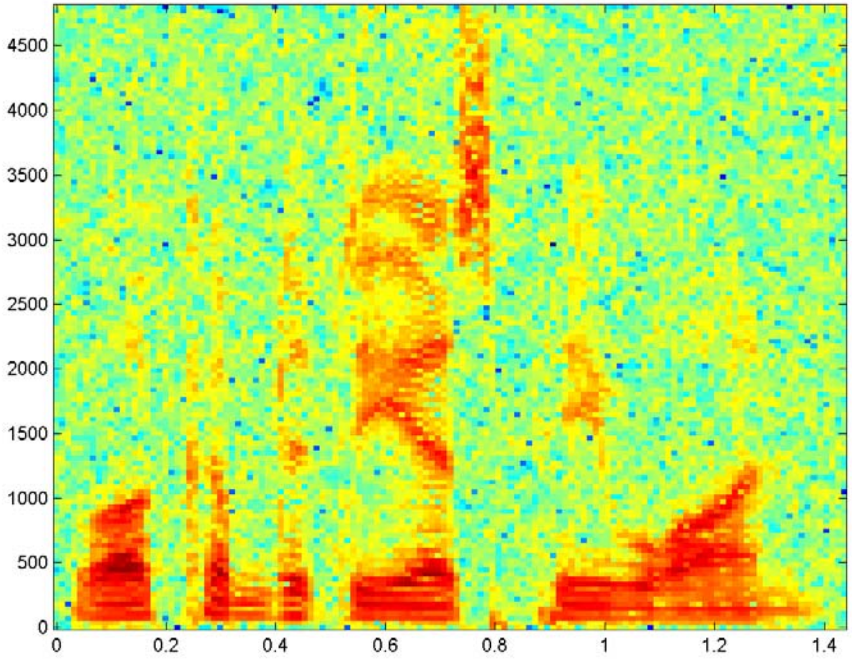
- The classification can be performed by a neural network

Spectrogram of a Clean Speech Signal



- A clean speech signal
  - Richard Stern saying “Welcome to DSP1”

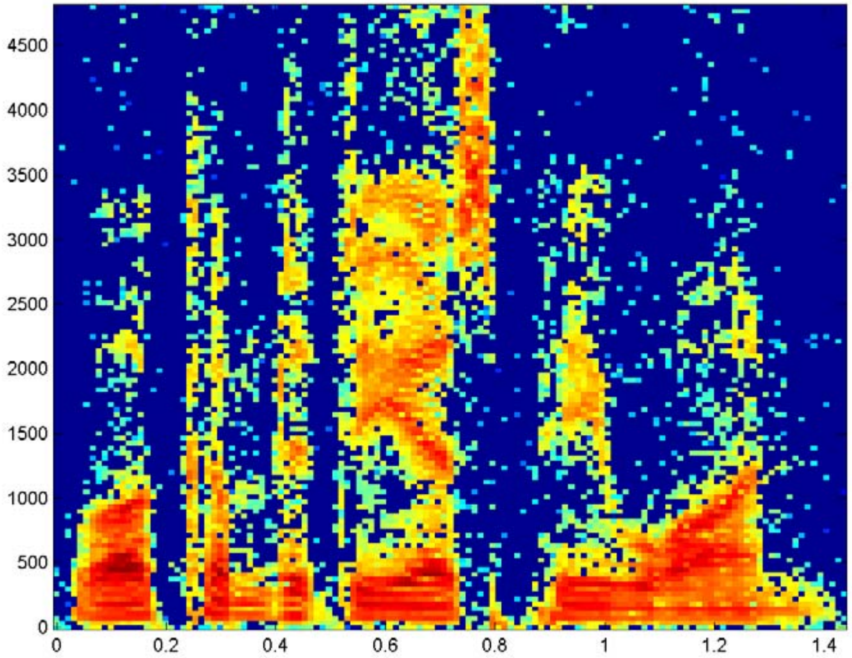
Spectrogram of Speech Corrupted to 5 dB by White Noise



- High-energy regions of spectrogram remain
- Low-energy regions now dominated by noise!
  - Most of the effects of noise expressed in these regions

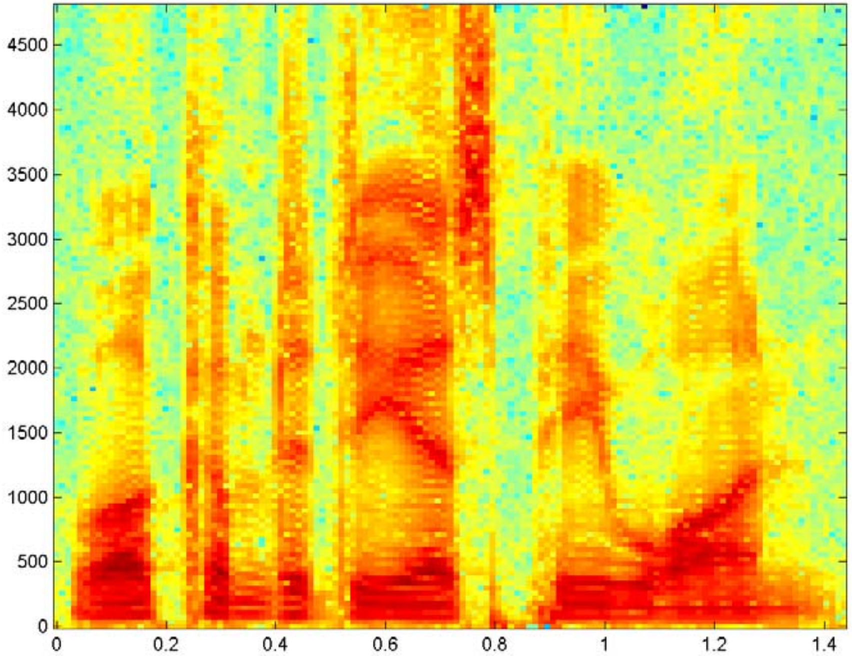
Carnegie Mellon

Erasing Noisy Regions of the Picture



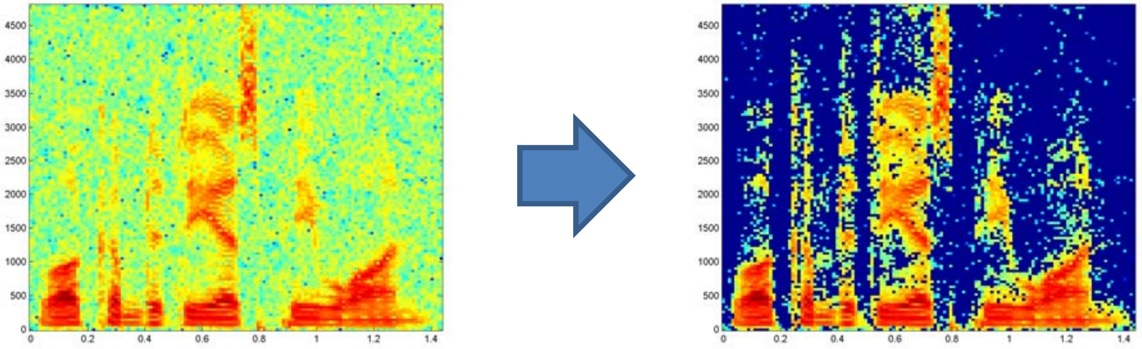
- Solution: “Mask” (erase) all noise corrupted regions in the spectrogram (floor them to 0)
- Reconstruct the signal from the partial spectrogram

Erasing Noisy Regions of the Picture



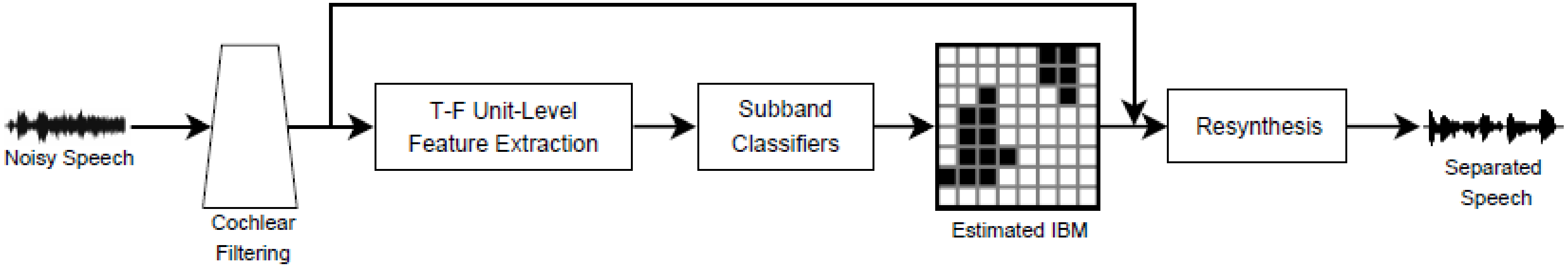
- Solution: Mask all noise corrupted regions from the spectrogram (floor them to 0)
- Reconstruct the signal from the partial spectrogram

Challenge

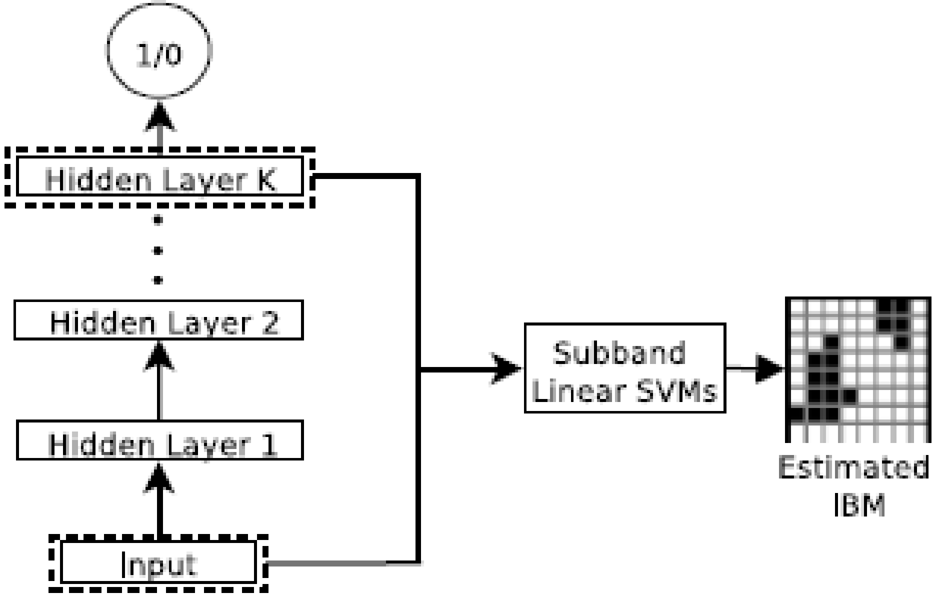


- From inspection of time-frequency components of spectrogram, how to determine which to erase
  - A hard classification problem
    - Many ineffective solutions proposed over the years
  - Ideally suited to learn with a neural network!

Estimating Masks



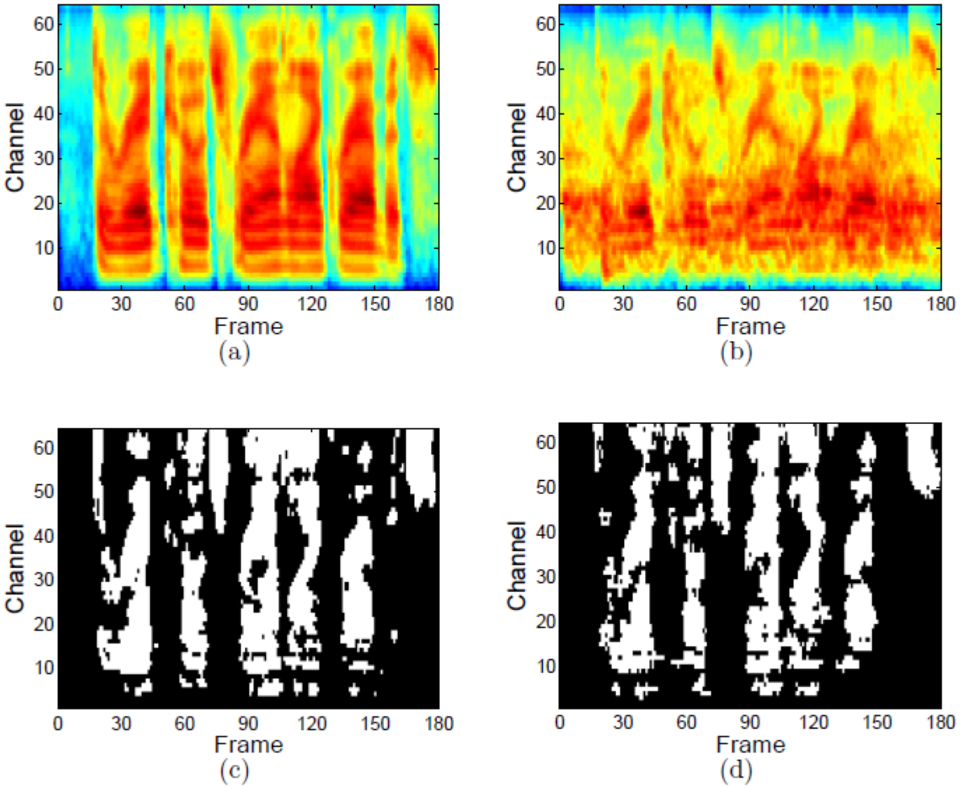
From: “supervised speech separation”,
PhD dissertation Y Wang, Ohio State Univ



- Top: General flow of solution
- Bottom: Classifier
  - The network itself produces a mask

Carnegie Mellon

Estimating Masks



(a)

(b)

(c)

(d)

Clean speech

Speech + babble

Ideal mask

Estimated mask

- Example solution by Yuxuan Wang
  - Network with only 2 hidden layers of 50 sigmoid units each
  - PhD dissertation with Deliang Wang at OSU
  - Results reported in terms of HIT-FA rates (70% achieved)

Sound demos

Speech mixed with unseen, daily noises

Cocktail party noise (5 dB)	🔈 Mixture	🔈 Separated
Destroyer noise (0 dB)	🔈 Mixture	🔈 Separated

Slide from Deliang Wang

Story so far

- Capabilities and Limitations of NNets
- NNets can be classifiers of unlimited versatility
- NNets can be regression functions of unlimited versatility
- NNets can be very good constructive dictionaries

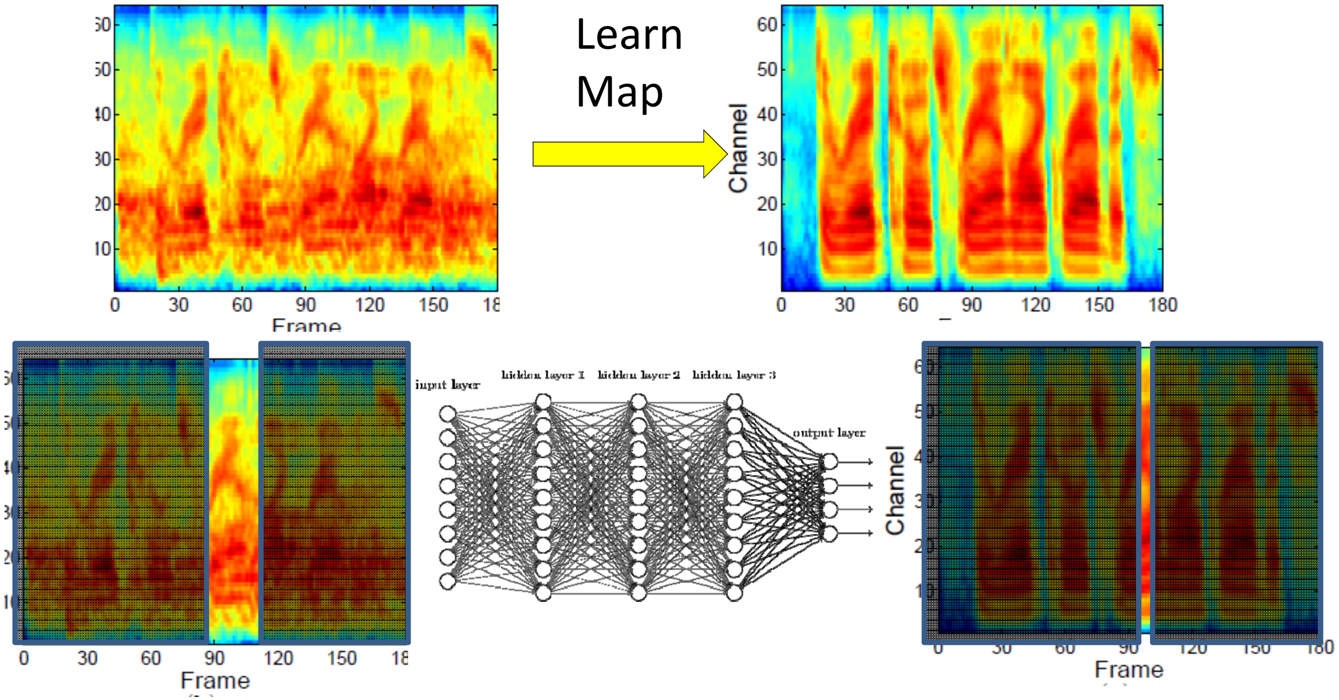
- **NNet classifiers can be used to enhance speech signals**

NNets as regression

- Neural networks can also compute continuous-values outputs
  - May also be viewed as regression models

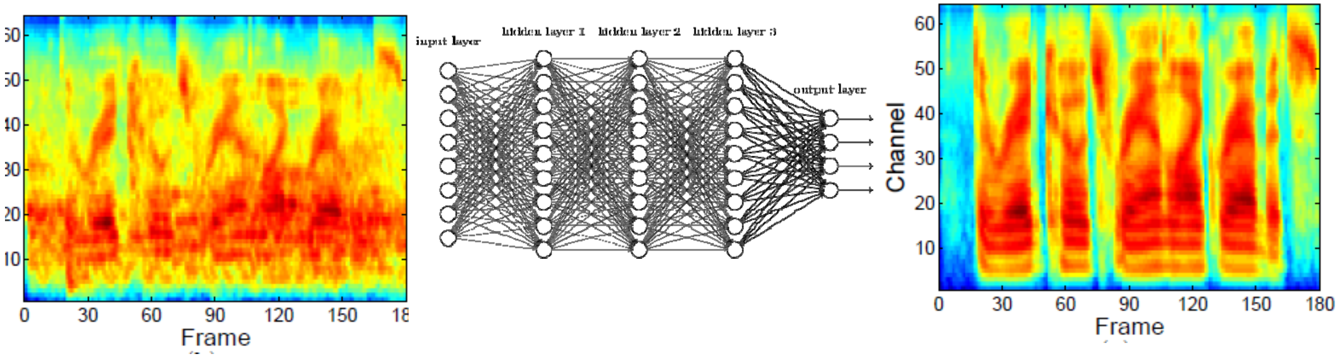
- NNet as regression:
  - Estimate clean speech from noisy speech directly
  - Replace filtering modules in conventional signal processing systems with learned nnet-based versions

NNets for denoising



- Xu, Du, Dai, Lee, IEEE Sig. Proc. Letters, Jan 2014.
- Simple model:
  - Given clean-noisy stereo pairs of signals
    - Represented spectrographically
  - Learn to predict a single clean frame from a window of noisy frames
- Given noisy speech, use the network to predict clean speech

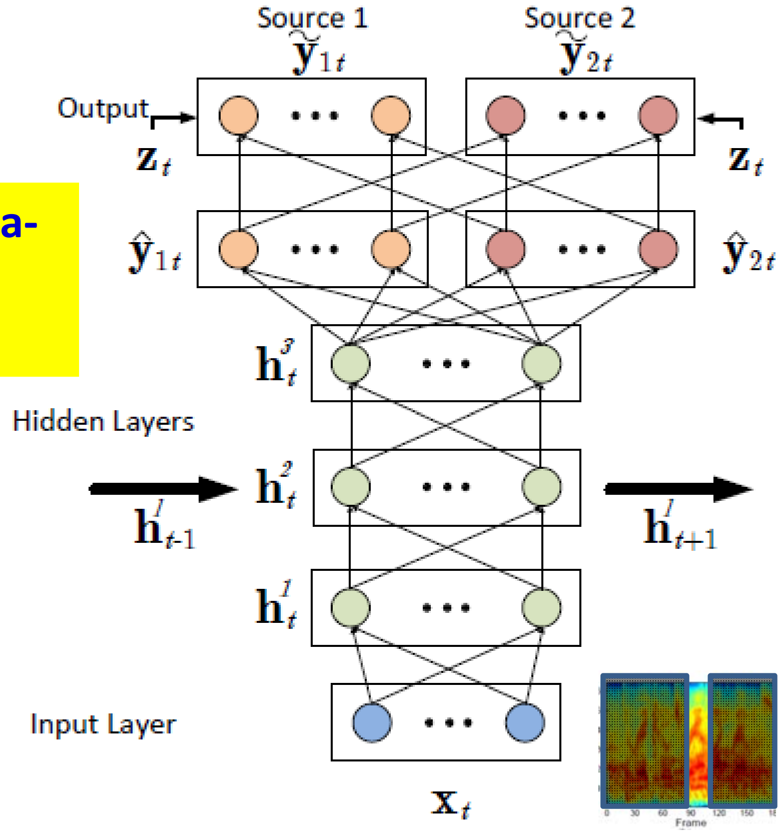
NNets for denoising



- Xu, Du, Dai, Lee, IEEE Sig. Proc. Letters, Jan 2014.
  - 3 hidden layers of 2048 neurons
- Example of signal corrupted to -12dB by babble noise

A more detailed solution for mixtures

Huang, Kim, Hasegawa-Johnson, Smaragdis, TASLP, Dec 2015



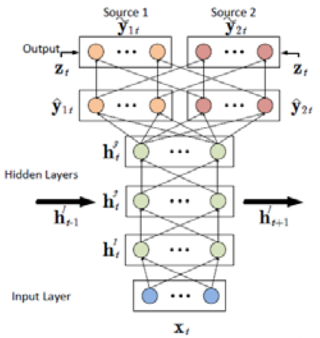
$$m_t = \frac{|\hat{y}_{1_t}|}{|\hat{y}_{1_t}| + |\hat{y}_{2_t}|}$$

Recurrent network

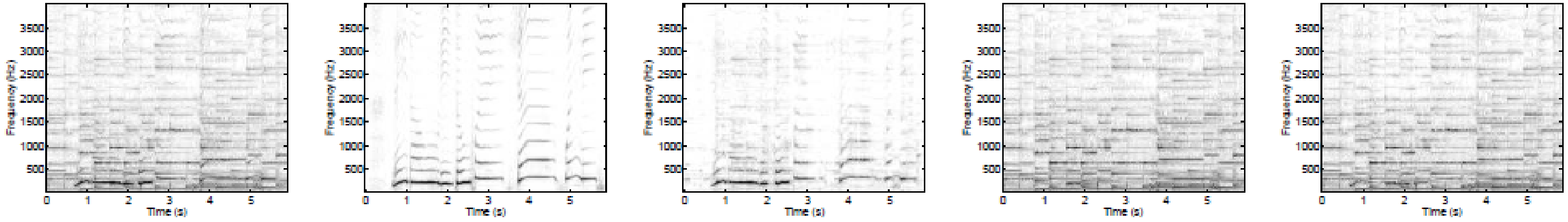
- Works on mixtures of pairs of sounds
- Model:
  - Input : sound mixture (window of spectrographic frames)
  - Output : *Both* sources (single spectrographic frame)

Network size
Training data?

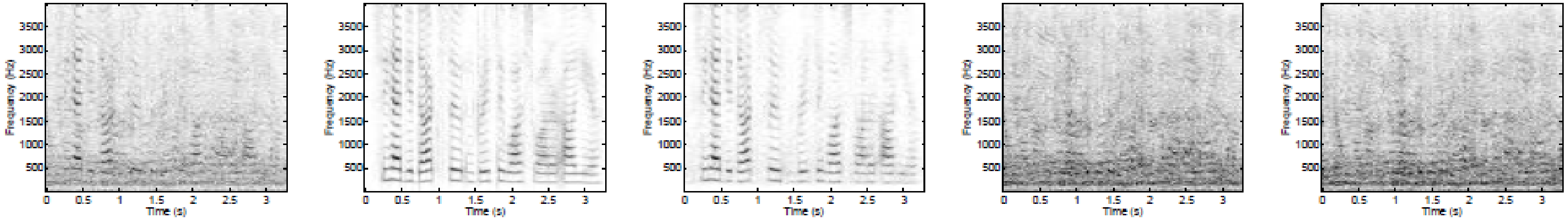
Huang et. al. results



Singing voice in music



(a) Mixture (b) Original singing (c) Recovered singing (d) Original music (e) Recovered music



(a) Mixture (b) Original speech (c) Recovered speech (d) Original noise (e) Recovered noise

Speech in babble noise

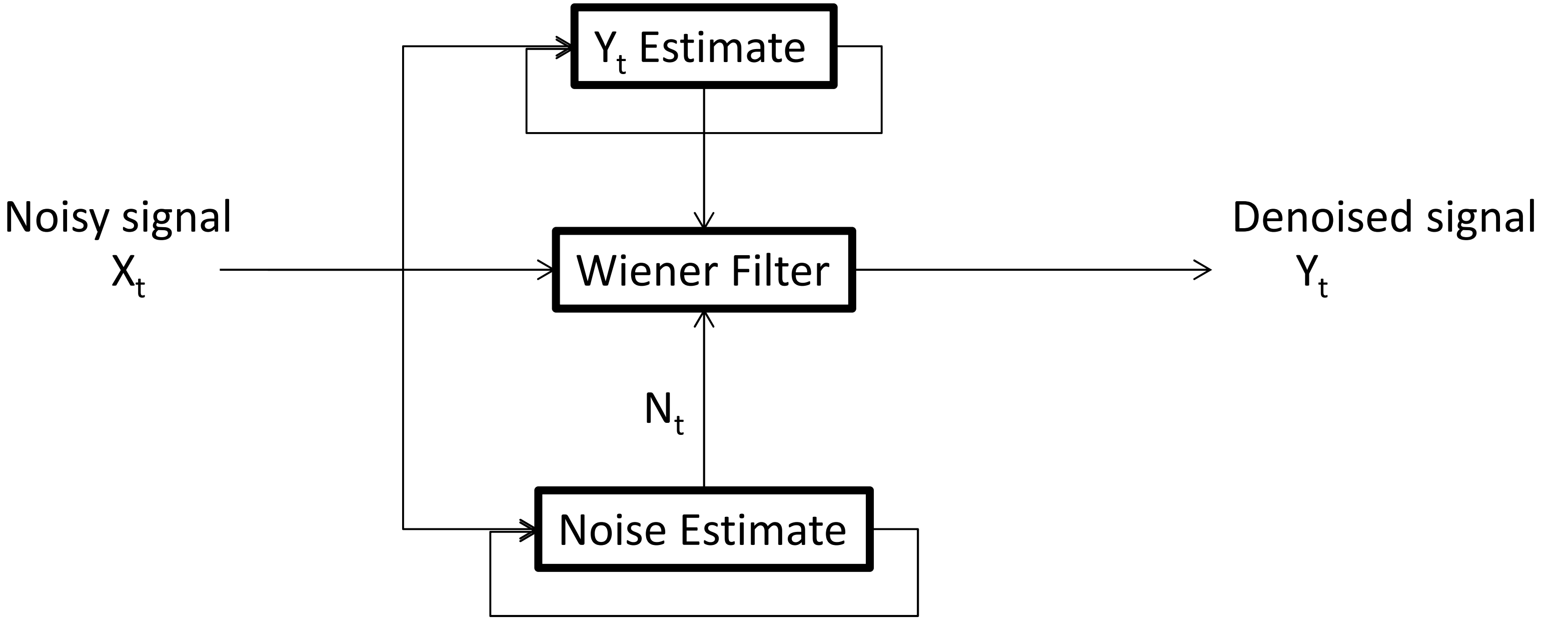
- Huang, Kim, Hasegawa-Johnson, Smaragdis, TASLP, Dec 2015
  - Recurrent nets, 2 (speech) or 3 (singing) hidden layers of 1000 neurons
- ~10dB improvement in speech to interference

NNets in Conventional Signal Processing

- Conventional signal processing techniques have been developed over several decades
  - Theoretical capabilities mathematically demonstrated
  - Practical capabilities empirically demonstrated

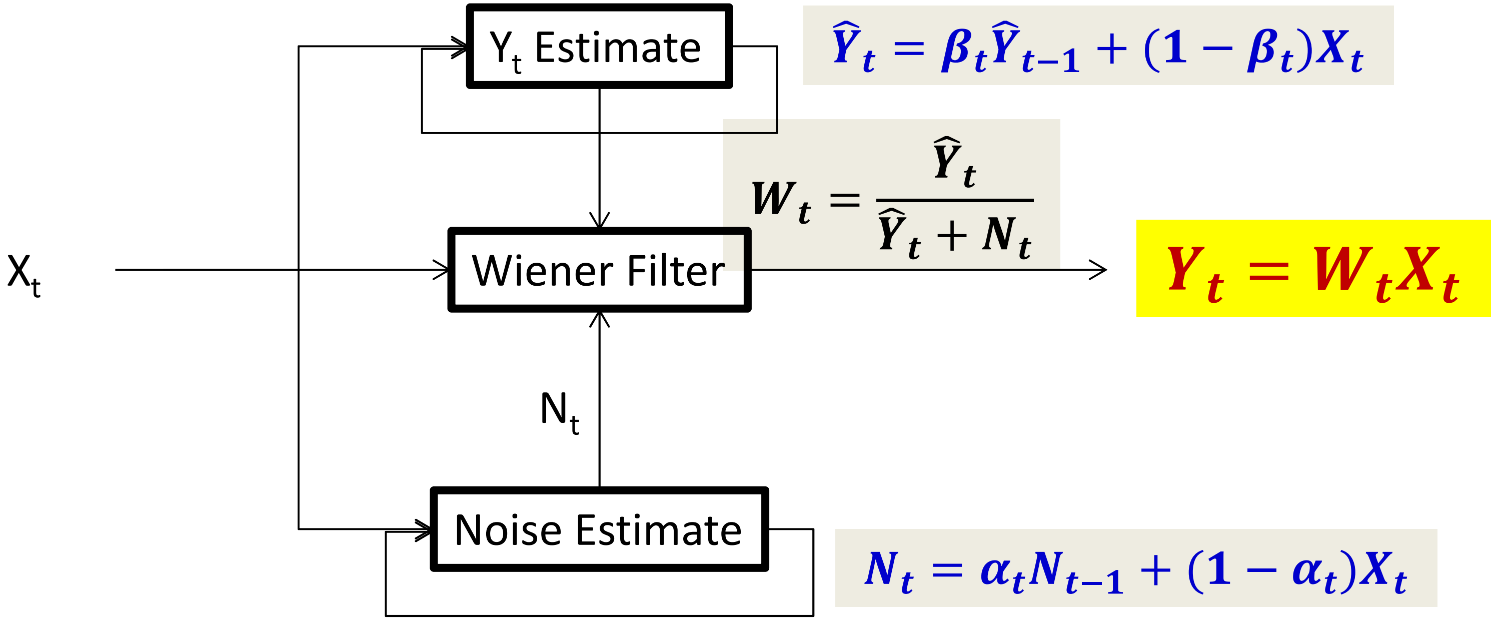
- Can NNet regressions be incorporated into these schemes?

An old faithful: Spectral Subtraction



- Estimate noise recursively
  - Update noise when noise dominates the signal
- Estimate "clean speech" recursively
  - Update when speech dominates
- Compose a filter from speech and noise estimates
- Filter the signal!

An old faithful: Spectral Subtraction



$\hat{\boldsymbol{Y}}_t = \boldsymbol{\beta}_t \hat{\boldsymbol{Y}}_{t-1} + (\mathbf{1} - \boldsymbol{\beta}_t)\boldsymbol{X}_t$

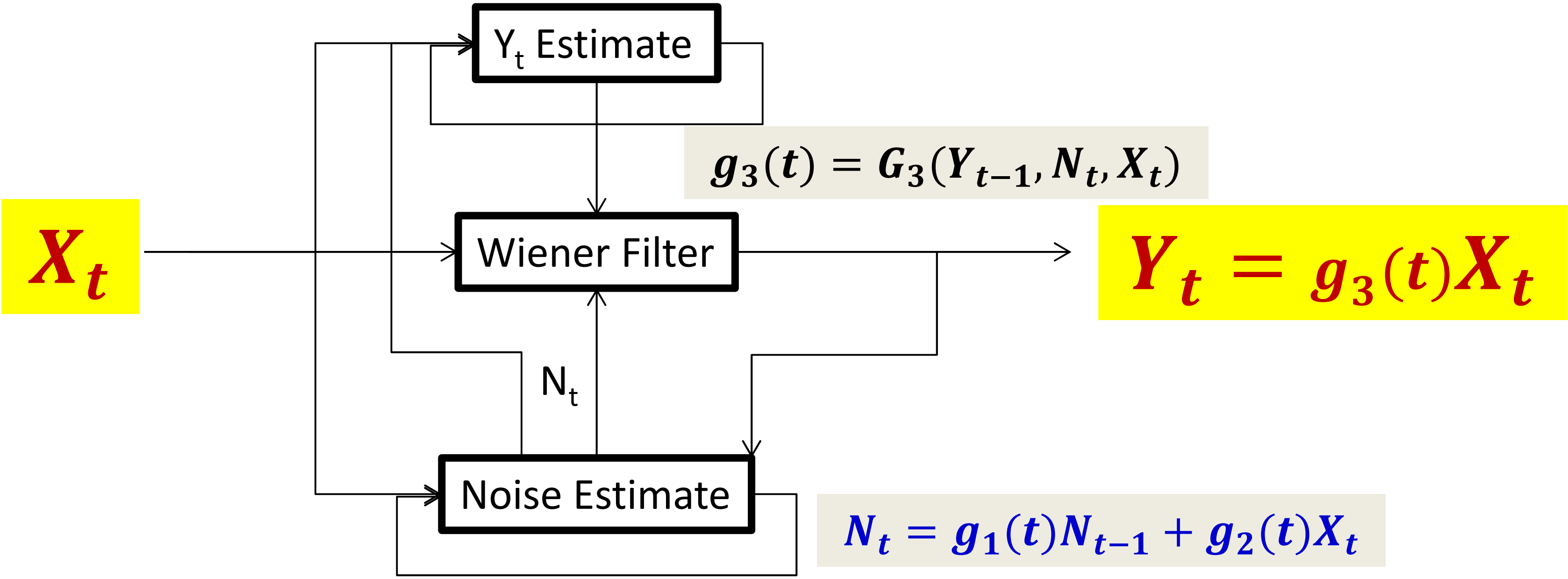
$$\boldsymbol{W}_t = \frac{\hat{\boldsymbol{Y}}_t}{\hat{\boldsymbol{Y}}_t + \boldsymbol{N}_t}$$

$\boldsymbol{Y}_t = \boldsymbol{W}_t \boldsymbol{X}_t$

$\boldsymbol{N}_t = \boldsymbol{\alpha}_t \boldsymbol{N}_{t-1} + (\mathbf{1} - \boldsymbol{\alpha}_t)\boldsymbol{X}_t$

- Estimate noise recursively
  - Update noise when noise dominates the signal
- Estimate “clean speech” recursively
  - Update when speech dominates
- Compose a filter from speech and noise estimates
- Filter the signal!

An old faithful: Spectral Subtraction



$$g_3(t) = G_3(Y_{t-1}, N_t, X_t)$$

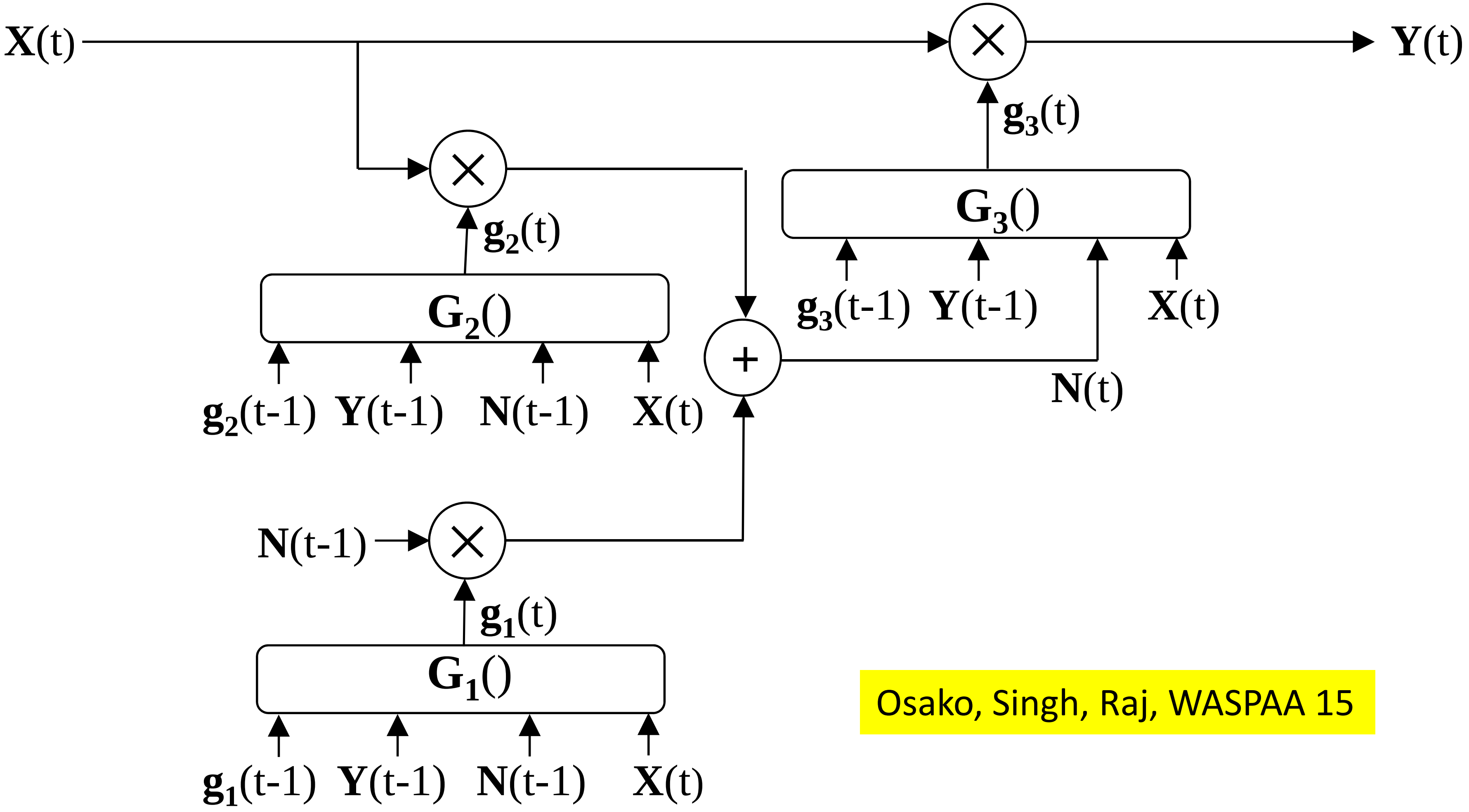
$$X_t$$

$$Y_t = g_3(t)X_t$$

$$N_t = g_1(t)N_{t-1} + g_2(t)X_t$$

- Instead of linear regression, model estimators as *learned* functions g_1, g_2 and g_3
  - *Model the functions as NNets*

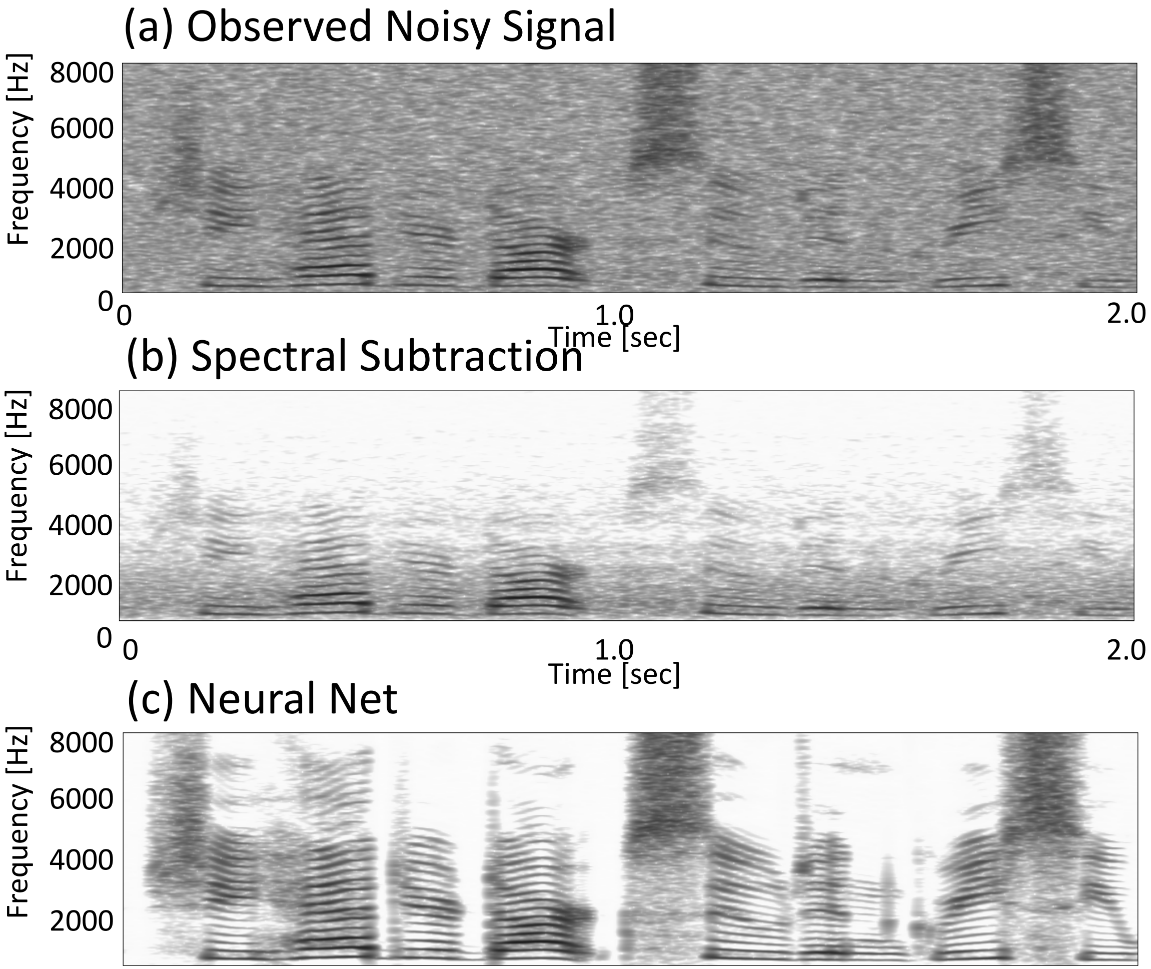
Neural Network Wiener Filter



Osako, Singh, Raj, WASPAA 15

Carnegie Mellon

Neural Network Wiener Filter



(a) Observed Noisy Signal

(b) Spectral Subtraction

(c) Neural Net

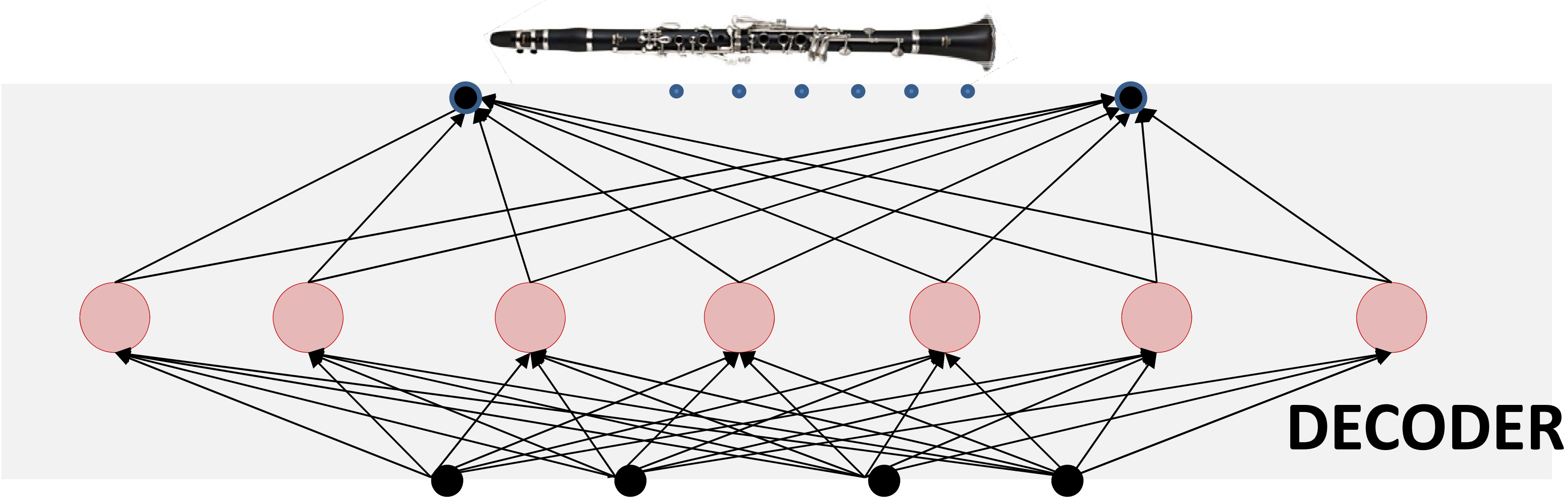
Networks:
4 layers of 128
Units

SDR improvement (over Spectral Subtraction): 8 – 10dB

Story so far

- Capabilities and Limitations of NNets
- NNets can be classifiers of unlimited versatility
- NNets can be regression functions of unlimited versatility
- NNets can be very good constructive dictionaries
- NNet classifiers can be used to enhance speech signals
- **NNet regressions can be used to enhance speech**
  - And even incorporated effectively into legacy signal processing schemes

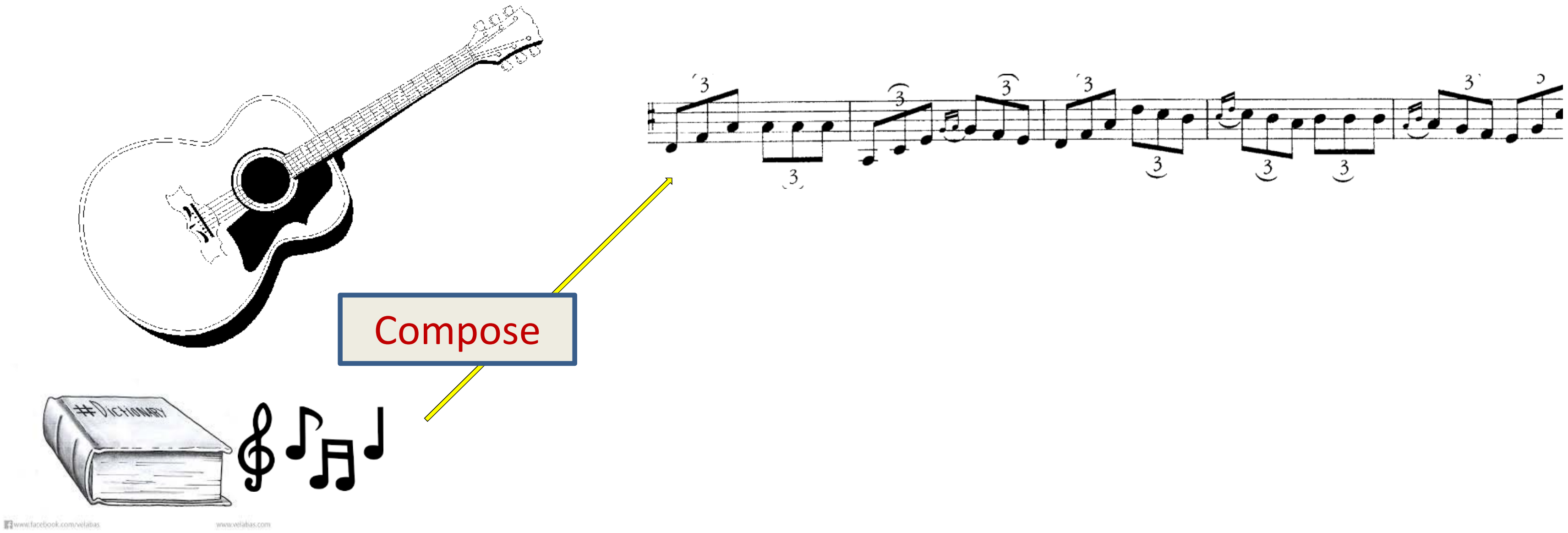
Neural Networks as Dictionaries



- Neural networks give us excellent “dictionaries”
  - – Constructive networks which, when “excited”, produce signals that are distinctly from the target source
- Use these in dictionary-based enhancement?

Carnegie Mellon

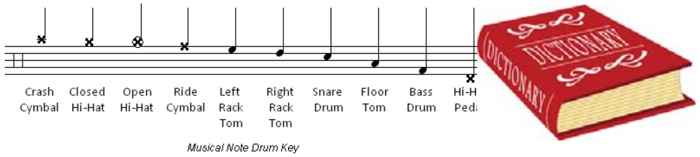
Dictionary-based techniques



- Basic idea: Learn a dictionary of “building blocks” for each sound source
- All signals by the source are composed from entries from the dictionary for the source

Dictionary-based techniques

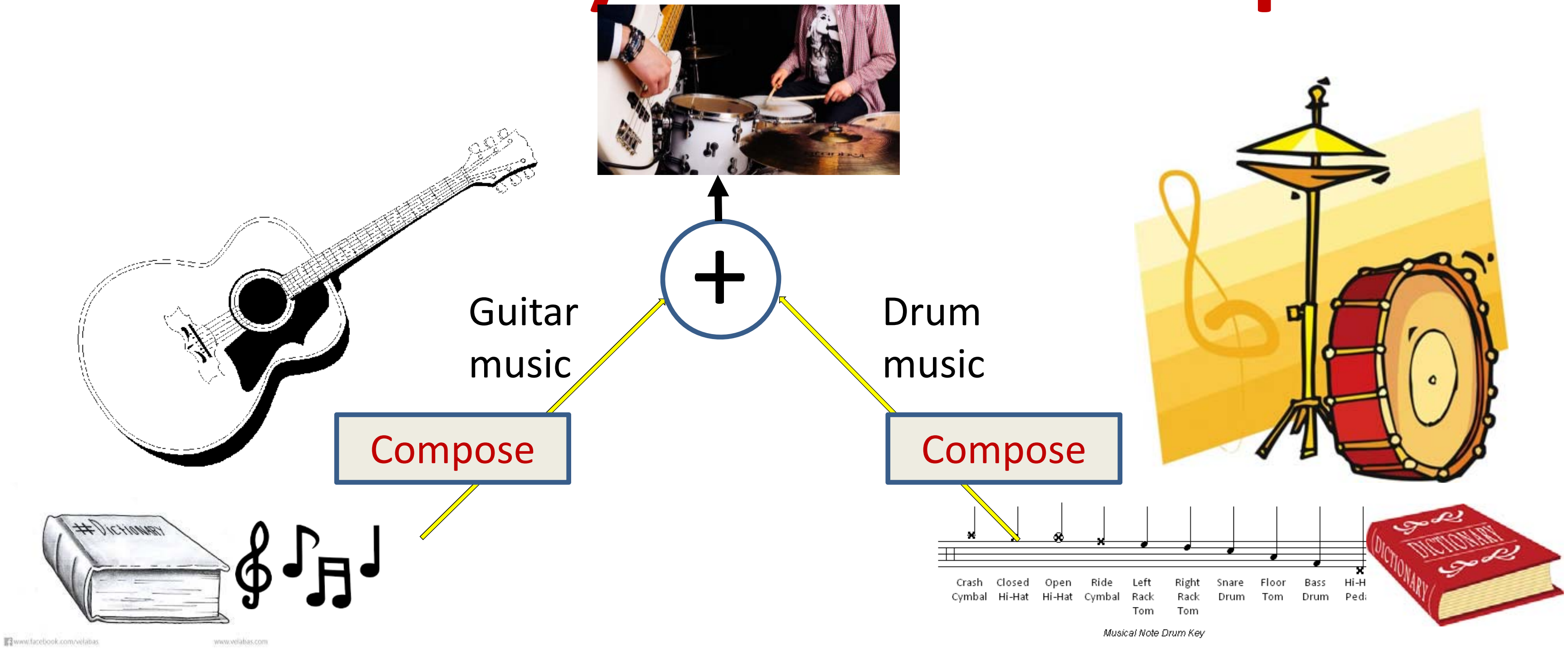
Compose



Musical Note Drum Key

- Learn a similar dictionary for all sources expected in the signal

Dictionary-based techniques



Guitar music

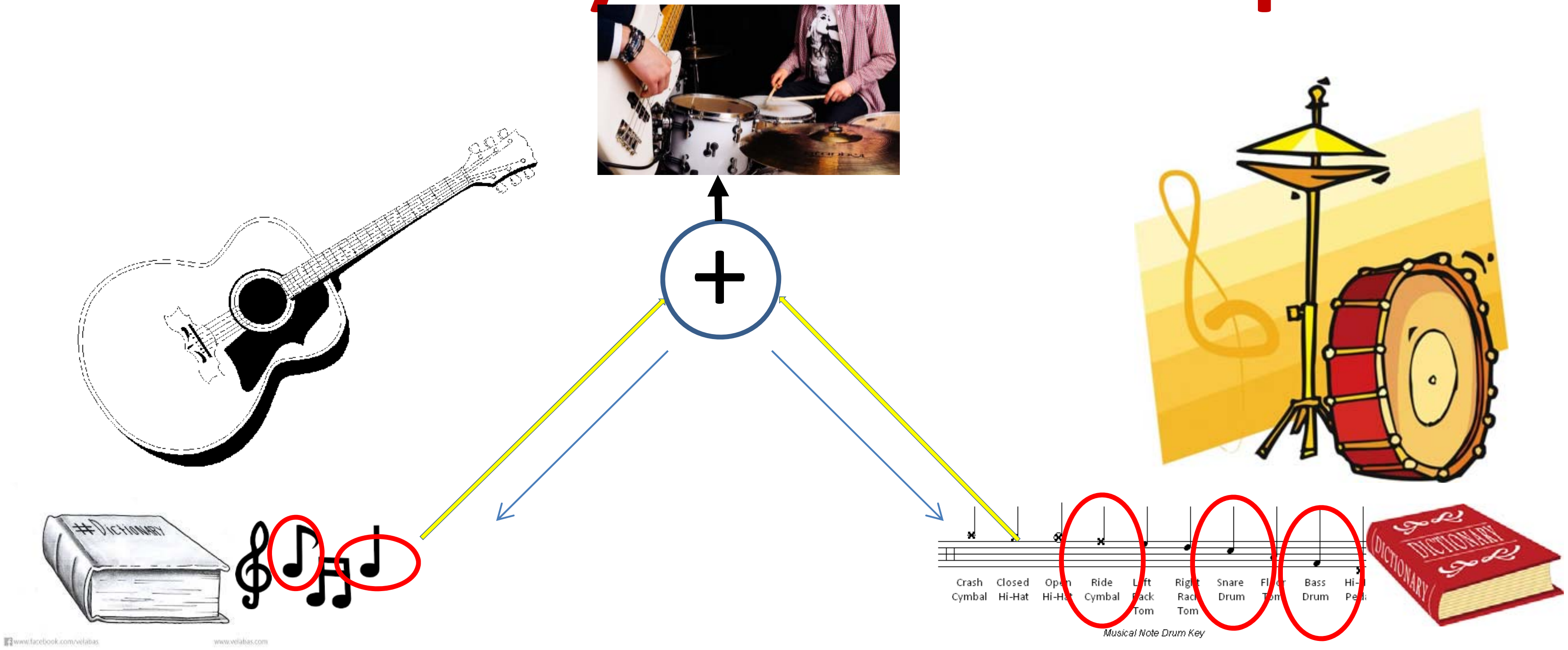
Drum music

Compose

Compose

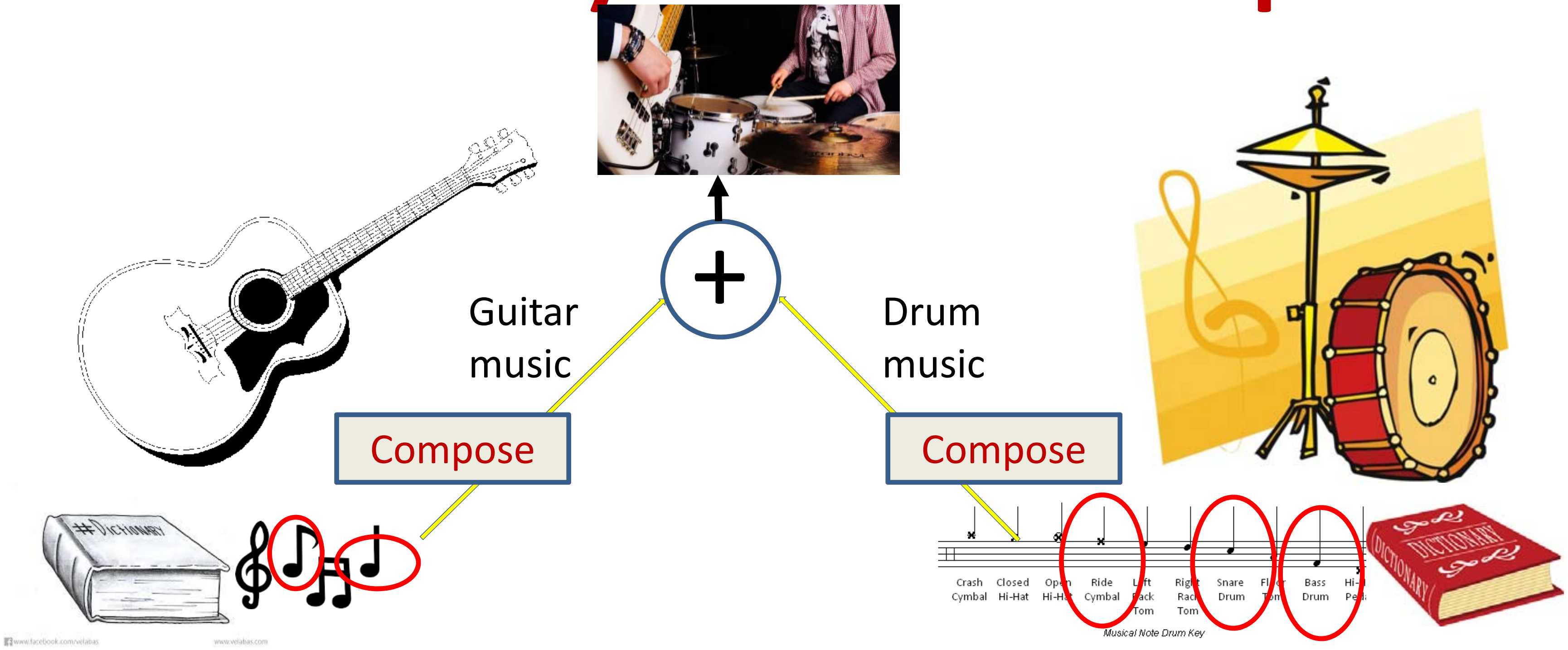
- A mixed signal is the linear combination of signals from the individual sources
  - Which are in turn composed of entries from its dictionary

Dictionary-based techniques



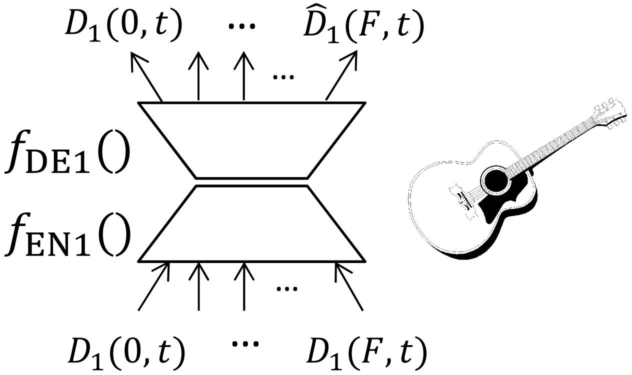
- Separation: Identify the combination of entries from both dictionaries that compose the mixed signal

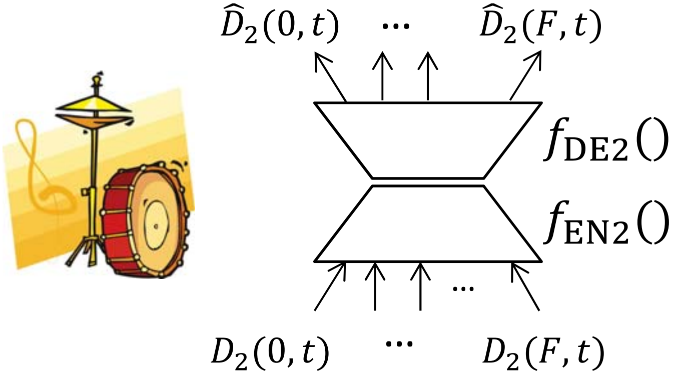
Dictionary-based techniques



- Separation: Identify the combination of entries from both dictionaries that compose the mixed signal
    - The composition from the identified dictionary entries gives you the separated signals

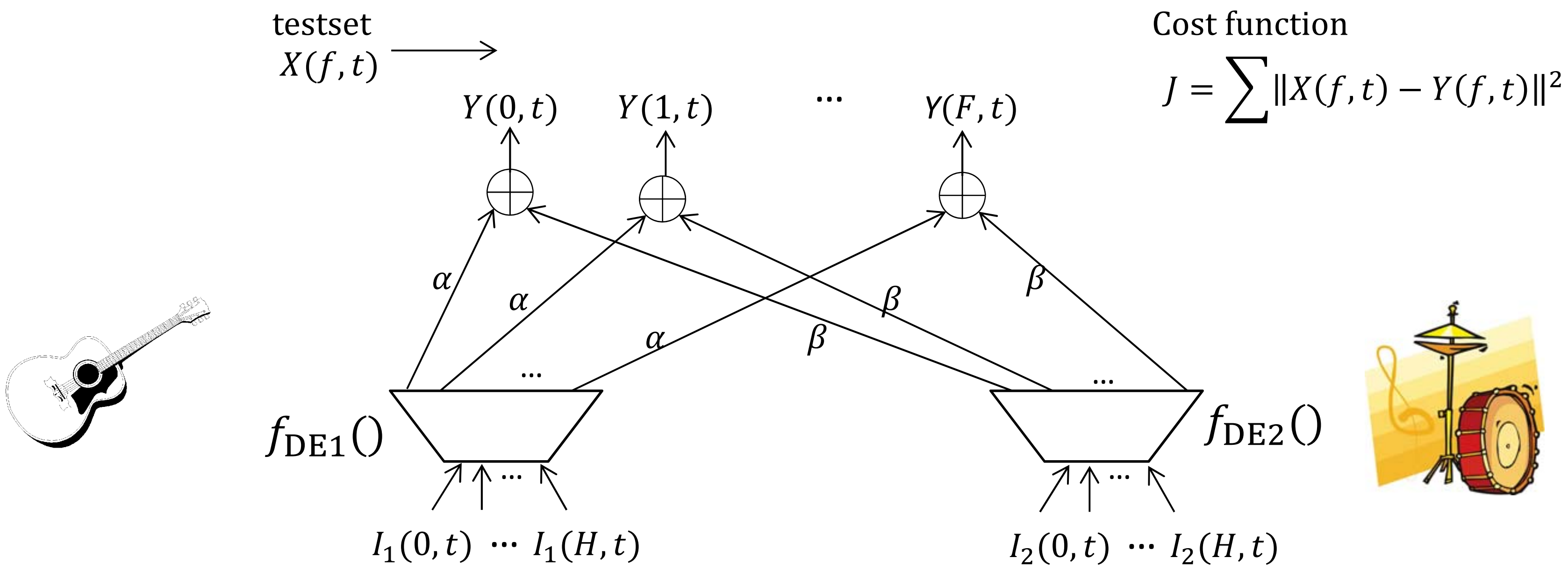
Learning Dictionaries





- Autoencoder dictionaries for each source
  - Operating on (magnitude) spectrograms
- For a well-trained network, the “decoder” dictionary is highly specialized to creating sounds for that source

Model for mixed signal



Cost function

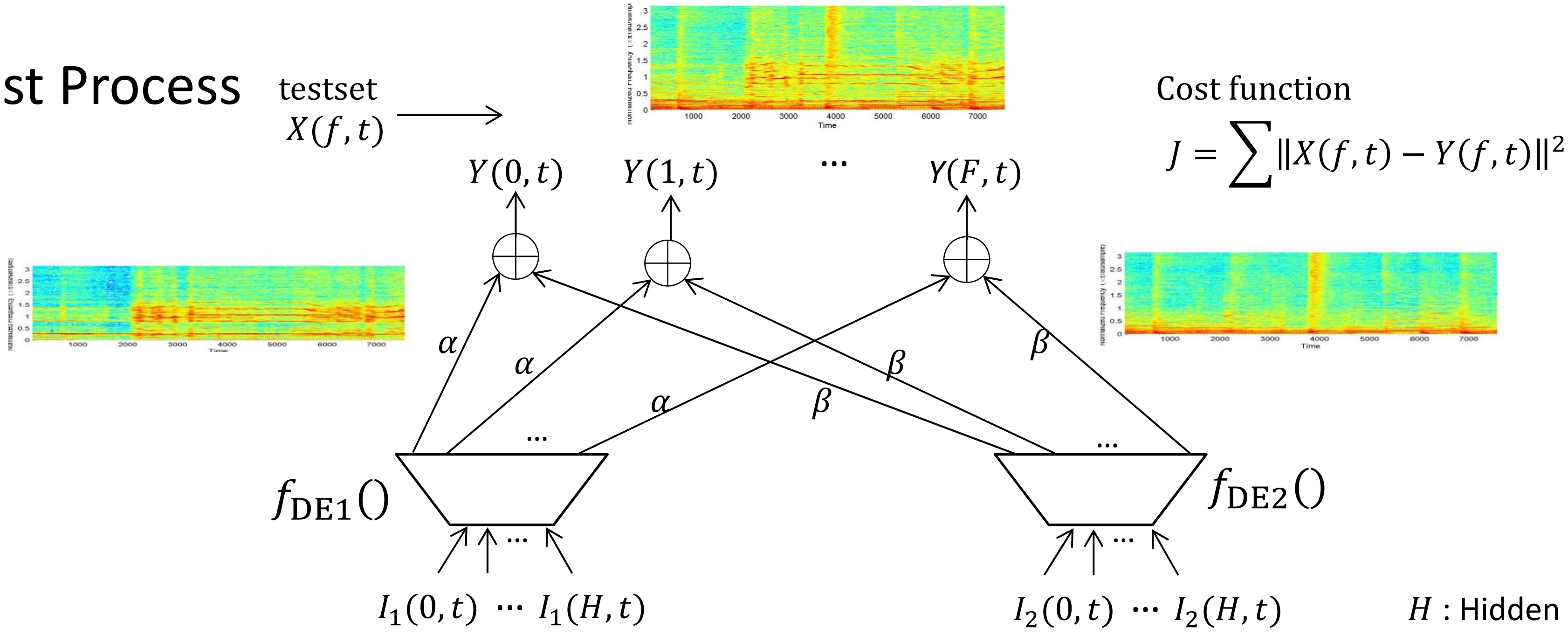
$$J = \sum \|X(f,t) - Y(f,t)\|^2$$

Estimate $I_1()$ and $I_2()$ to minimize cost function $J(\,)$

- The sum of the outputs of both neural dictionaries
  - For some unknown input

Separation

Test Process



testset

$X(f,t)$

Cost function

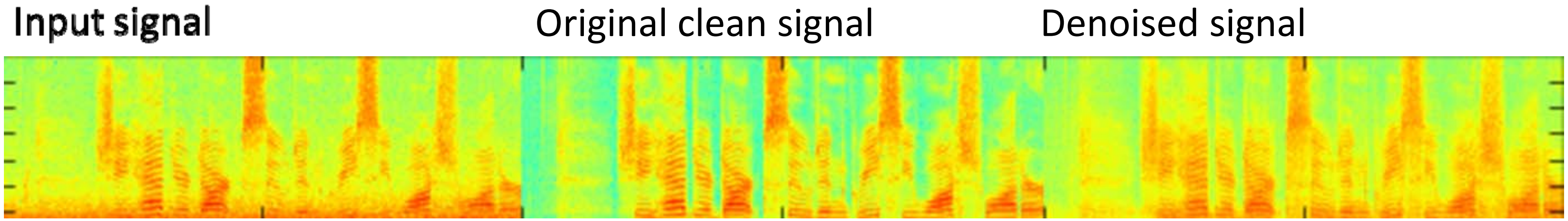
$$J = \sum \|X(f,t) - Y(f,t)\|^2$$

H : Hidden layer size

Estimate $I_1()$ and $I_2()$ to minimize cost function $J(\ )$

- Given mixed signal and source dictionaries, find excitation that best recreates mixed signal
  - Simple backpropagation
- Intermediate results are separated signals
  - Smaragdis 2016, Osako, Mitsufuji, Raj, 2016.

Example Results

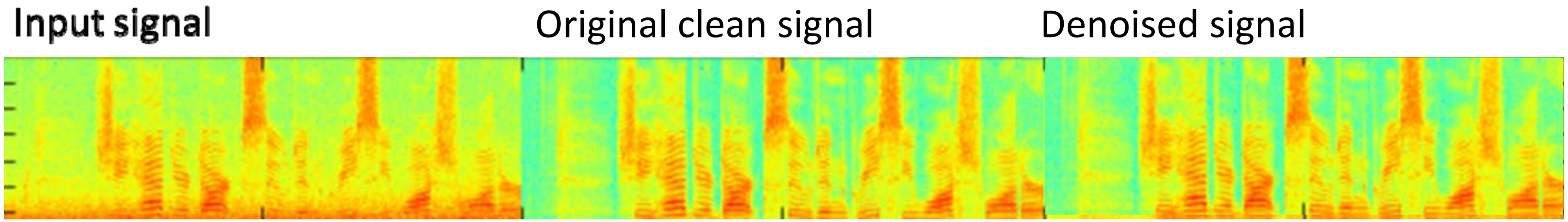


Input signal

Original clean signal

Denoised signal

Dictionary with single hidden layer of 100 neurons



Input signal

Original clean signal

Denoised signal

5-layer dictionary

- Speech in automotive noise
  – Dictionaries for speech and automotive noise

Example Results

Mixture 🔊

Separated 🔊 🔊 Original

Separated 🔊 🔊 Original

5-layer dictionary, 600 units wide

- Separating music

DNN dictionary methods

- Training dictionaries separately for each source:
  - Scaleable
  - Can easily add new sound/target source to mix
  - Can go beyond mixtures of two sounds
- Problem:
  - Does not tune dictionary for *separation*
    - Only for generation
- Extension : Discriminative training of dictionaries
  - Specialized for separation
  - Use “stereo” training data (combination of noisy and clean data)
  - Performance is superior to generative methods
  - Not scaleable, non-trivial to incorporate new sources

Summary

- We learned
  - Capabilities and Limitations of NNets
  - That NNets can be classifiers of unlimited versatility
  - That NNets can be regression functions of unlimited versatility
  - That NNets can be very good constructive dictionaries
- NNet classifiers can be used to enhance speech signals
- NNet regressions can be used to enhance speech
  - And even incorporated effectively into legacy signal processing schemes
- NNet dictionaries can be used to enhance speech

In Conclusion

- Have left out much more than I touched upon
- A lot more than what I’ve outlined
- Recurrence
- The magic of “attention”
- Beamforming – multi-channel processing
- Joint optimization of signal enhancement and speech recognition
- Unsupervised segregation of mixed signals into sources

- The work continues at a rapid pace..