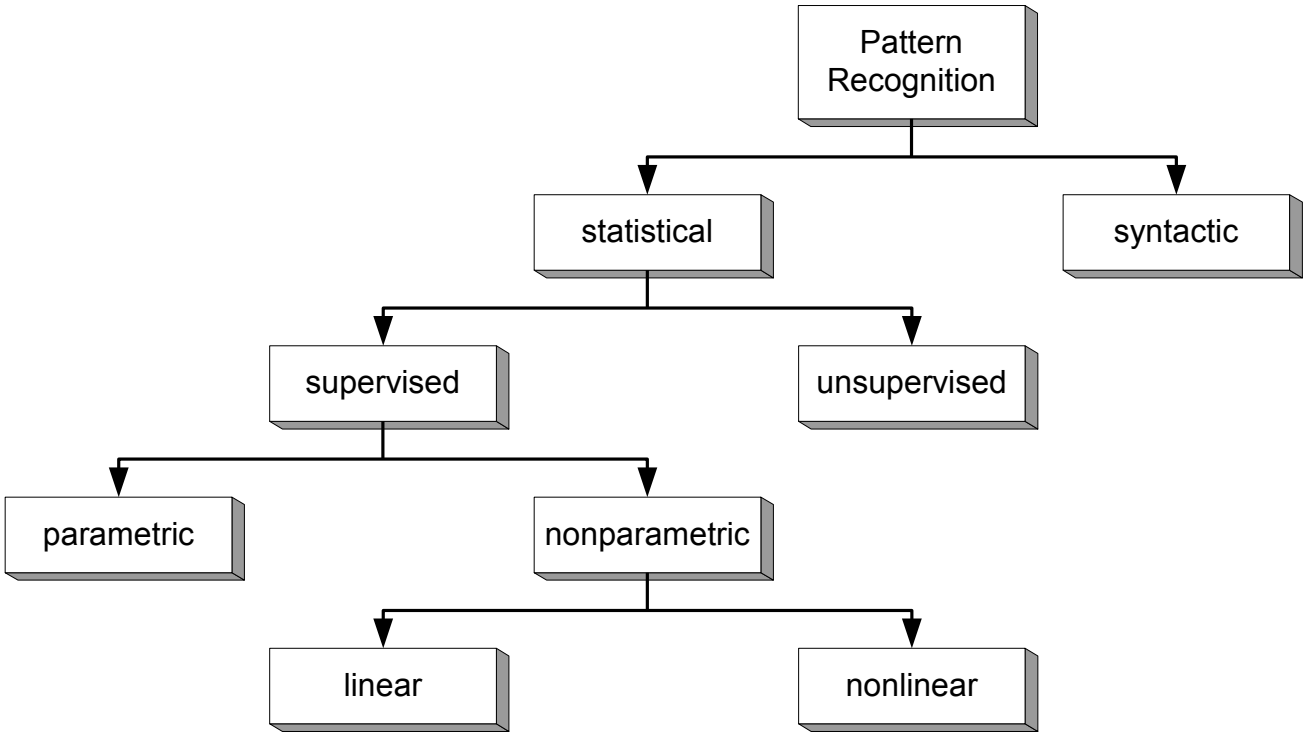
Machine Learning

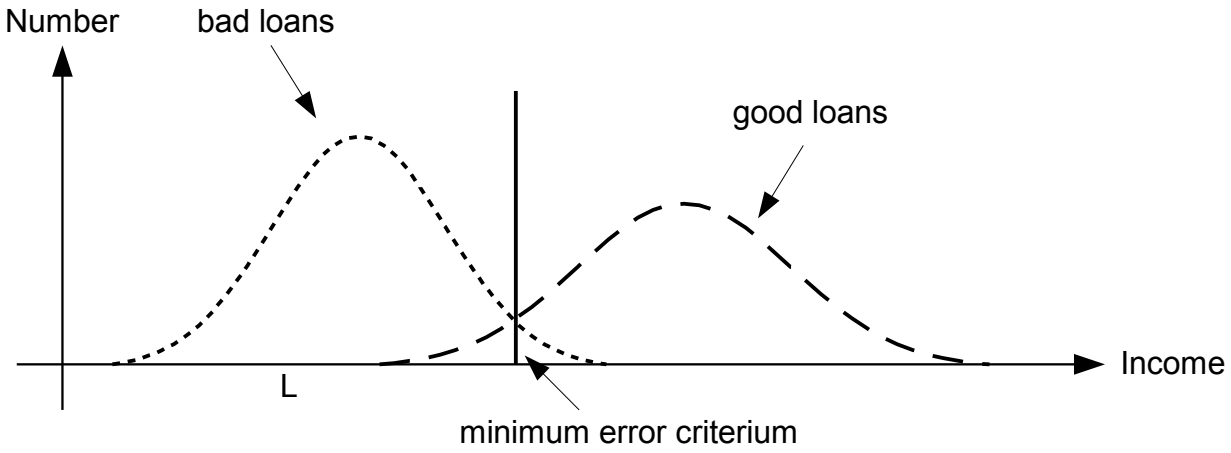
Alex Waibel

22.-29.5.2017

Pattern Recognition

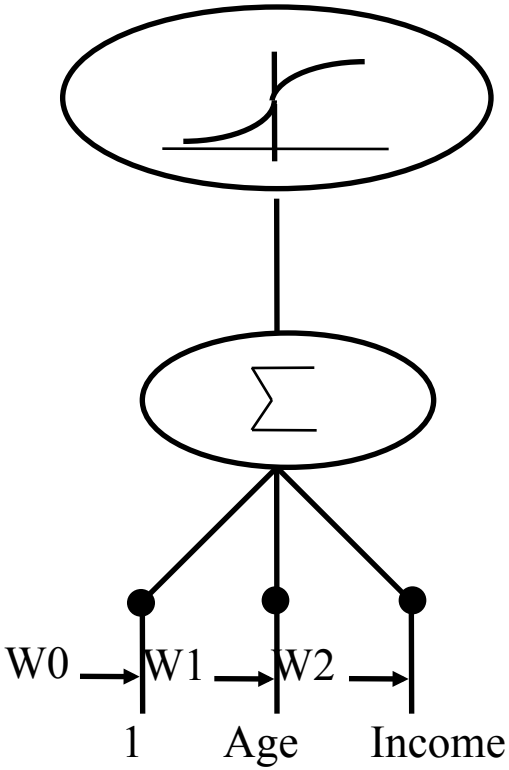


Parametric - Non-parametric



- Parametric:
  - assume underlying probability distribution;
  - estimate the parameters of this distribution.
  - Example: "Gaussian Classifier"
- Non-parametric:
  - Don't assume distribution.
  - Estimate probability of error or error criterion directly from training data.
  - Examples: Parzen Window, k-nearest neighbor, perceptron...

The Perceptron



Decision Function g(x)

$g(\vec{x}) > 0 \quad \Rightarrow$ Class A

$g(\vec{x}) < 0 \quad \Rightarrow$ Not class A

$g(\vec{x}) = 0 \quad \Rightarrow$ No decision

$$g(\vec{x}) = \sum_{i=1}^{n} w_1 x_1 + w_0 = \vec{w}^T \vec{x} + w_0 = \vec{w} \cdot \vec{x} + w_0$$

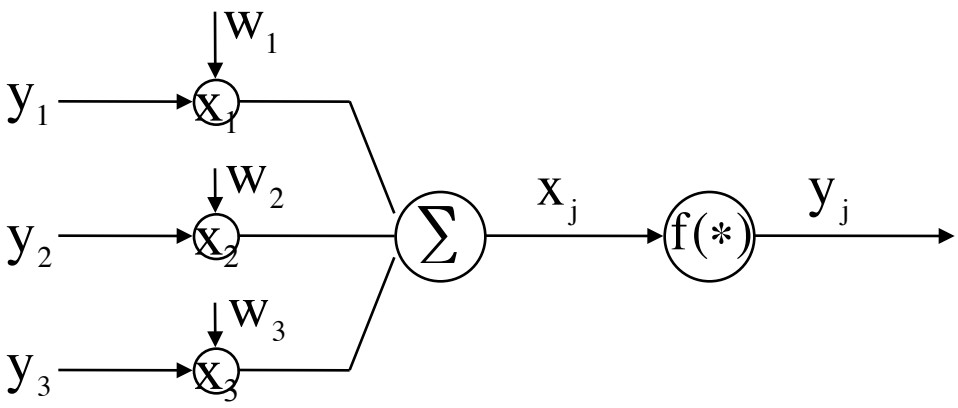
$\vec{x} = (x_1, \ldots, x_n)^T$ Feature vector

$\vec{w} = (w_1, \ldots, w_n)^T$ Weight vector

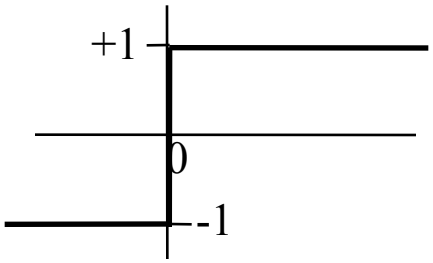
w_0 Threshold weight

$\cdot$ denotes scalar product

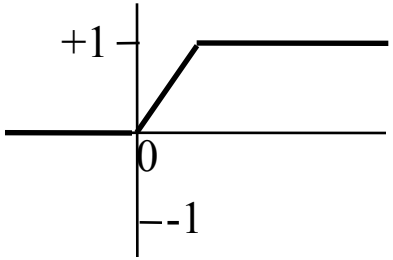
Perceptrons - Nonlinearities



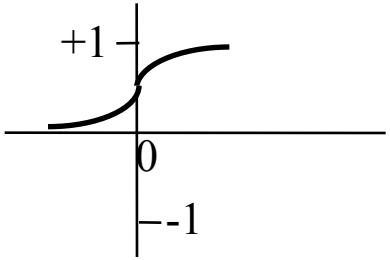
Three common non-linearities f(*):



Hard Limiter



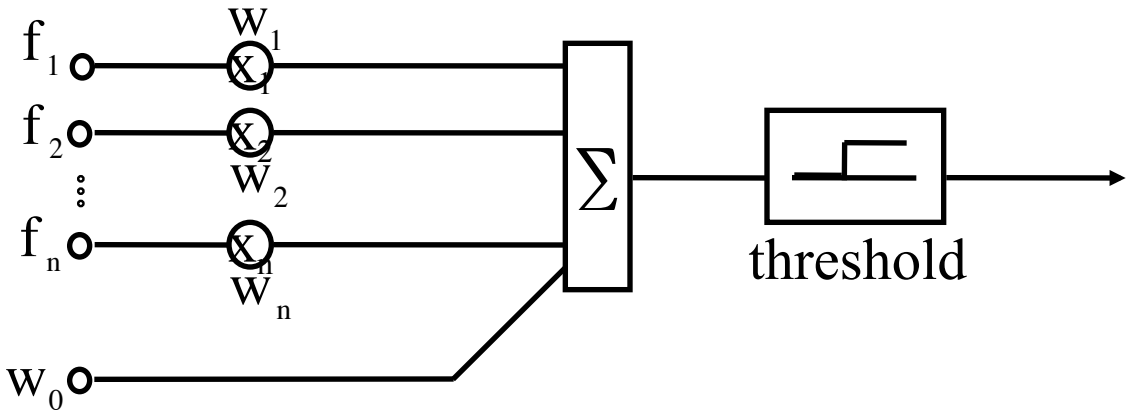
Threshold Logic



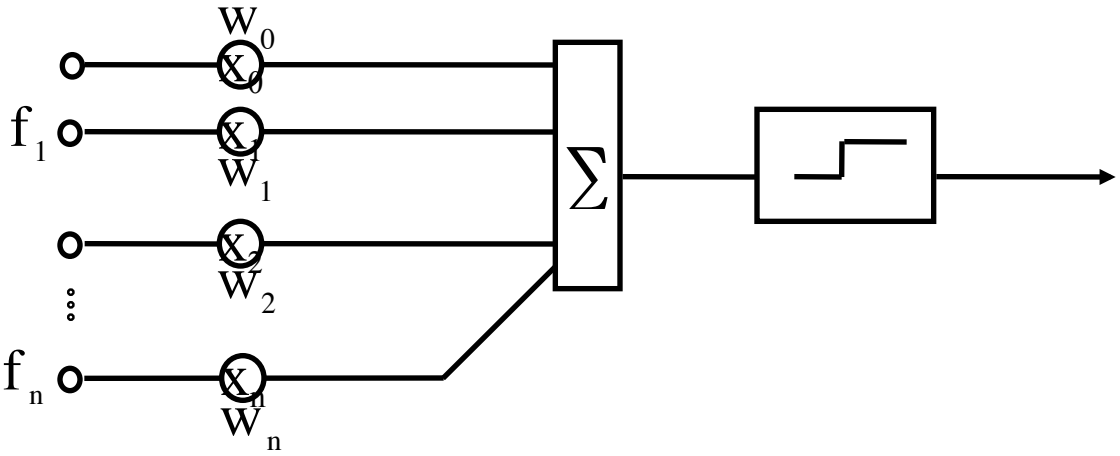
sigmoid $y_j = \frac{1}{1 + e^{-x_j}}$

The Perceptron

Linear Decision Element:



$$g(x) = w_0 + \sum_{i=1}^{n} w_i x_i$$



$$g(x) = \sum_{i=0}^{n} w_i x_i ;$$

$$x_0 = 1$$

Classifier Discriminant Functions

$$g_i(x), i = 1,\ldots,c$$

Assign x to class ω_i, if

$$g_i(x) > g_j(x) \quad \text{for all } j \neq i$$

$$g_i(x) = P(\omega_i / x)$$

$$= \frac{p(x / \omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(x / \omega_j)(\omega_j)} \longleftarrow \text{independent of class i}$$

$$g_i(x) = p(x / \omega_i)P(\omega_i)$$

$$g_i(x) = \log(p(x / \omega_i)) + \log(P(\omega_i))$$

class conditional probability density function

A priori probability

Linear Discriminant Functions

- No assumption about distributions (non-parametric)
- Linear Decision Surfaces
- Begin by supervised training (given classes of training data)
- Discriminant function:

$$g(x) = w_0 + \sum_{i=1}^{n} w_i x_i = \sum_{i=0}^{n} w_i x_i; \qquad x_0 = 1$$

- g(x) gives distance from decision surface
- Two category case:

$$g_1(x) > 0 \quad \Rightarrow \quad \text{class } 1$$

$$g_1(x) < 0 \quad \Rightarrow \quad \text{class } 2$$

Perceptron

$$g(x) = \sum_{i=0}^{n} w_i x_i ; \quad x_0 = 1$$

find $\vec{w}$

All vectors $\vec{x}_j$ are labelled correctly, if for all j

$\vec{w} \cdot \vec{x}_j > 0$ $\vec{x}_j$ if labelled ω_1

$\vec{w} \cdot \vec{x}_j < 0$ $\vec{x}_j$ if labelled ω_2

Now we set all samples belonging to ω_2 to their negative $(-\vec{x}_j)$.

Then all vectors are classified correctly, if $\vec{w} \cdot \vec{x}_j > 0$ for all j

Perceptron Criterion Function

$$J_p(\vec{w}) = \sum_{\vec{x} \in X} (-\vec{w} \cdot \vec{x})$$

X is the set of misclassified tokens

Since $\vec{w} \cdot \vec{x}$ is negation for misclassified tokens, $J_p(\vec{w})$ is positive

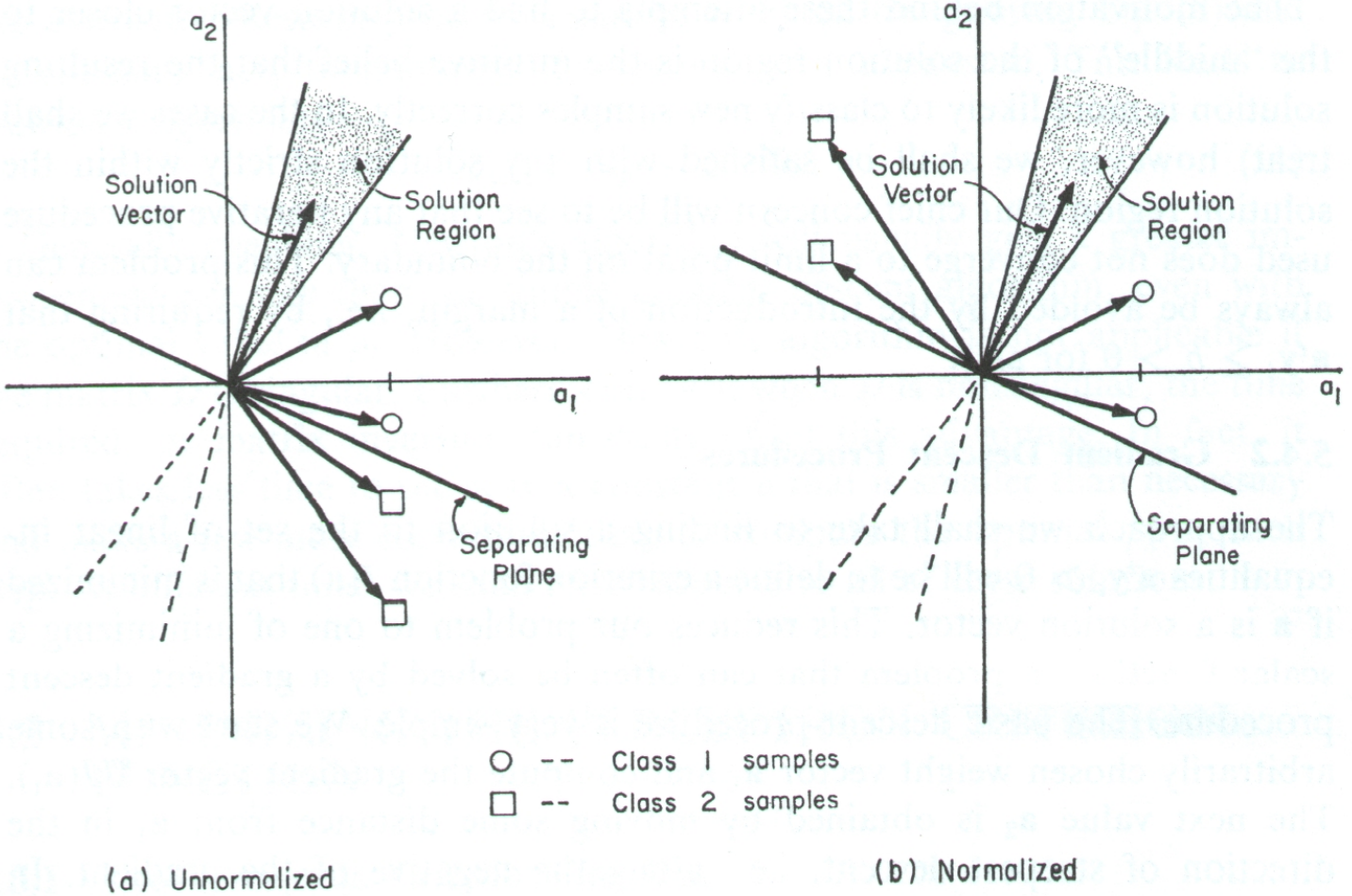
When J_p is zero a solution vector $\vec{w}$ is found.

J_p is proportional to the sum of distances of misclassified samples to decision boundary

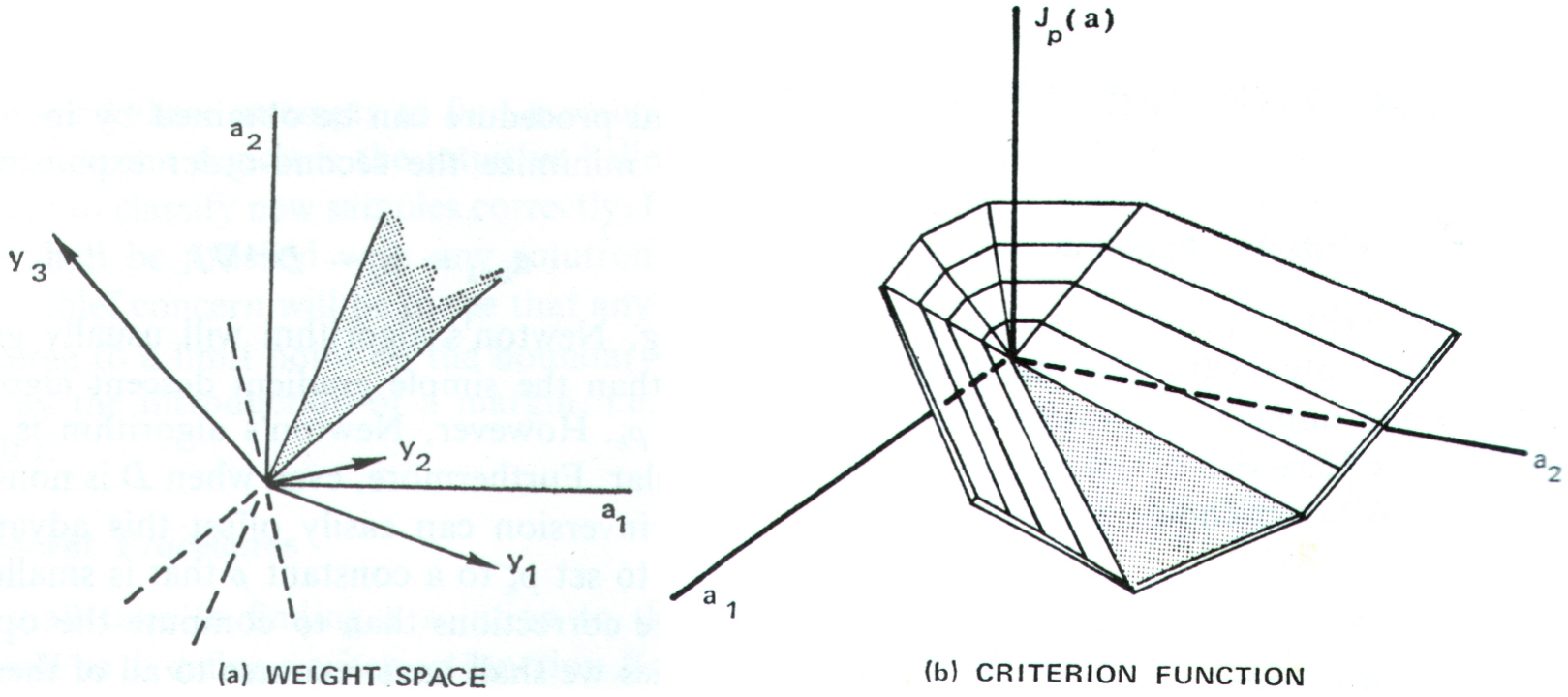
$$\nabla J_p = \sum_{\vec{x} \in X} (-\vec{x})$$

$$\vec{w}_{k+1} = \vec{w}_k + \varsigma_k \sum_{\vec{x} \in X} \vec{x}$$

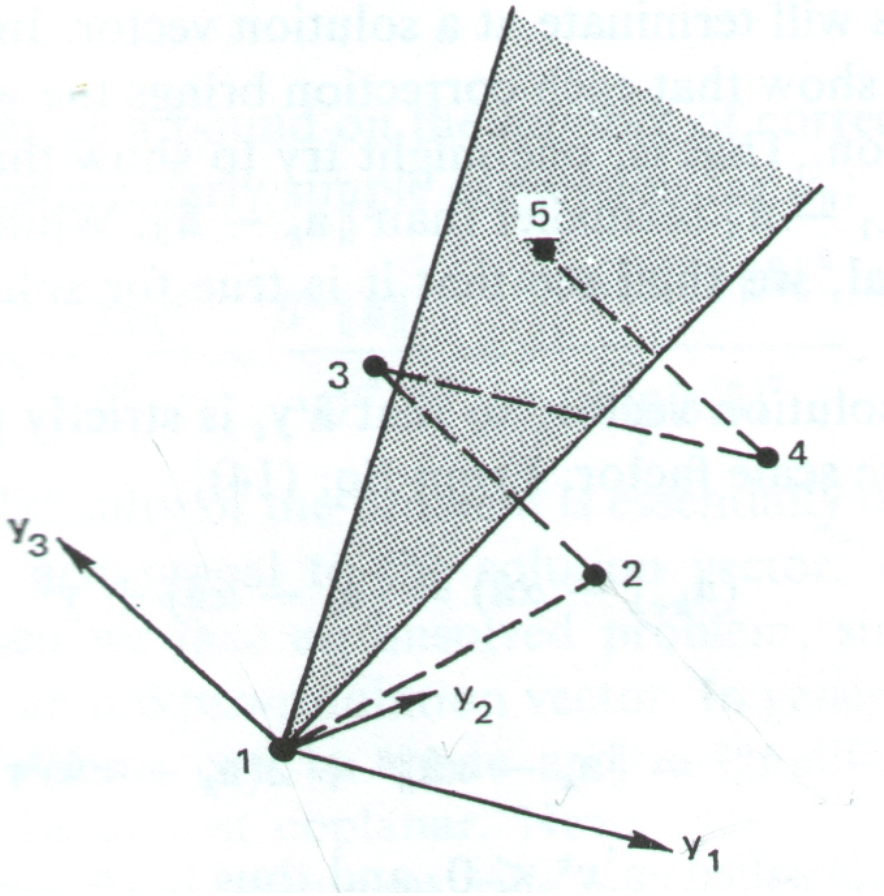
Linearly Separable Samples and the Solution Region in Weight Space



The Perceptron Criterion Function



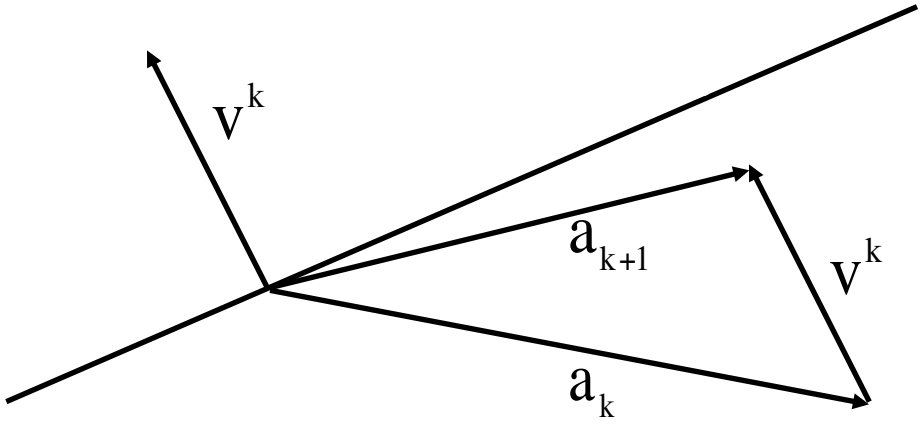
Finding a Solution Region by Gradient Search



Perceptron Learning

- Issues:
  - How to set the Learning Rate
  - How to set initial weights
- Problems:
  - Non-Separable Data
  - Separable Data, but Which Decision Surface
  - Non-Linearly Separable

Perceptron Learning



Fixed increment rule:

$$\varsigma_k = \text{constant}$$

$$\Rightarrow \dot{\underline{w}}_1 = \text{arbitrary}$$

$$\dot{\underline{w}}_{k+1} = \dot{\underline{w}}_k + \dot{\underline{y}}_k \;;\; k \geq 1$$

Variations

Relaxation Procedure

$$J_q(\vec{w}) = \sum_{\vec{x} \in X} (\vec{w} \cdot \vec{x})^2$$

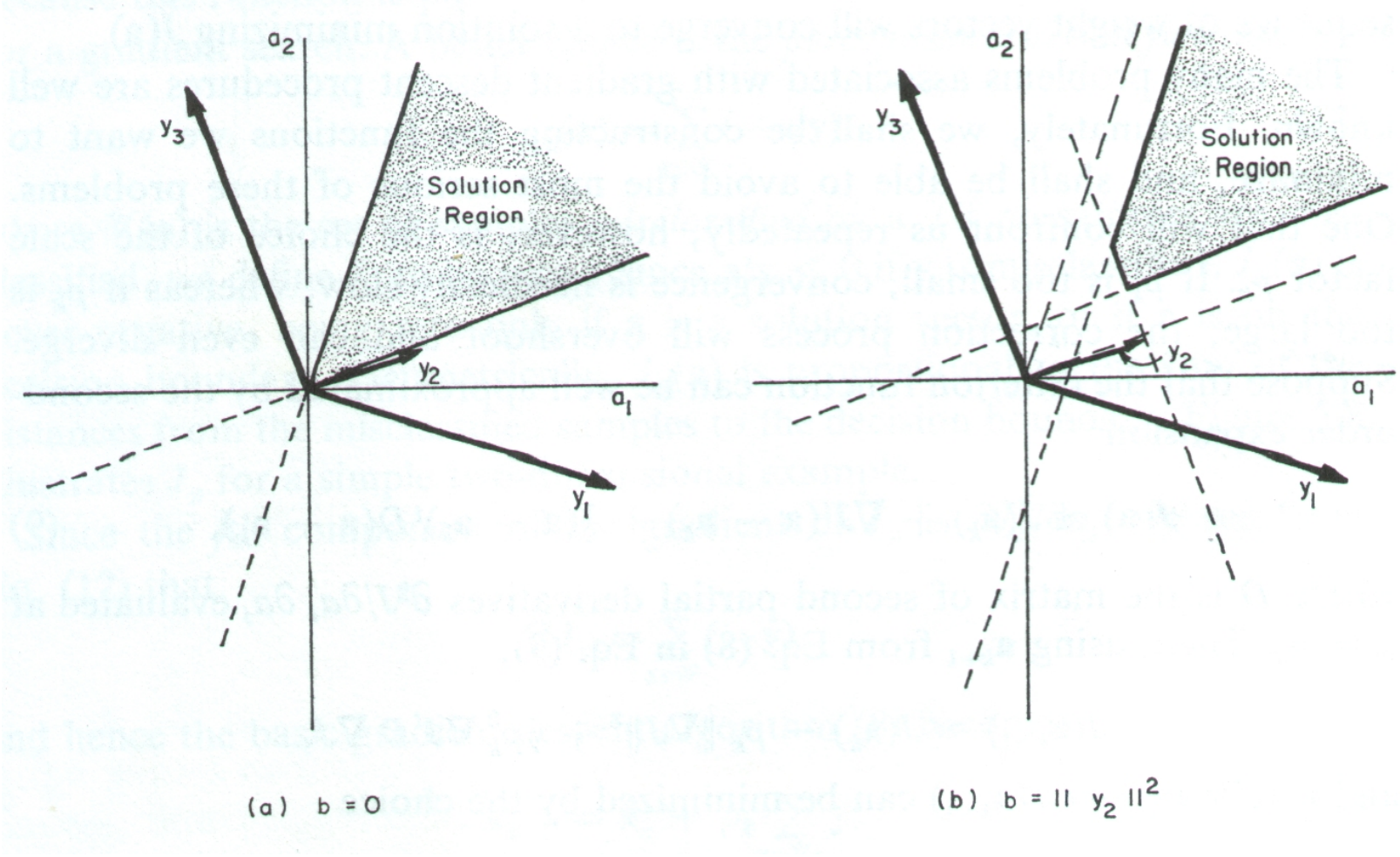
Gradient is continuous -> smoother surface

Margin

$$J_r(\vec{w}) = \frac{1}{2} \sum_{\vec{x} \in X} \frac{(\vec{x} \cdot \vec{w} - b)^2}{\|\vec{x}\|^2}$$

margin b

Effect of the Margin on the Solution Region



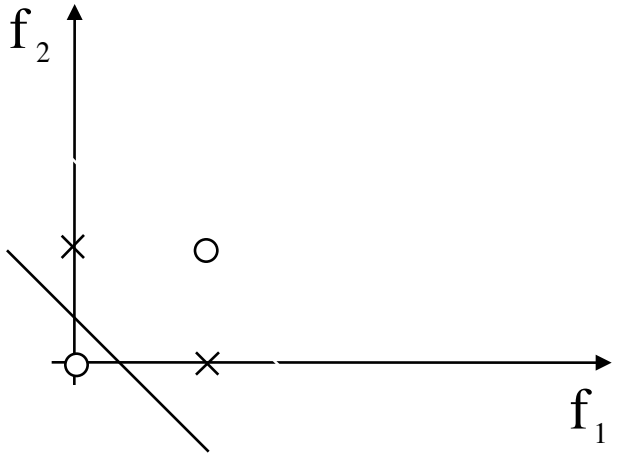
Nonseparable behavior

- Minimize MSE

$$J_s(\vec{w}) = \sum_{\vec{x}_j \in X} (\vec{w} \cdot \vec{x}_j - b)^2$$

The Perceptron:

- The good news:
  - Simple computing units
  - Learning algorithm
- The bad news:
  - Single units generate linear decision surfaces (The infamous XOR problem).



  - Multilayered machines could not be learned. Local optimization.

Neural Networks

Alex Waibel

Literaturempfehlung

- Tom Mitchell. Machine Learning. Kapitel 4
- Christoper M. Bishop. Neural Networks for Pattern Recognition. Kapitel 3 und 5
- Christopher M. Bishop. Pattern Recognition and Machine Learning

Weiterführende Veranstaltungen

- Wintersemester: Seminar Neuronale Netze, 2400078
- Sommersemester: Vorlesung Neuronale Netze, 2400024

The brain is:

- ~ 1000 supercomputers
- ~ 1/10 pocket calculator
- The brain is very good for some problems: vision, speech, language, motor-control.
- It is very poor at others: arithmetic

Neural Nets

- Terminology:
  - Artificial Neural Networks
  - Connectionist Models
  - Parallel Distributed Processing (PDP)
  - Massive Parallel Processing
  - Multi-layer Perceptron (MLP)

Neural Nets

- Terminology:
  - Artificial Neural Networks
  - Connectionist Models
  - Parallel Distributed Processing (PDP)
  - Massive Parallel Processing
  - Multi-layer Perceptron (MLP)

Von Neumann Computer - Neural Computation

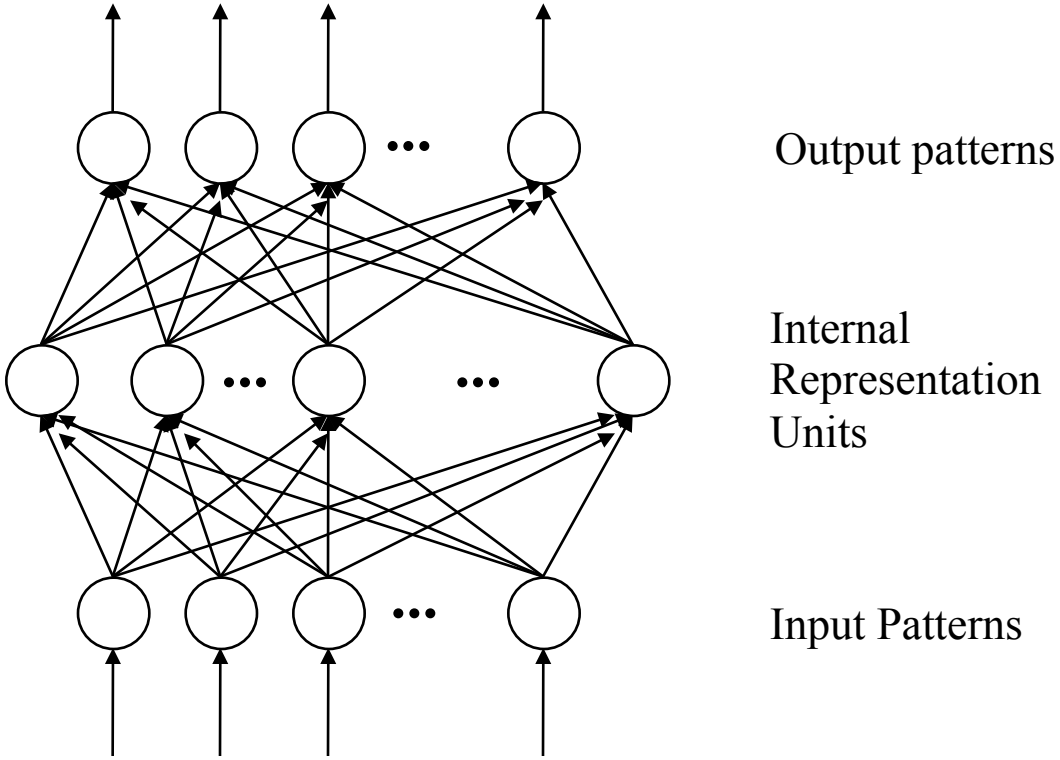
- Processing: Sequential - Parallel
- Processors: One - Many
- Interaction: None - A lot
- Communication: Poor - Rich
- Processors: Fast, Accurate - Slow, Sloppy
- Knowledge: Local - Distributed
- Hardware: General Purpose - Dedicated
- Design: Programed - Learned

Why Neural Networks

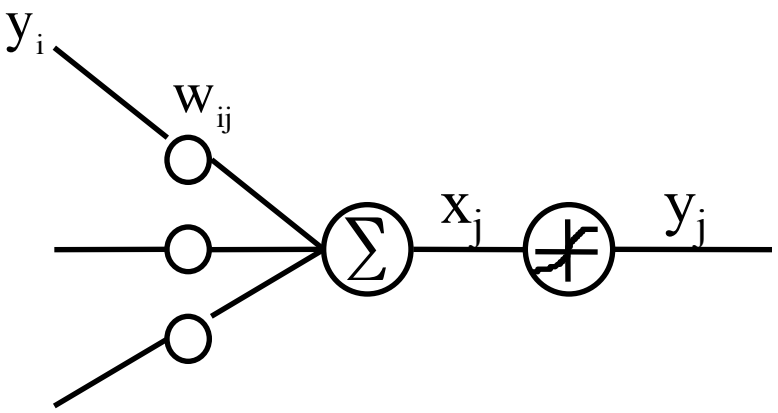
- Massive parallelism.
- Massive constraint satisfaction for ill-defined input.
- Simple computing units.
- Many processing units, many interconnections.
- Uniformity (-> sensor fusion)
- Non-linear classifiers/ mapping (-> good performance)
- Learning/ adapting
- Brain like ??

Networks of Neurons/ Multi-Layer Perceptron

- Many interconnected simple processing elements:

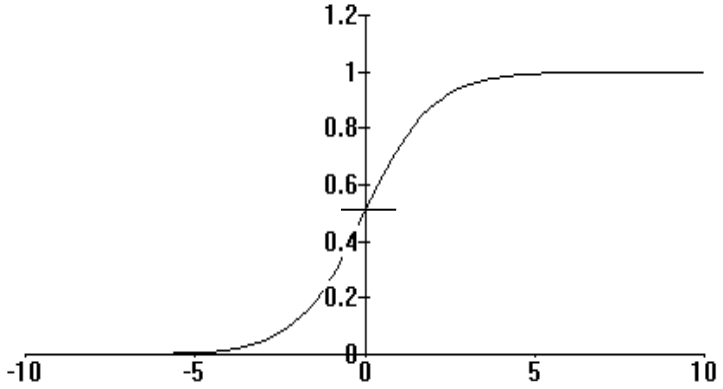


Connectionist Units



$$x_j = \sum_i y_i w_{ij}$$

$$y_j = \frac{1}{1 + e^{-x_j}}$$



Backpropagation of error:

$$E = \frac{1}{2} \sum_j (y_j - d_j)^2$$

Training the MLP by Error Back-Propagation

- Choose random initial weights
- Apply input, get some output
- Compare output to *desired* output and compute error
- Back-propagate error through net and compute $\partial E / \partial w_{ij}$, the contribution of each weight to the overall error.
- Adjust weights slightly to reduce error.

Backpropagation of Error

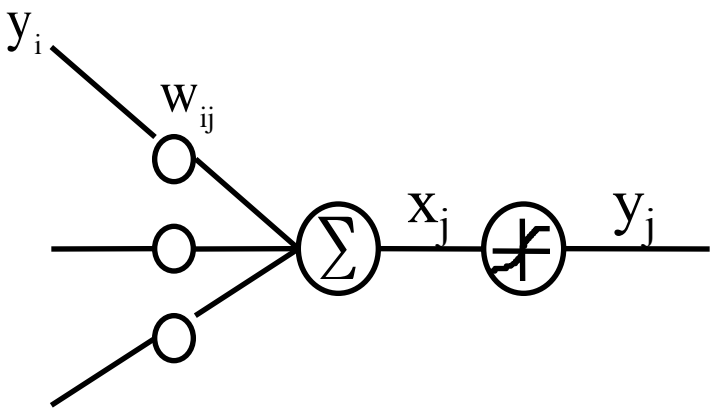
$$E = \frac{1}{2}\sum_j (y_j - d_j)^2 \qquad y_j = \frac{1}{1+e^{-x_j}} \qquad x_j = \sum_i y_i w_{ij}$$

1). $$\frac{\partial E}{\partial y_j} = y_j - d_j$$

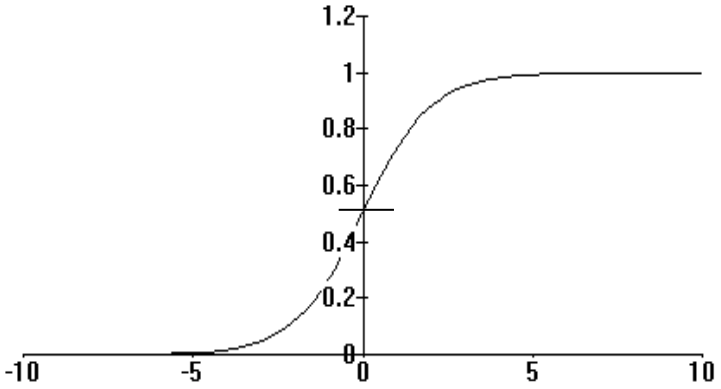
2). $$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_j} = \frac{\partial E}{\partial y_j} y_j [1 - y_j]$$

3). $$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} \cdot y_i$$

4). $$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} \cdot \frac{\partial x_j}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} \cdot w_{ij}$$



Derivative dy/dx



$$\text{sigmoid } y_j = \frac{1}{1+e^{-x_j}}$$

$$\frac{\partial y_j}{\partial x_j} = (-1)(1+e^{-x_j})^{-2} \cdot (-1)e^{-x_j}$$

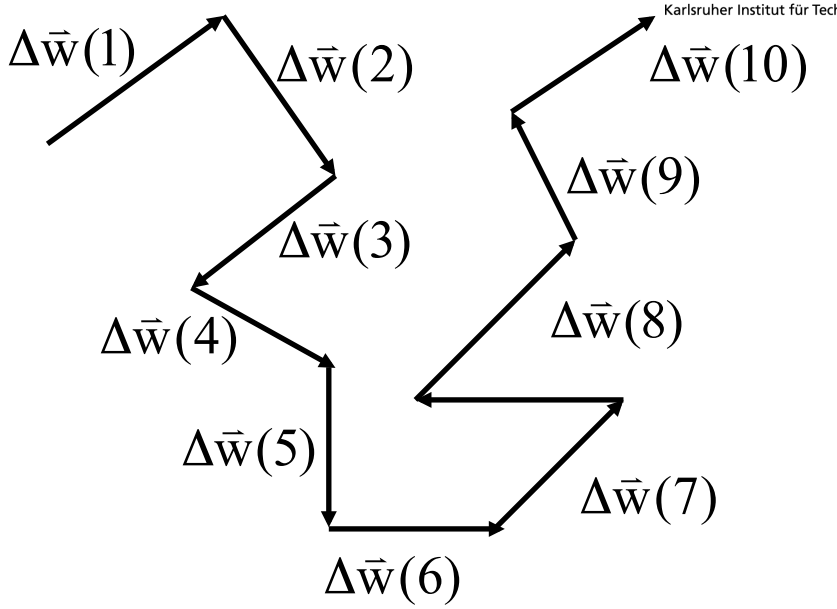
$$= \frac{e^{-x_j} \cdot 1}{(1+e^{-x_j}) \cdot (1+e^{-x_j})}$$

$$= \frac{e^{-x_j}}{1+e^{-x_j}} \cdot y_j$$

$$= y_j(1-y_j)$$

Momentum

$$\Delta w_{ij}(t) \sim \frac{\partial E}{\partial w_{ij}}$$



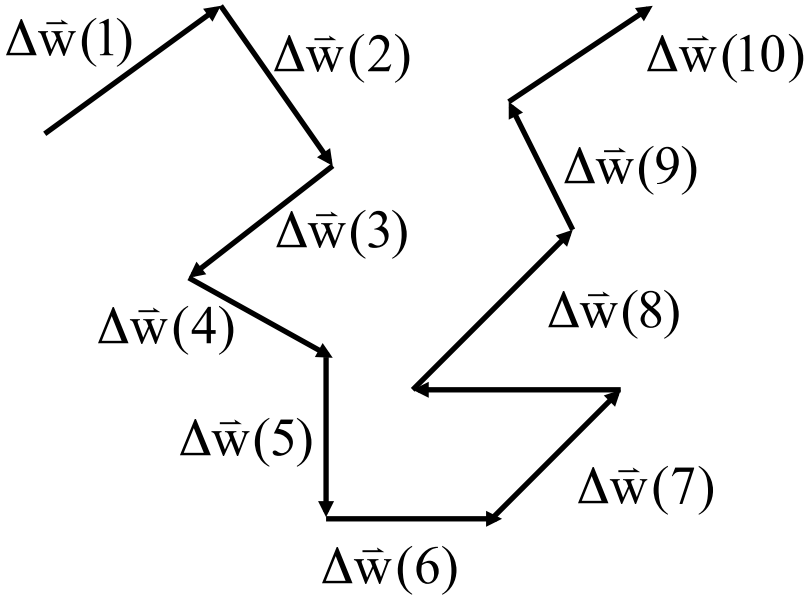
$$\Delta w_{ij}^{*}(t+1) = \Delta w_{ij}(t+1) + \alpha \Delta w_{ij}(t)$$

Step Size

$$\Delta w_{ij}(t) \sim \frac{\partial E}{\partial w_{ij}}$$

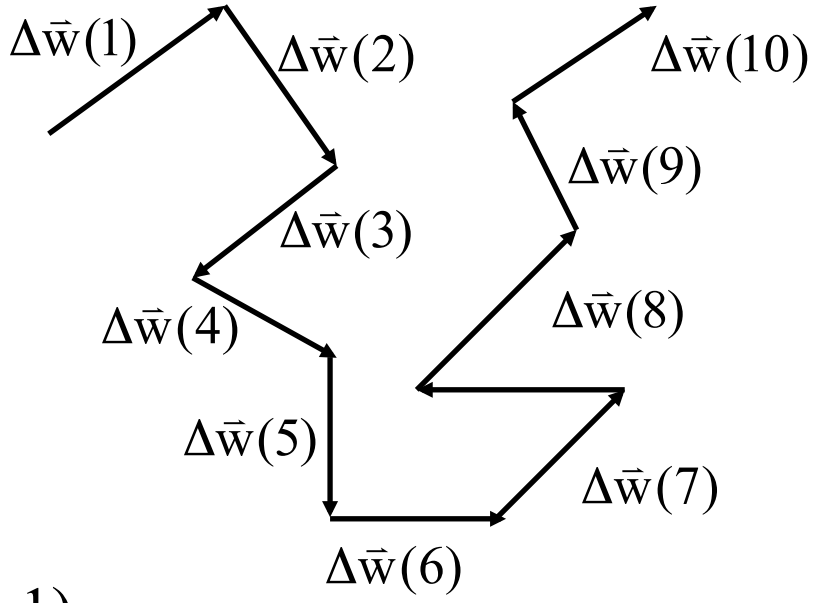
$$\Delta w_{ij}(t) = - \varepsilon \, \partial E / \partial w_{ij}(t)$$

Step Size
0...1



Step Size and Momentum

$$\Delta w_{ij}(t) \sim \frac{\partial E}{\partial w_{ij}}$$



$$\Delta w_{ij}(t) = - \varepsilon \, \partial E/\partial w_{ij}(t)$$

$$\Delta w_{ij}(t) = - \varepsilon \, \partial E/\partial w_{ij}(t) + \alpha \, \Delta w_{ij}(t-1)$$

Step Size
0...1

Momentum
0...1

Statistical Interpretation of MLP's

- What is the output of an MLP?
- Output represents a Posteriori Probabilities P(w|x)
- Assumptions:
  - Training Targets: 1, 0
  - Output Error Function: Mean Squared Error
  - Other Functions Possible

Statistical Interpretation of MLP's

- What is the output of an MLP?
- It can be shown, that the output units represent the a posteriori probability
  - Provided certain objective functions (e.g. MSE, ..)
  - Sufficiently large training database

Statistical Interpretation of MLP's

- What is the output of an MLP?
- Using MSE, for two class problem, $c_1 \doteq 1$, $c_2 \doteq 0$

$$E = \frac{1}{N}\left[\sum_{x \in c_1}[f(x)-1]^2 + \sum_{x \in c_2}[f(x)]^2\right]$$

if N large and reflects prior distribution

$$E \approx \int (f(x)-1)^2 p(x,c_1)dx + \int (f(x))^2 p(x,c_2)dx$$

Statistical Interpretation of MLP's

$$E = \int f^2[p(x_1 c_1) + p(x_1 c_2)]dx - 2\int f(x)p(x_1 c_1)dx + 1 \cdot \int p(x_1 c_1)dx$$

$$= \int f^2(x)p(x)dx - 2\int f(x)p(x_1 c_1)dx + \int p(x_1 c_1)dx$$

$$= \int [f^2(x)p(x) - 2f(x)p(x_1 c_1) + p(x_1 c_1)]dx$$

$$= \int [f^2(x) - 2f(x)p(c_1 / x) + p(c_1 / x)]p(x)dx$$

$$E = \int \underbrace{[f(x) - p(c_1 / x)]^2}_{\text{net approx. posterior}} p(x)dx - \int p^2(c_1 / x)p(x)dx + P(c_1)$$

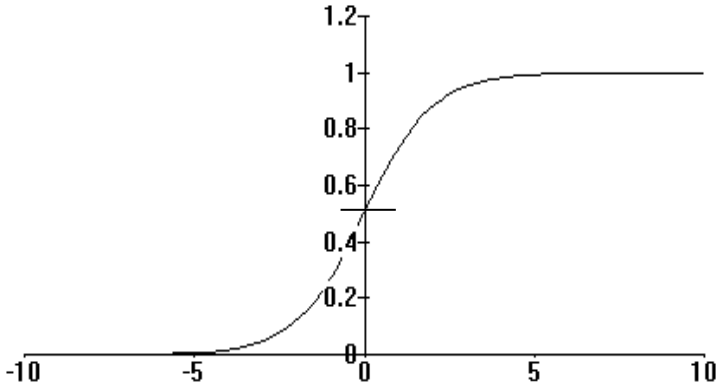
What does the sigmoid do?

$$y_j = \frac{1}{1 + e^{-x_j}}$$

$$e^{-x_j} = \frac{1}{y_j} - 1$$

$$e^{x_j} = \frac{y_j}{1 - y_j}$$

$$x_j = \log\left[\frac{y_j}{1 - y_j}\right]$$



$$\text{sigmoid } y_j = \frac{1}{1 + e^{-x_j}}$$

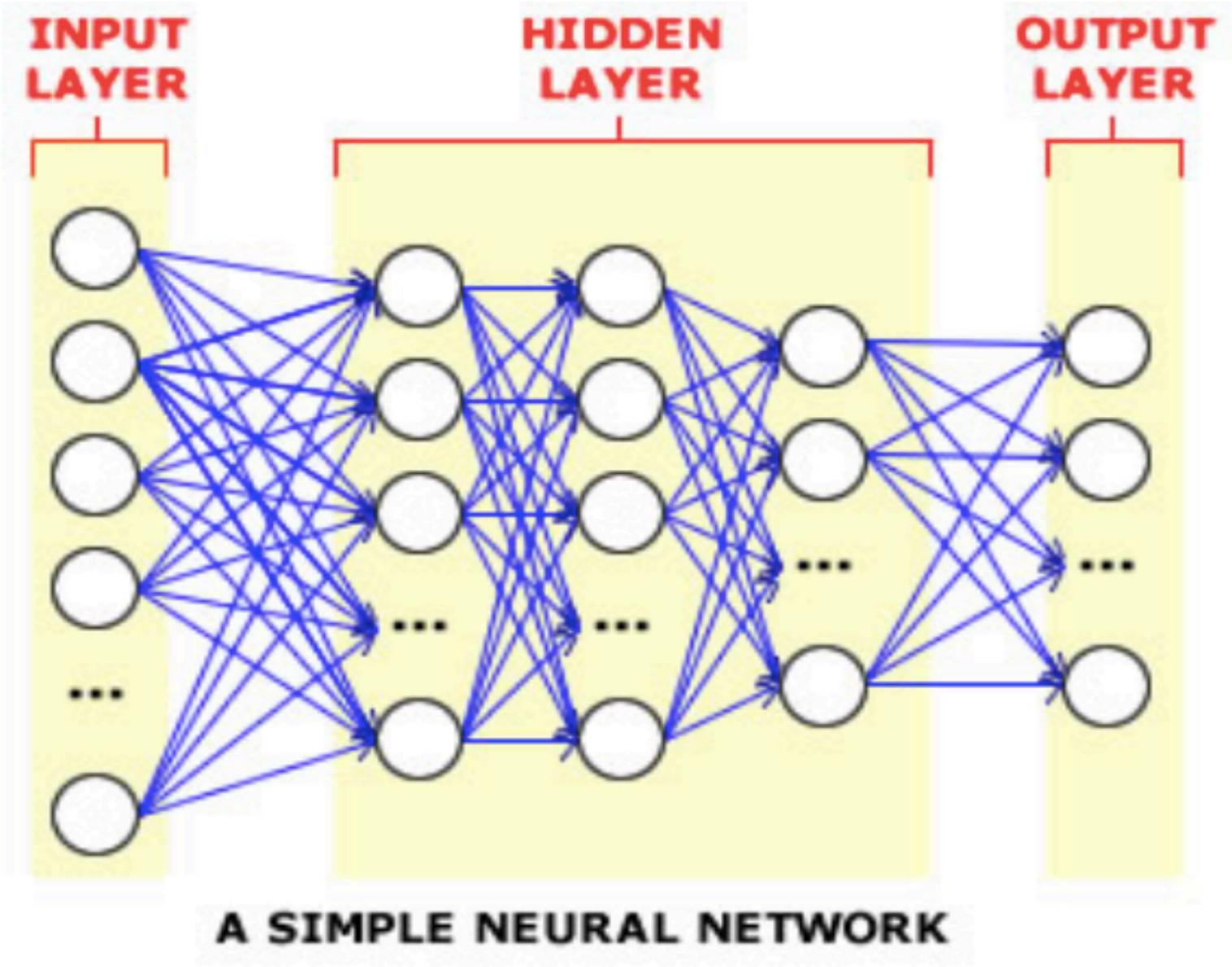
What does the sigmoid do?

If $y_j = p(q / x)$ **then**

$$x_j = \log[\frac{p(q / x)}{p(\bar{q} / x)}]$$

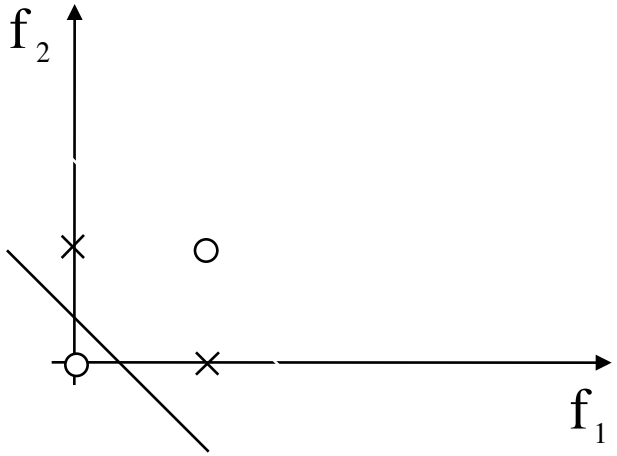
$$= \underbrace{\log[\frac{p(x / q)}{p(x / \bar{q})}]}_{\text{inputs}} + \underbrace{\log[\frac{p(q)}{p(\bar{q})}]}_{\text{Bias}}$$

log. likelihood ratio or "odds"


INPUT LAYER
HIDDEN LAYER
OUTPUT LAYER
A SIMPLE NEURAL NETWORK

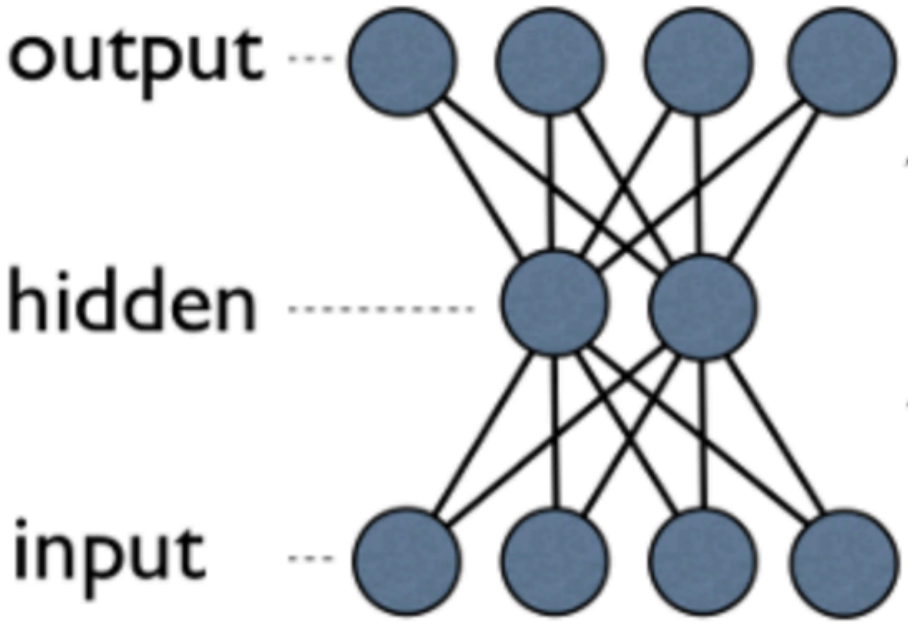
The Perceptron:

- The good news:
  - Simple computing units
  - Learning algorithm
- The bad news:
  - Single units generate linear decision surfaces (The infamous XOR problem).

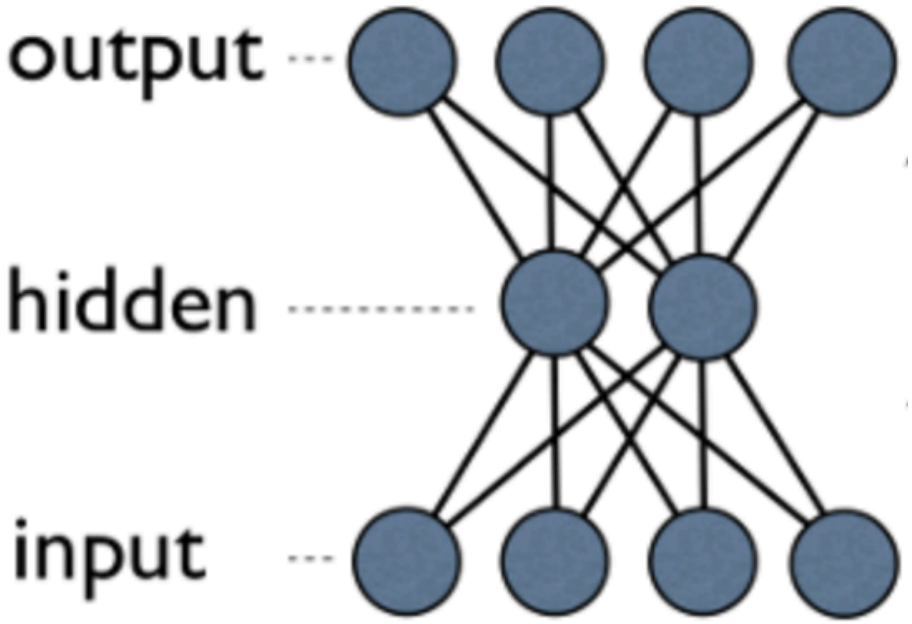


  - Multilayered machines could not be learned. Local optimization.

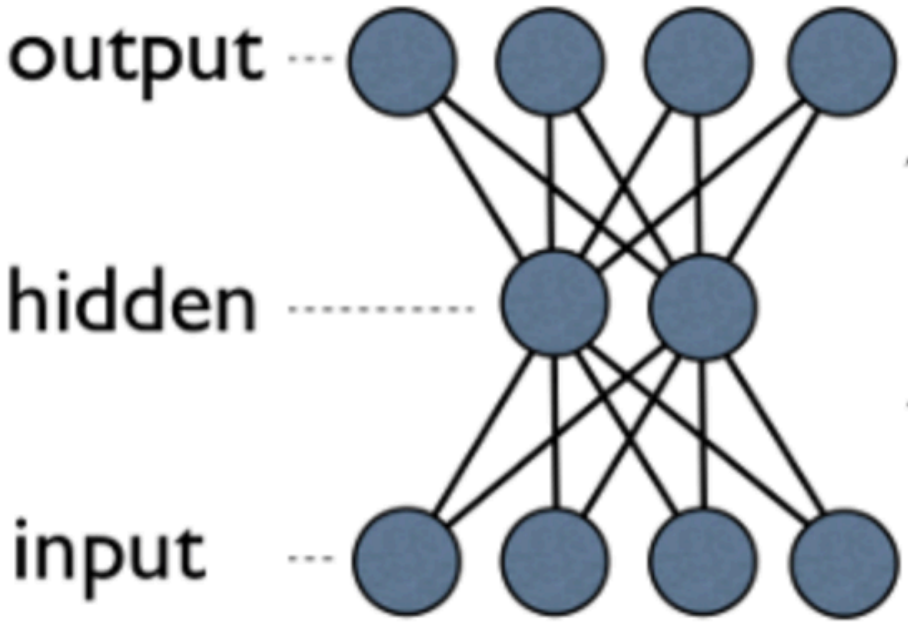
Neural Networks



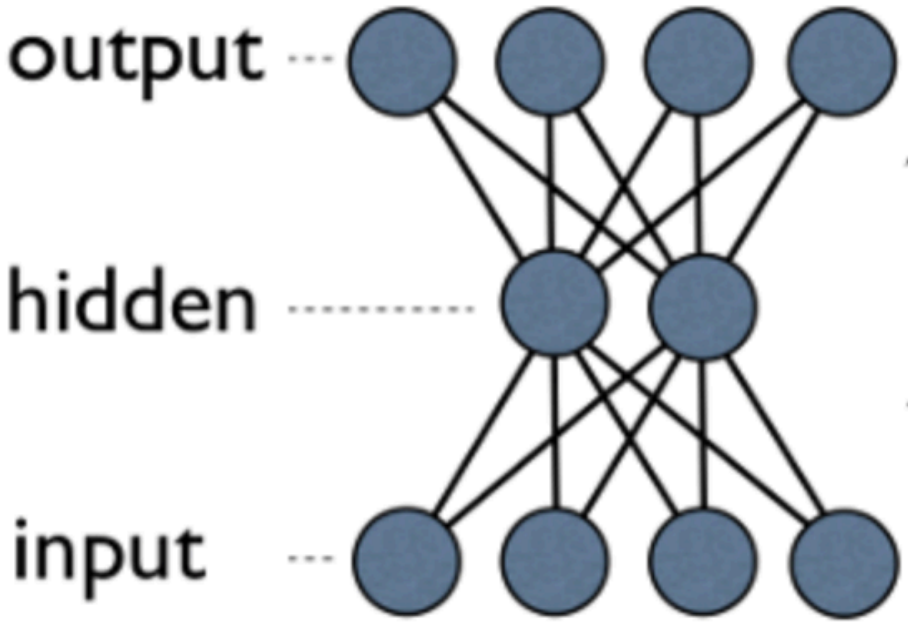
Neural Network Autoencoder



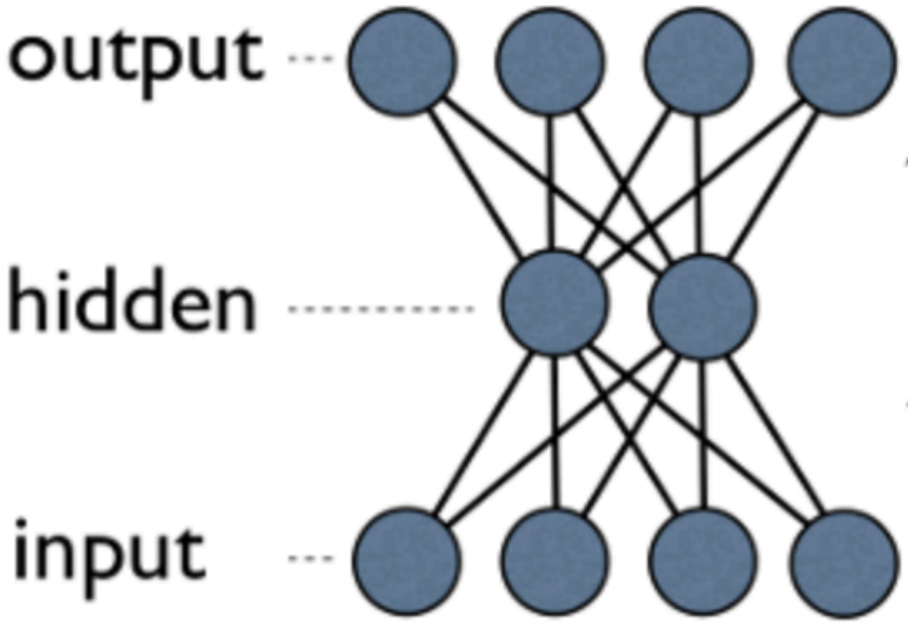
Neural Network Autoencoder



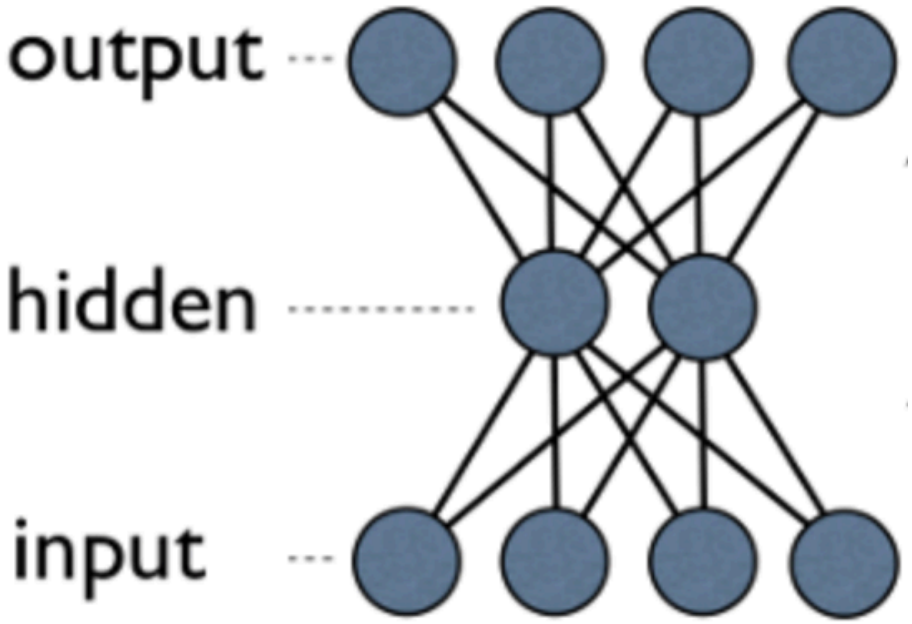
Neural Network Autoencoder



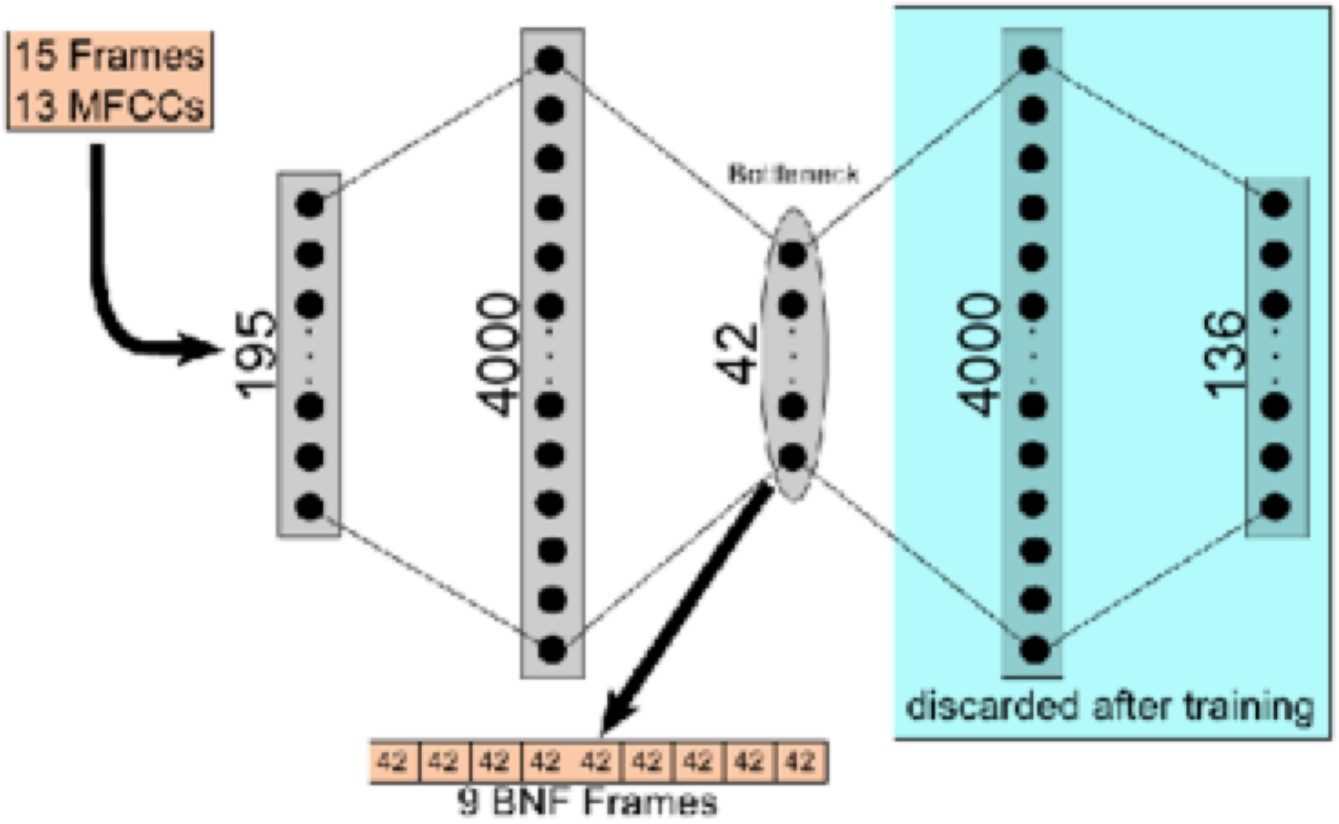
Neural Network Autoencoder



Neural Network Autoencoder



Coding of Speech: “Bottleneck Features”



Using Neural Nets

- Classification
- Prediction
- Function Approximation
- Continuous Mapping
- Pattern Completion
- Coding

Design Criteria

- Recognition Error Rate
- Training Time
- Recognition Time
- Memory Requirements
- Training Complexity
- Ease of Implementation
- Ease of Adaptation

Network Specification

- What parameters are typically chosen by the network designer:
  - Net Topology
  - Node Characteristics
  - Learning Rule
  - Objective Function
  - (Initial) Weights
  - Learning Parameters

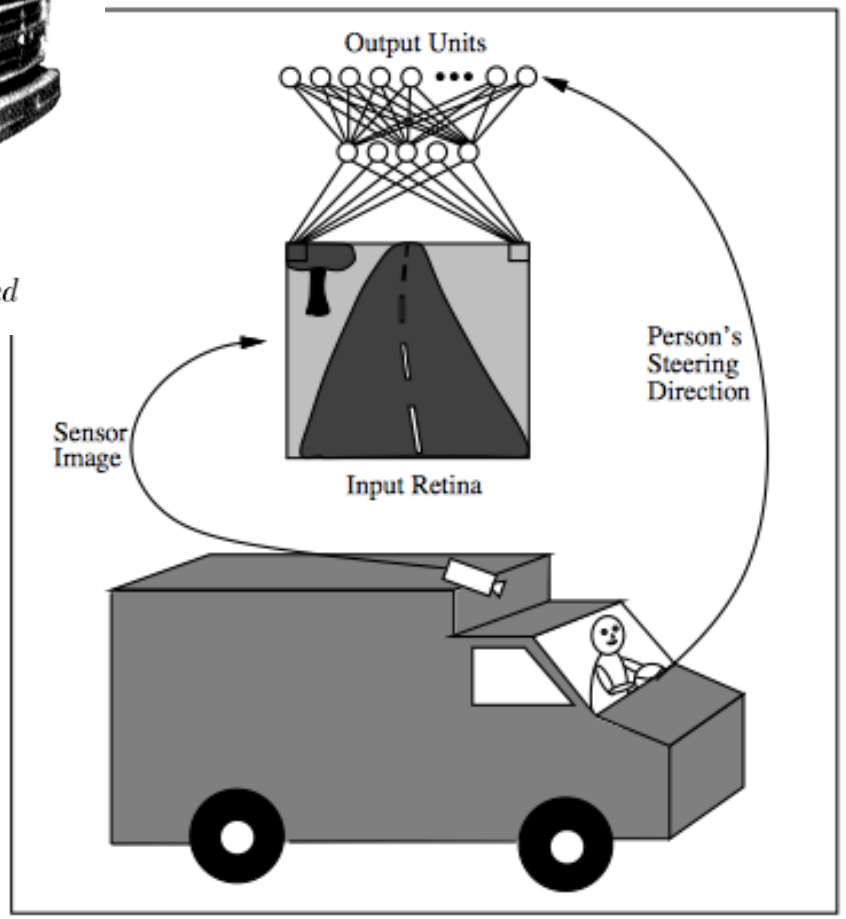
Neural Models

- Back-Propagation
- “Deep” Neural Networks
- Boltzman Machines
- Decision Tree Classifiers
- Feature Map Classifiers
- Learning Vector Quantizer (LVQ, LVQ2)
- High Order Networks
- Radial Basis Functions
- Modified Nearest Neighbor
- ART
- TDNN’s/Convolutional Nets

Applications

- Space Robot*
- Autonomous Navigation*
- Speech Recognition and Understanding*
- Natural Language Processing*
- Music*
- Gesture Recognition
- Lip Reading
- Face Recognition
- Household Robots
- Signal Processing
- Banking, Bond Rating,...
- Sonar
- etc....

Figure 1: *The CMU Navlab Autonomous Navigation Testbed*



How I built a neural network controlled self-driving (RC) car!

Tweet 163

+1 +143 Recommend this on Google

Updated January 11th 2013: Watch my BACON talk on this project on vimeo.

Updated January 2nd 2012: the source code is now open source and available on github.

Recently, I have been refreshing my knowledge of Machine Learning by taking Andrew Ng's excellent Stanford Machine Learning course online. The lecture module on Neural Networks ends with an intriging motivating video of the ALVINN autonomous car driving itself along normal roads at CMU in the mid 90s.

I was inspired by this video to see what I could build myself over the course of a weekend. From a previous project I already had a cheap radio controlled ca which I set about trying to control.

Advanced Neural Models

- Time-Delay Neural Networks (Waibel)
- Recurrent Nets (Elman, Jordan)
- Higher Order Nets
- Modular System Construction
- Adaptive Architectures

Neural Nets - Design Problems

- Local Minima
- Speed of Learning
- Architecture must be selected
- Choice of Feature Representation
- Scaling
- Systems, Modularity
- Treatment of Temporal Features and Sequences

Neural Nets - Modeling

- Neural Nets are
  - Non-Linear Classifiers
  - Approximate Posterior Probabilities
  - Non-Parametric Training
- The Problem with Time, Context, Sequences
  - How to Consider Context
  - How to Operate Shift-Invariantly
  - How to Process Pattern Sequences

Time Varying Patterns

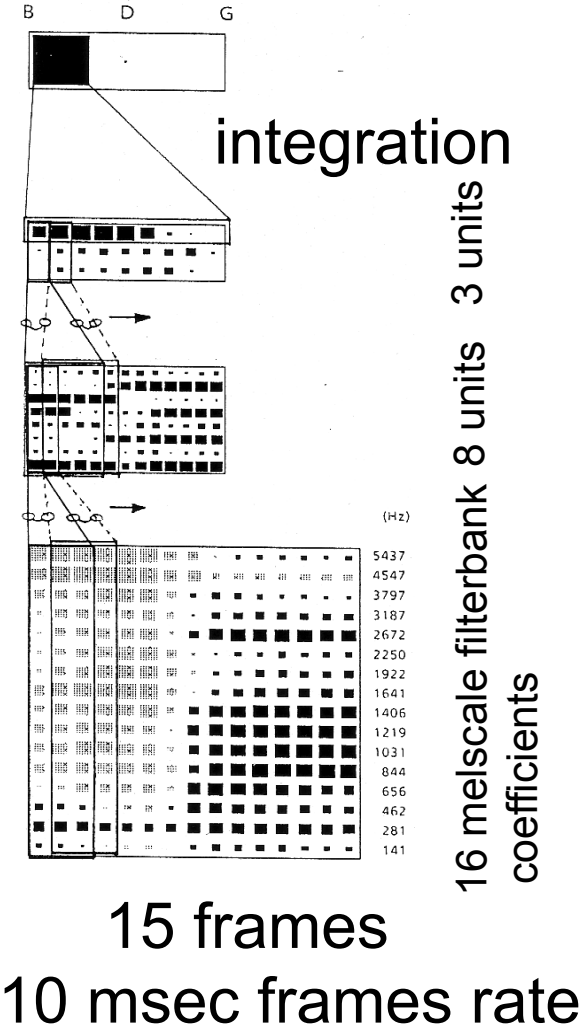
- Detect B in Context A
  - AAABAAA ---> 1
  - AAABCAA ---> 0
- Detect B Independent of Point in Time
  - AAAABAAAAAAA ---> 1
  - AAAAAAAABAAA ---> 1
  - AAAAAAAAAAAA ----> 0
- Detect Sequence ABC
  - AAAAAABBCCCC ---> 1
  - ABBBBBBCCC ---> 1
  - CCAAABB ---> 0

Time-Delay Neural Networks

- Multilayer Neural Network - Nonlinear Classifier
- Time-Delays on Connections
  - At Input Layer
  - At all Layers
- Shift-Invariant Learning
  - All Units Learn to Detect Patterns Independent of Location in Time
  - No Presegmentation or Prealignment Necessary
  - Approach: Weight Sharing

Time-Delay Neural Network (TDNN)

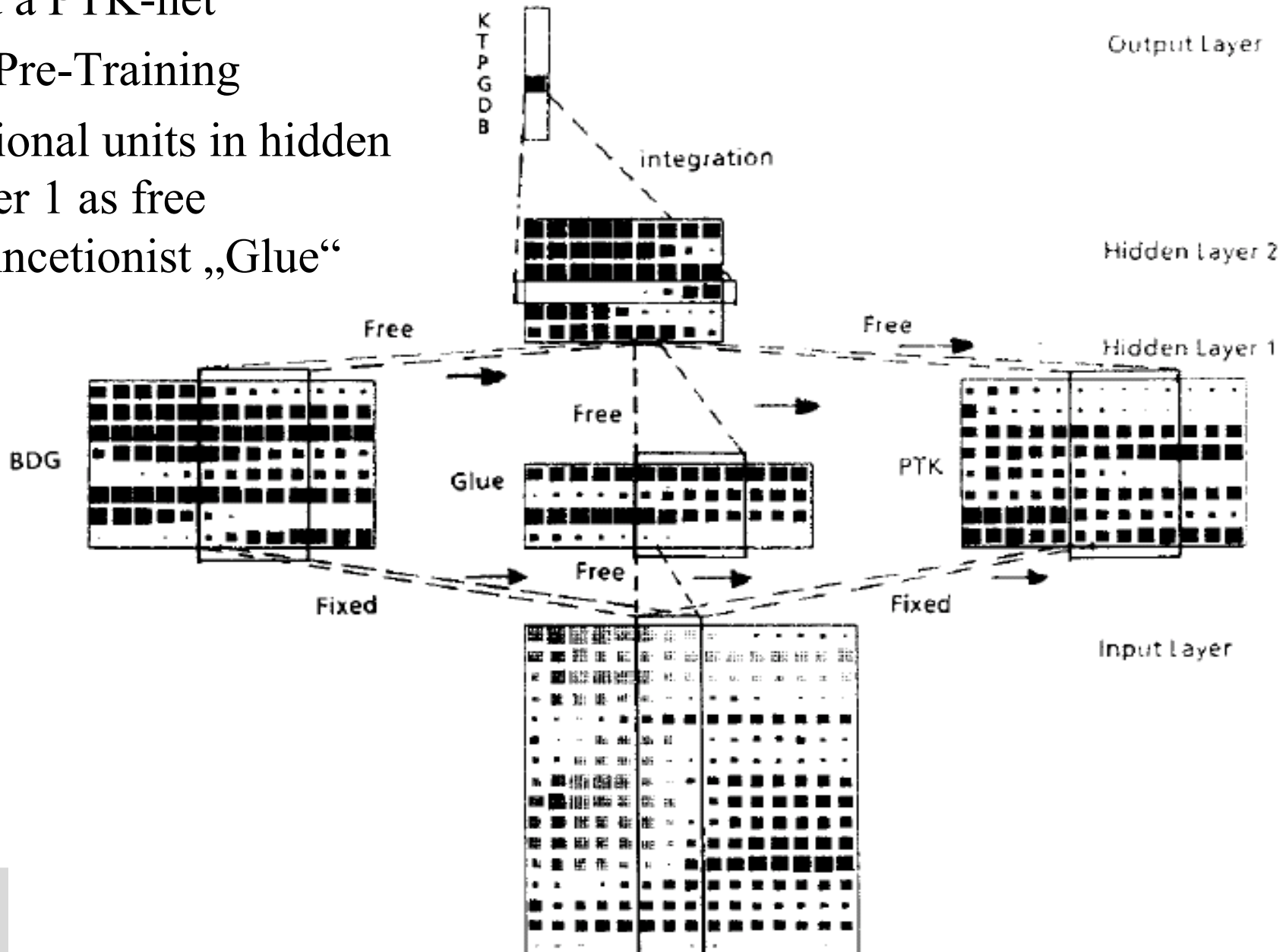
- **TDNN, the first "Convolutional" Neural Net**
  - Waibel' 87
  - Shift Invariance
  - Application to Speech
  - Learns Internal Representations of Speech… Acoustic Phonetic Features
  - Neural Features
  - Noise Suppression
  - Fast Training



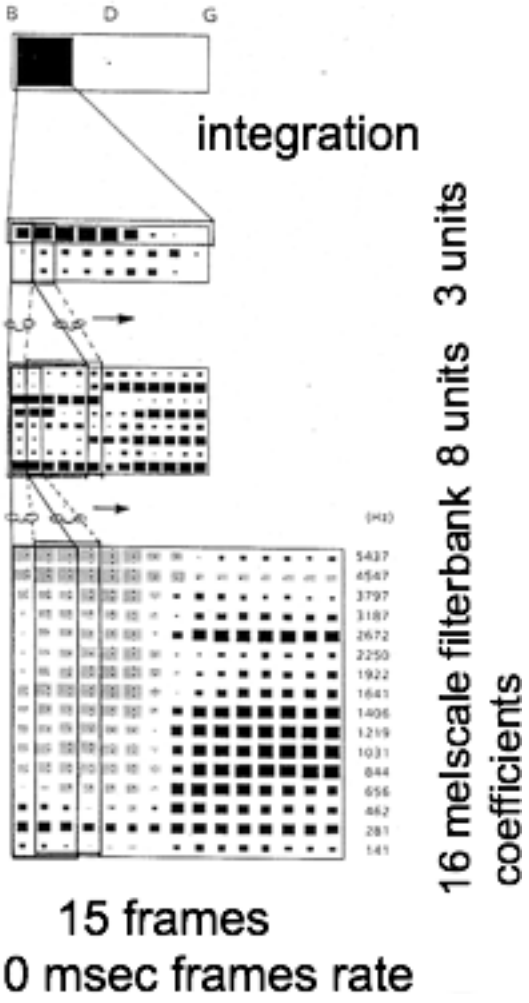
Combination of a BDG-net and a PTK-net

→ Pre-Training

additional units in hidden layer 1 as free conncetionist „Glue“



TDNN / CNN – Waibel 1987



1987

Conferences

A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, *Phoneme Recognition Using T*
Telephony Research Laboratories, October 30, 1987
[BibTeX] [PDF]

A. Waibel, H. Sawai, K. Shikano, *Phoneme Recognition by Modular Constructi*
Telephony Research Laboratories, October 30, 1987
[BibTeX] [PDF]

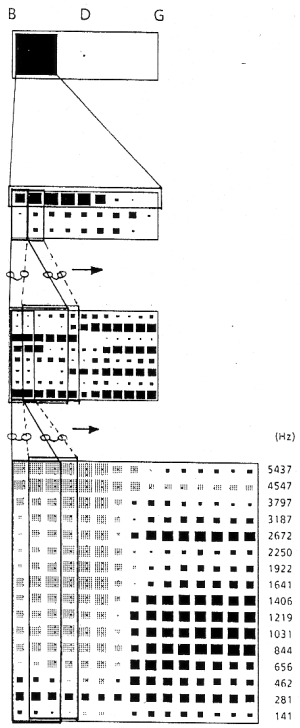
Alex Waibel, *Phoneme Recognition Using Time-Delay Neural Networks*, Meeting
Communication Engineers (IEICE), Tokyo, Japan, December, 1987
[BibTeX] [PDF]

tion [39]. Fixed-size convolutional networks t
weights along a single temporal dimension are
Time-Delay Neural Networks (TDNNs). TDNNs
used in phoneme recognition (without sub-samp
[41], spoken word recognition (with sub-sampl
[43], on-line recognition of isolated handwritte
ters [44], and signature verification [45].

Today's TDNN's/CNN's – Bigger, Deeper, Faster

ologie

(1987) (1989) (>2013)



TDNN: Shift-Invariance, Waibel '87

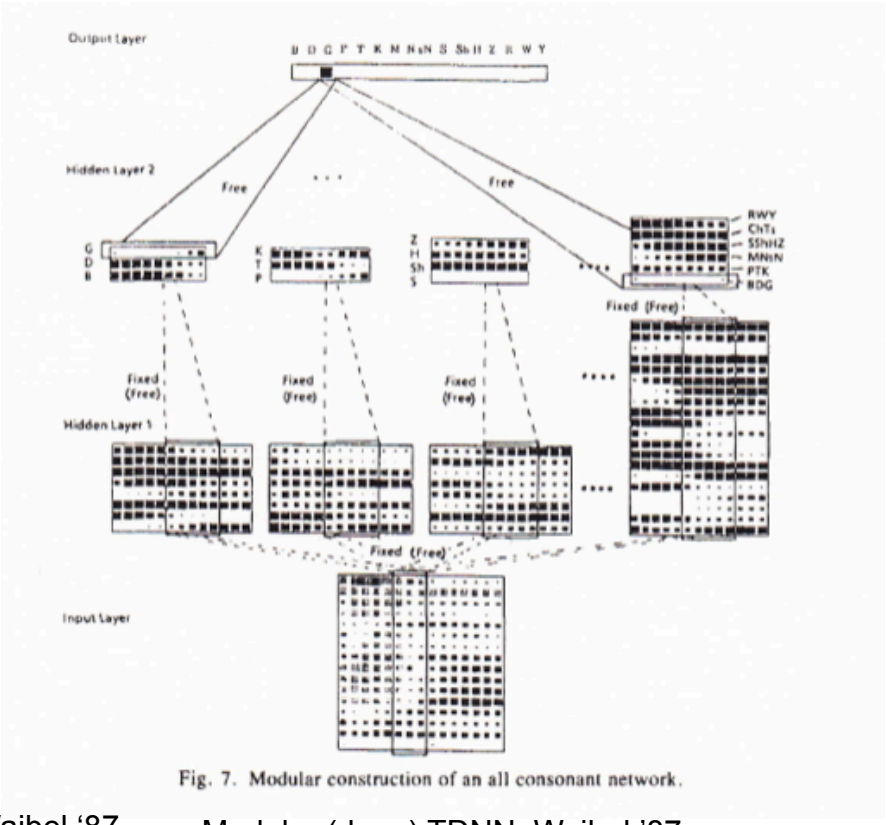
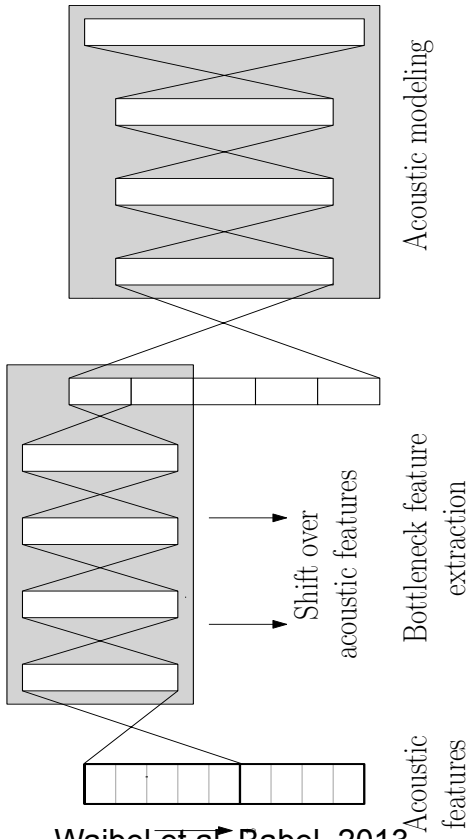


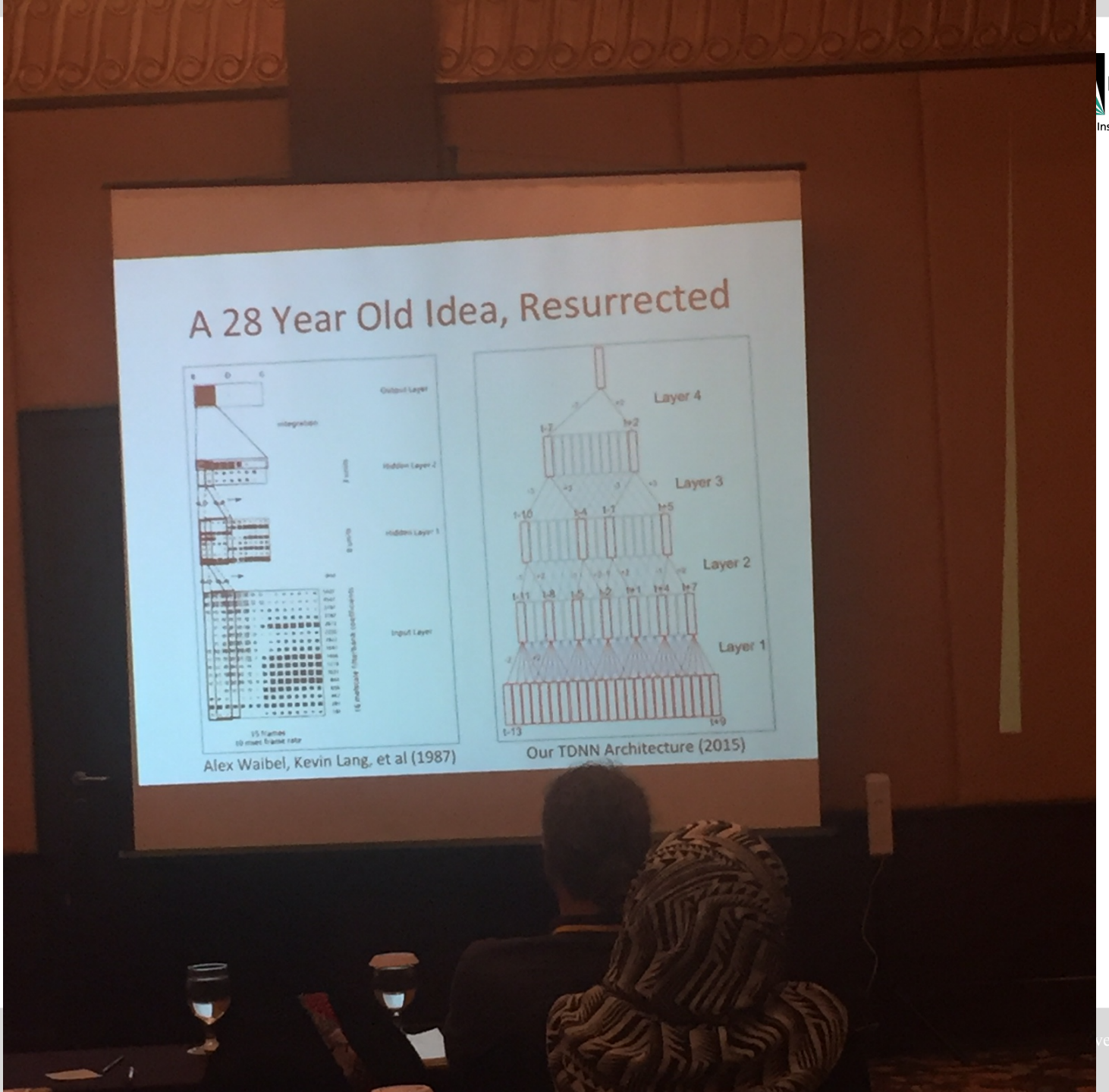
Fig. 7. Modular construction of an all consonant network.

Modular (deep) TDNN: Waibel '87



Waibel et al. Babel, 2013

Weights:	~6,000	~40,0000	~33,000,000
TrnData[hrs]:	~0.1	~1	~1,000
Time[weeks]	~1	~1	~1


A 28 Year Old Idea, Resurrected
Alex Waibel, Kevin Lang, et al (1987)
Layer 4
Layer 3
Layer 2
Layer 1
Our TDNN Architecture (2015)

Reverberation Robust Speech Reco

- Reverberation, a persistent, hard speech recognition problem
- Problem: Removing Echo..: Separate Speech from it's own delayed copies
- Peddinti, Chen, Povey, Khudanpur, *Reverberation Robust Acoustic Modeling Using i-Vectors with Time Delay Neural Networks,* Interspeech 2015.
- Finally, dramatic improvements

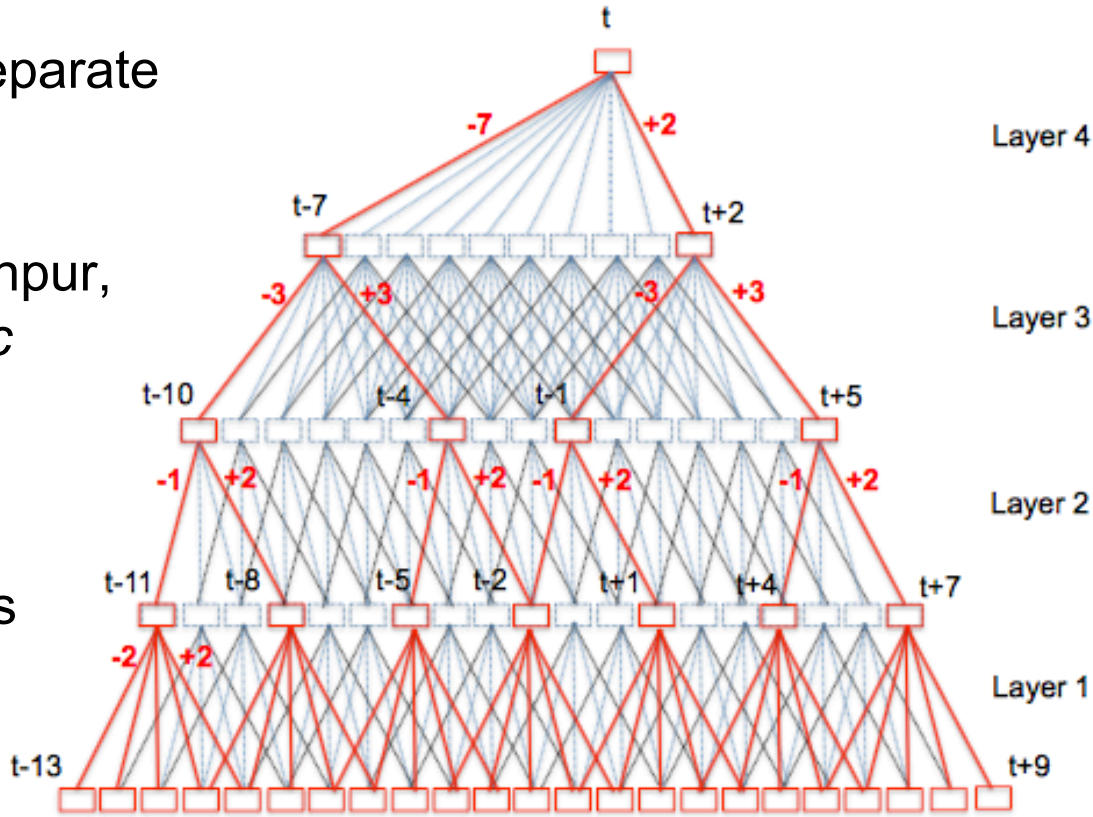
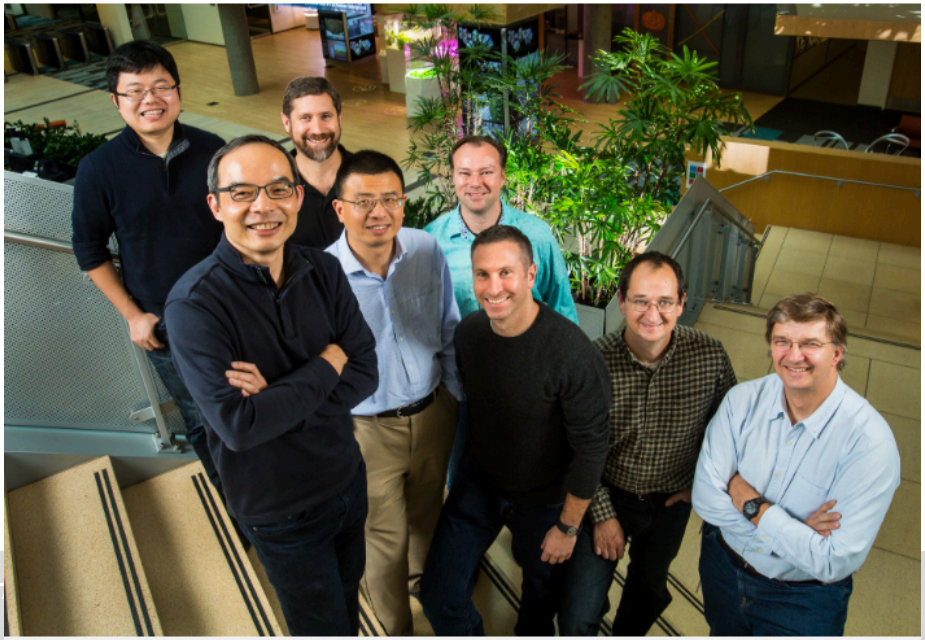


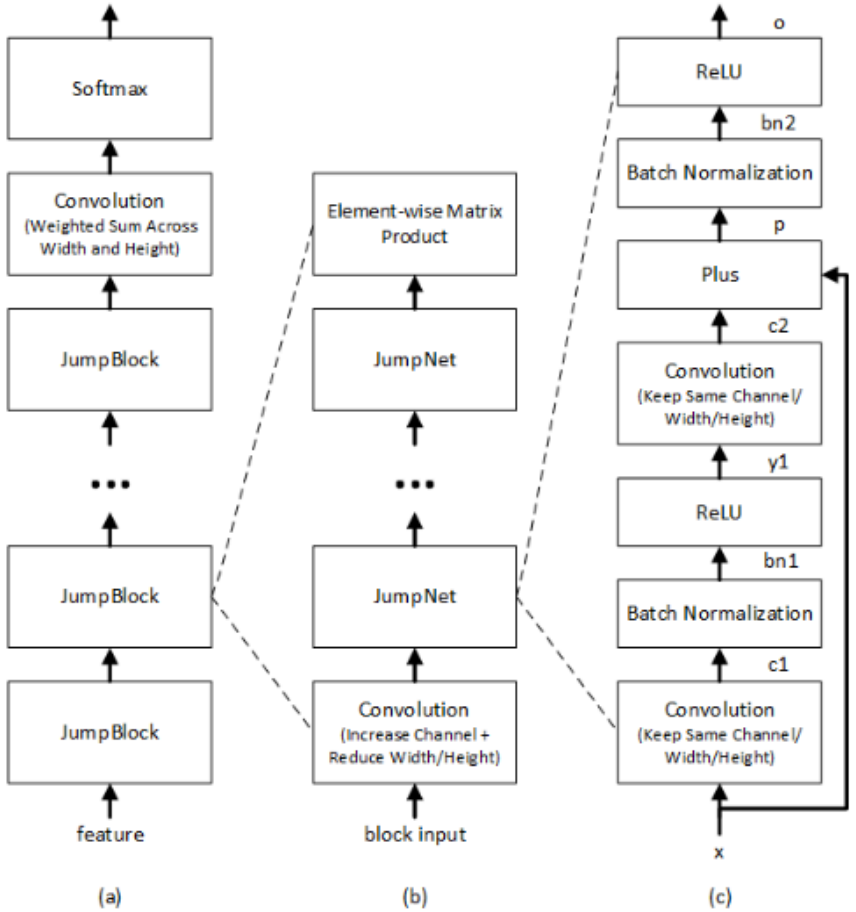
Figure 1: Computation in TDNN with sub-sampling (red) an without sub-sampling (blue+red)

Conversational Speech

- Conversational Speech: Another Hard Problem, High Error Rates Persisted
  for 25 years. Recent Breakthrough uses TDNN/CNN
- Microsoft Achieves Human Parity: Xiong et al., The Microsoft Conversational Speech Recognition System, ICASSP 2017.

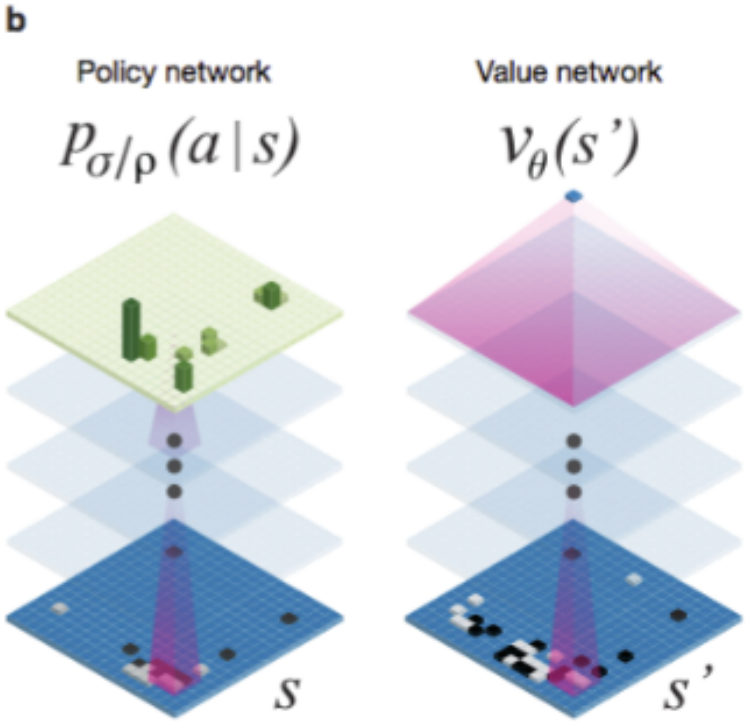
Historic Achievement: Microsoft researchers reach human parity in conversational speech recognition





Mastering the Game of Go

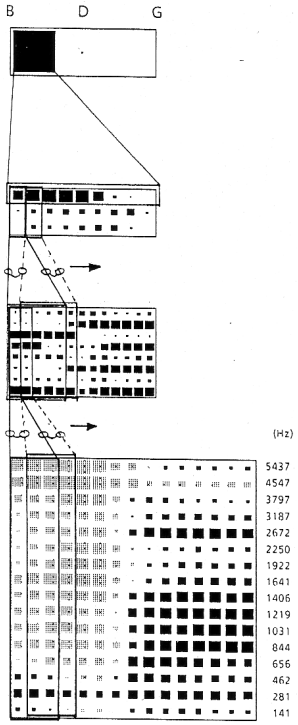
- Alpha Go – Uses Reinforcement Learning and Deep Convolutional Neural Nets
- D. Silver et al., Mastering the Game of Go with Deep Neural Networks and Tree Search, Google Deep Mind, 2016



The beginning of the end: Google's AI has beaten a top human player at the complex game of Go

Neural Nets: Bigger, Deeper, Faster

(1987) (1989) (2013)



TDNN: Shift-Invariance, Waibel '87

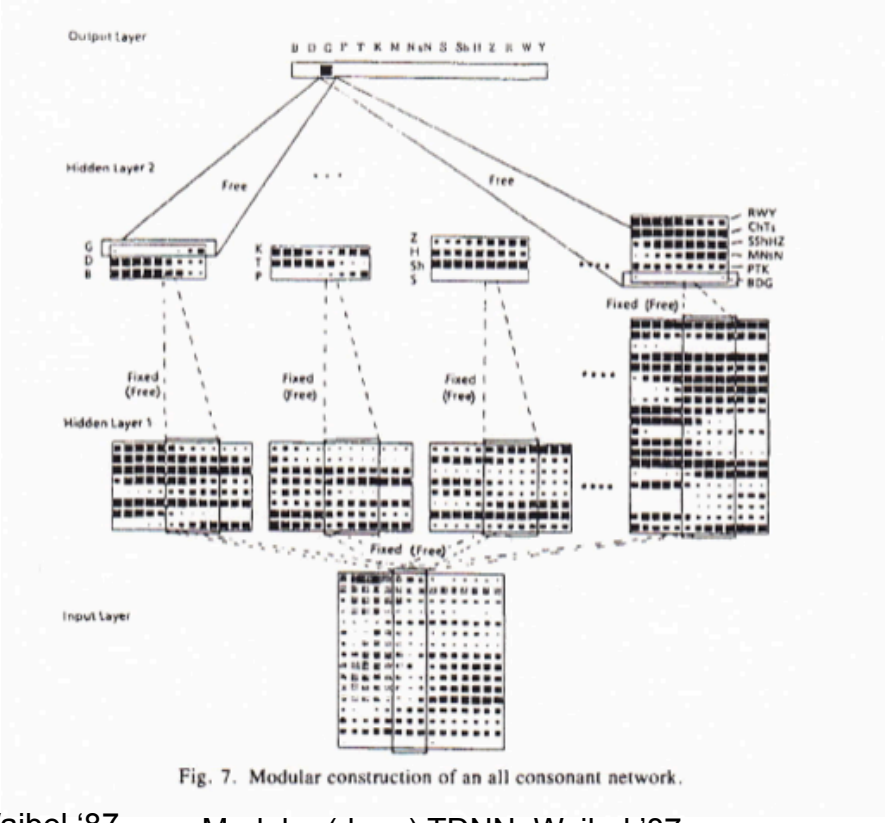
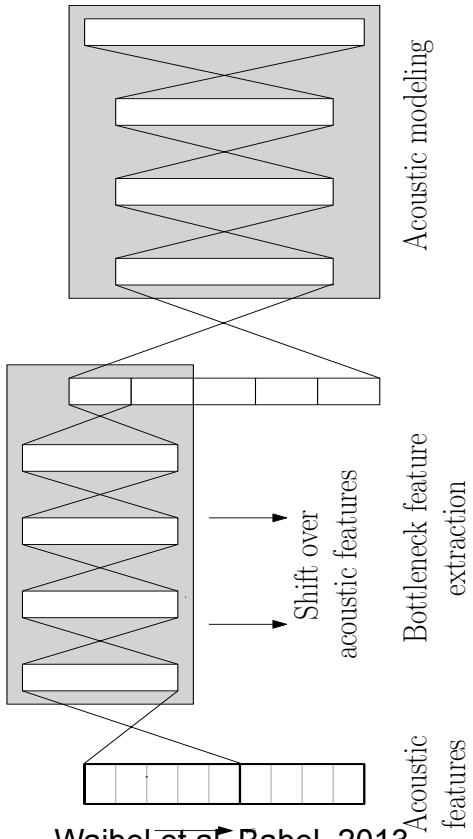


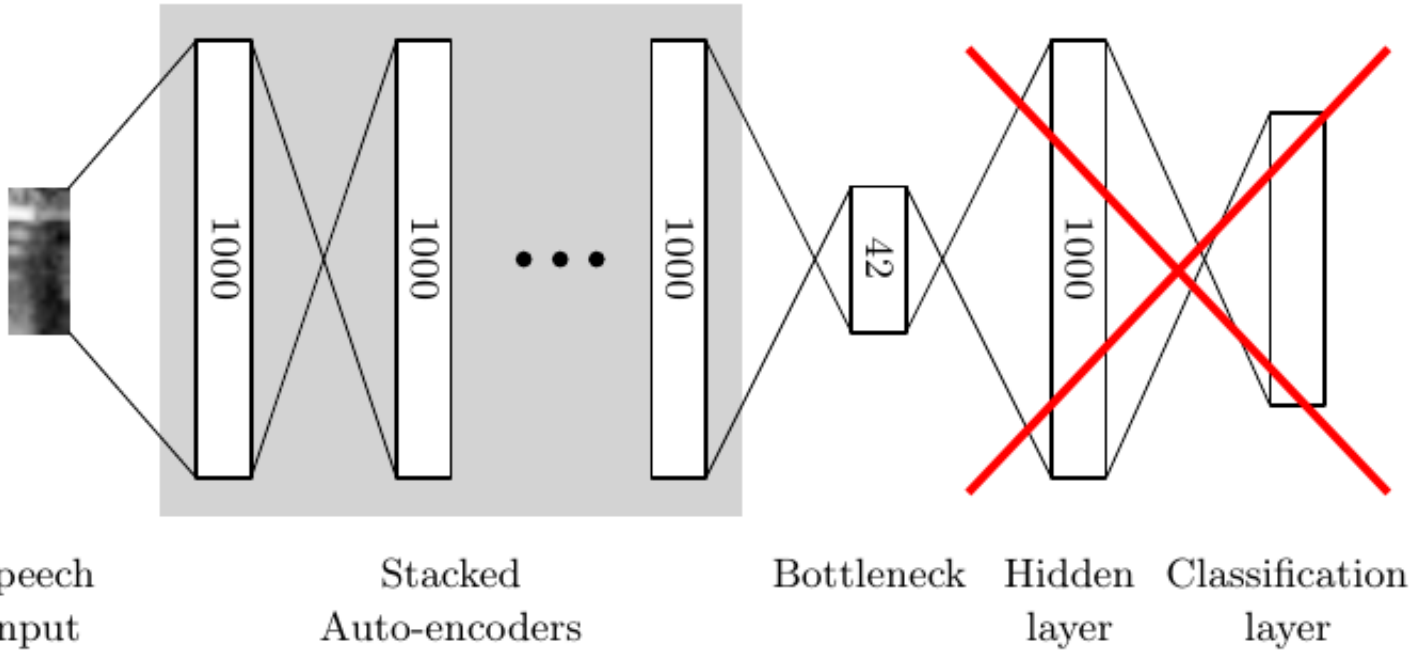
Fig. 7. Modular construction of an all consonant network.

Modular (deep) TDNN: Waibel '87

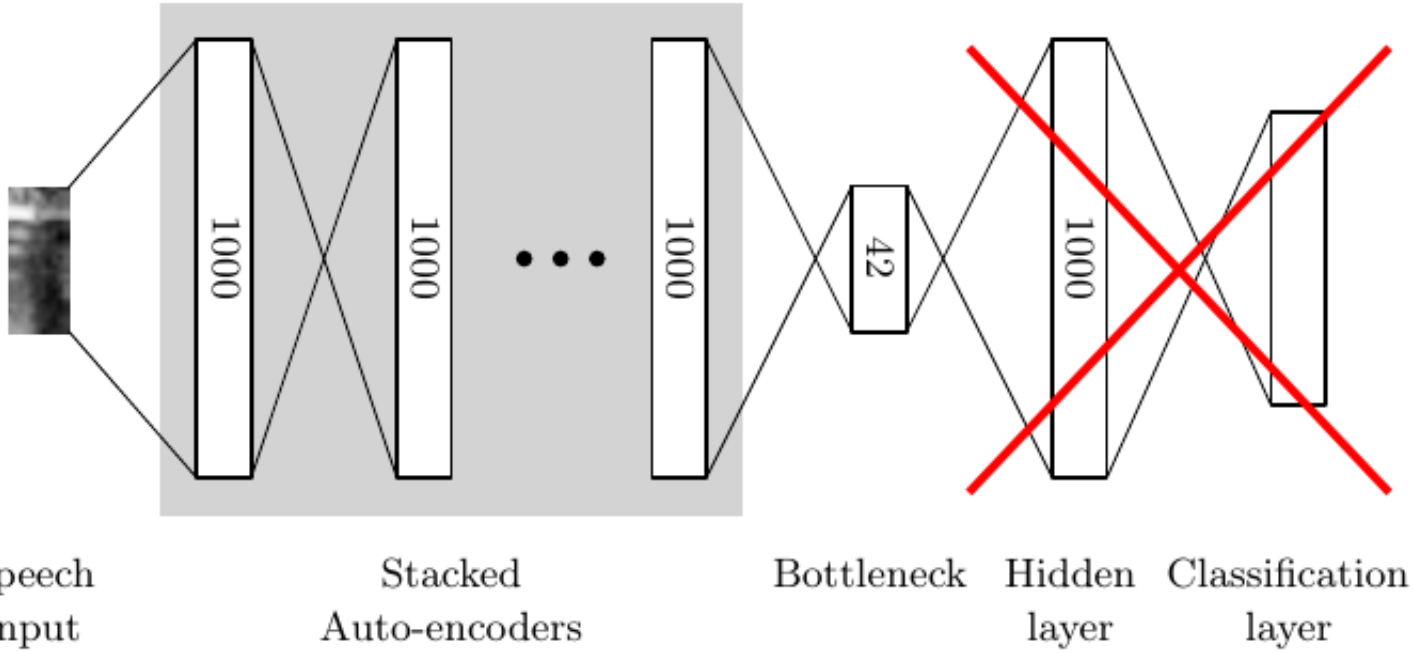


Waibel et al. Babel, 2013

Deep Bottleneck Features



Deep Bottleneck Features 2012



lTraining Prarmeters: 360 * 1000 + 1000 + 4*(1000*1000+1000)+ 1000 * 42 + 42 + 42* 1000 + 1000 +1000 * 100000 = 14450042

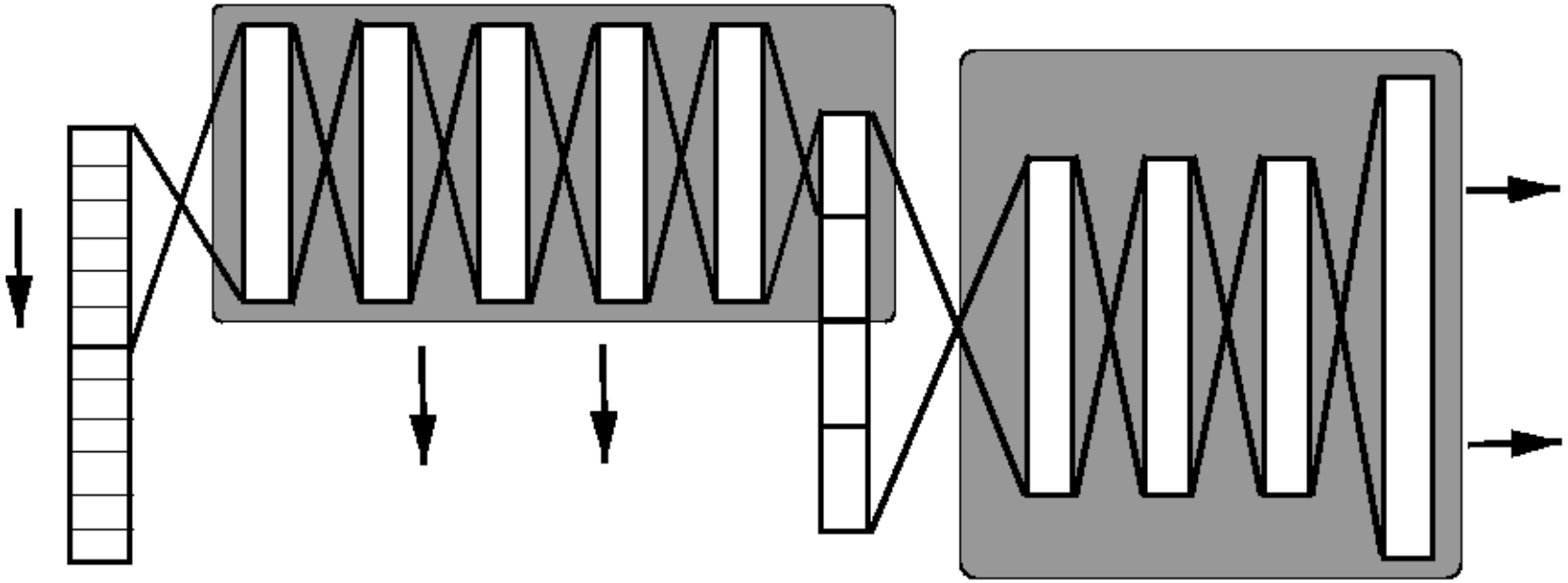
lDecoding Prarmeters: 360 * 1000 + 1000 + 4*(1000*1000+1000)+ 1000 * 42 + 42 = 4407042

lPretraining: ~40 hours on a M2075 GPU (>> 400 h on a CPU)

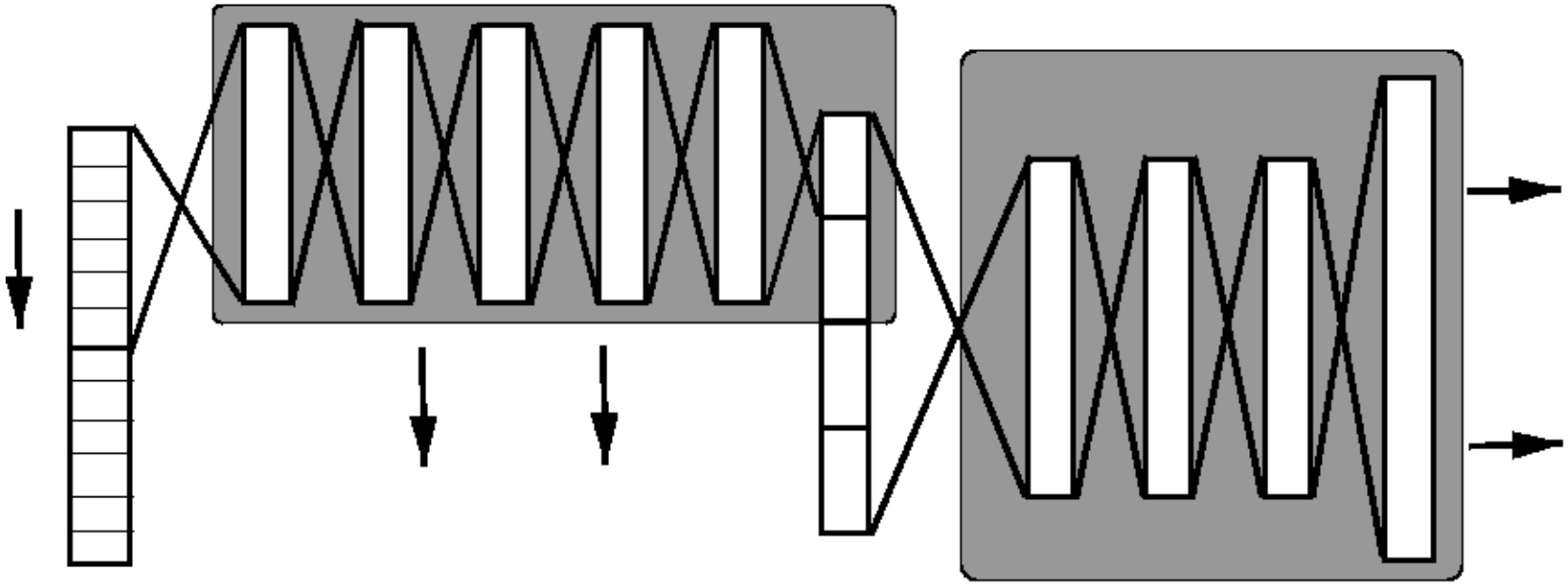
lTraining Time: ~80 hours on a M2075 GPU (>> 800 h on a CPU)

lWER: 16.6%

Hybrid System



Hybrid System



lTraining Prarmeters: 54*13 * 1600 + 1600 + 4* (1600*1600+1600) + 1600 * 42 + 42 + 42*13*1600+1600 + 2* (1600*1600+1600) + 1600*10000+10000 = 33446842

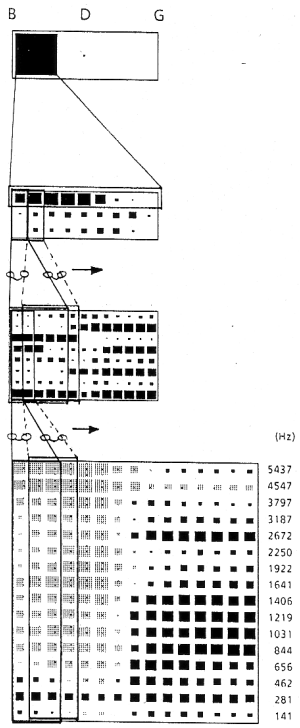
lPretraining: ~60 hours on a K20 GPU (~100 h on a M2075 GPU)

lTraining Time: ~50 hours + ~90 hours on a K20 GPU

lWER: 14.32%

Neural Nets: Bigger, Deeper, Faster

(1987) (1989) (2013)



TDNN: Shift-Invariance, Waibel '87

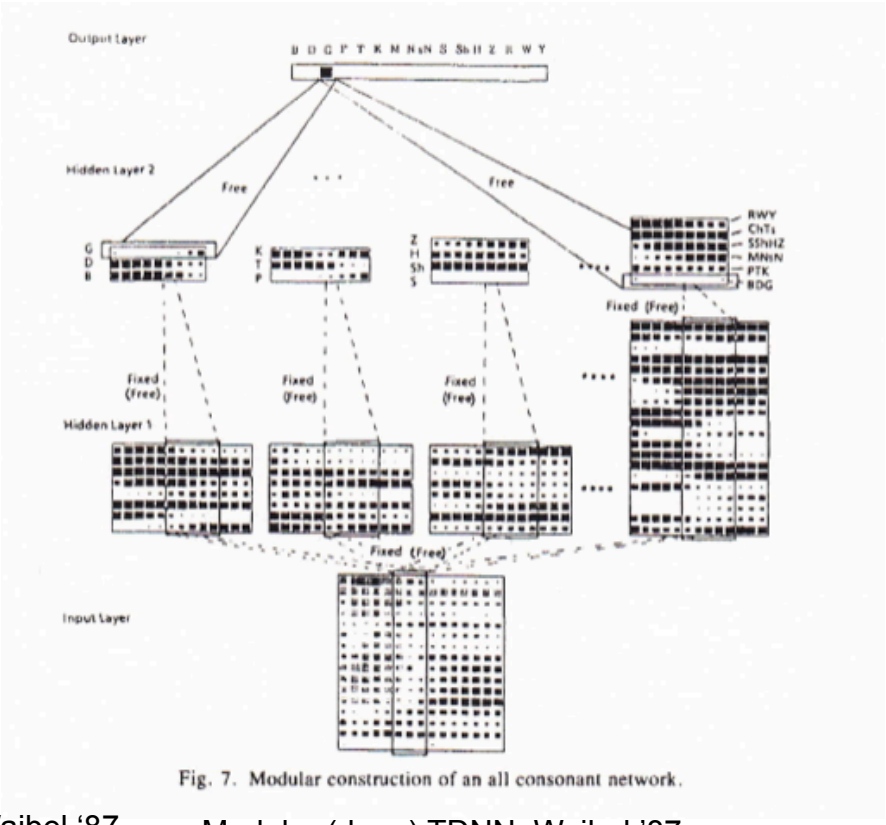
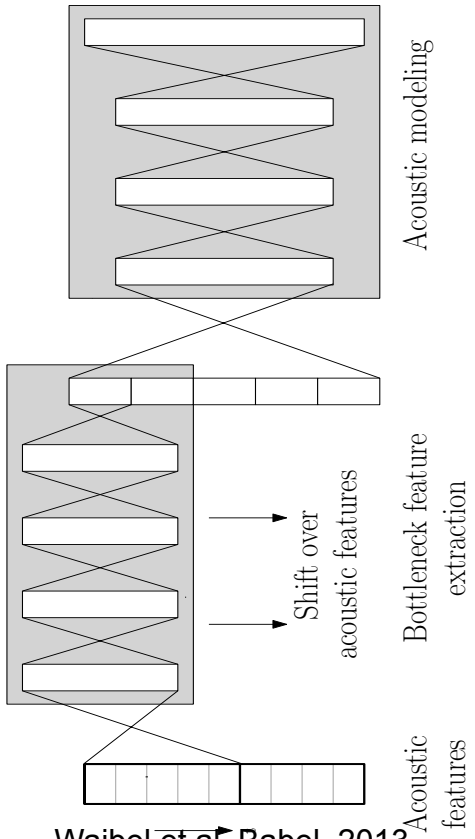


Fig. 7. Modular construction of an all consonant network.

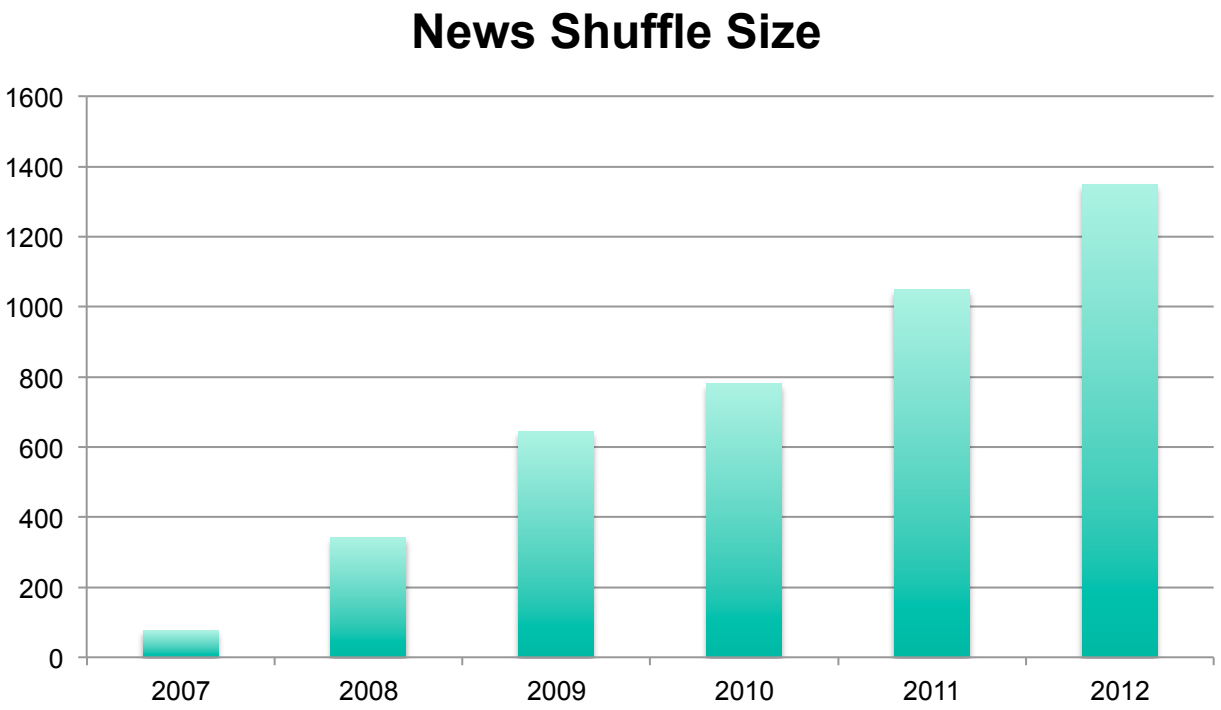
Modular (deep) TDNN: Waibel '87



Waibel et al. Babel, 2013

Weights:	~6,000	~40,0000	~33,000,000
TrnData[hrs]:	~0.1	~1	~1,000
Time[weeks]	~1	~1	~1

English Text Copora



- Computer MT or ASR systems train on >> 1GWords
  - News Shuffle, GigaWord, Europarl, VideoLectures, …
- Human speaks 0.5 GigaWords in a Lifetime!!

- 2009: Google processed about 24 petabytes of data per day
  - Portal.acm.org. Retrieved 2009-08-16
- 2010: Wikipedia's raw data uses a 6 terabyte dump
  - Meta.wikimedia.org. Retrieved 2013-04-14
- 2012: The Internet Archive contains about 10 petabytes of data in cultural material
  - Collections Team blog. Internet Archive. October 26, 2012. Retrieved 2012-10-27
- 2012: Facebook processes 2.5 billion pieces of content and 500+ terabytes of data each day
  - http://techcrunch.com/2012/08/22/how-big-is-facebooks-data-2-5-billion-pieces-of-content-and-500-terabytes-ingested-every-day/, Retrieved 2014-05-28
- 2012: Twitter users write 500 million tweets per day
  - http://www.telegraph.co.uk/technology/twitter/9945505/Twitter-in-numbers.html, Retrieved 2014-05-28
- 2014: YouTube users upload 100+ hours of new video every minu
  - http://www.youtube.com/yt/press/statistics.html, Retrieved 2014-05-28

The Facebook Graph

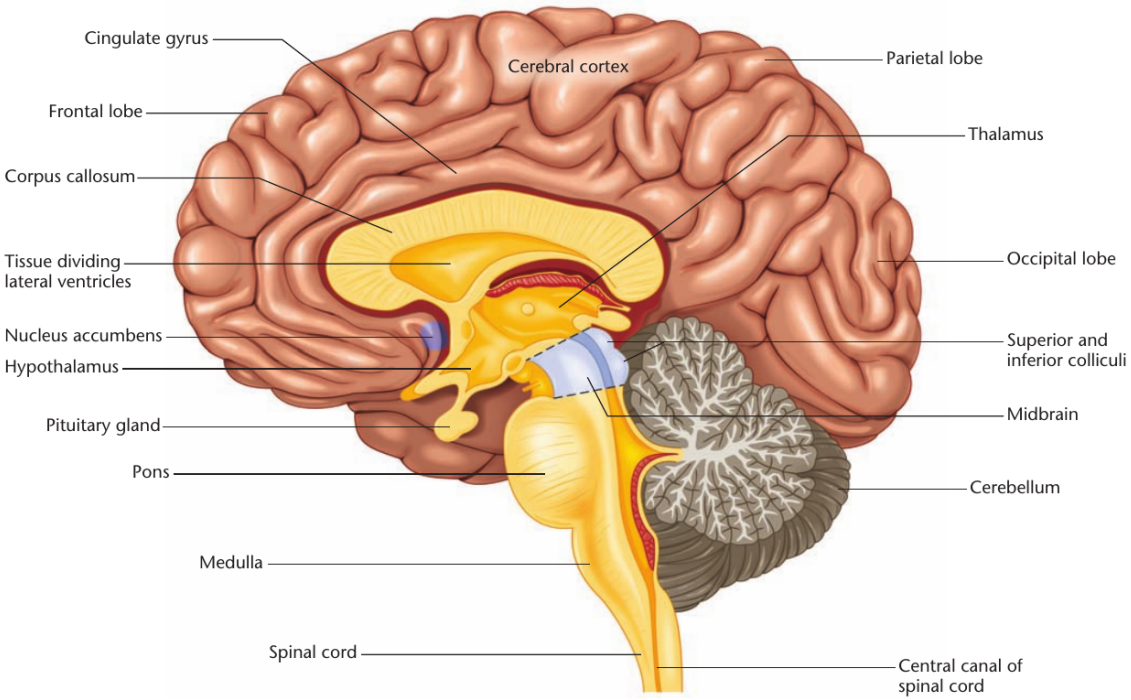

Facebook Datacenter

Facebook Datacenter

InterACT Rechnerausstattung

- GPUs
  - Karlsruhe: 8 x M2075 -> 8 Tflops
    6 x K20m -> 21 Tflops
  - CMU PGH: 12 x K20m -> 42 Tflops
  - CMU SV: >60 Tflops
- CPU Cluster:
  - CMU-Pittsburgh, 500 Cores: 17 TFlops.
  - Karlsruhe, 500+ Cores: >17TFlops

Human Brain



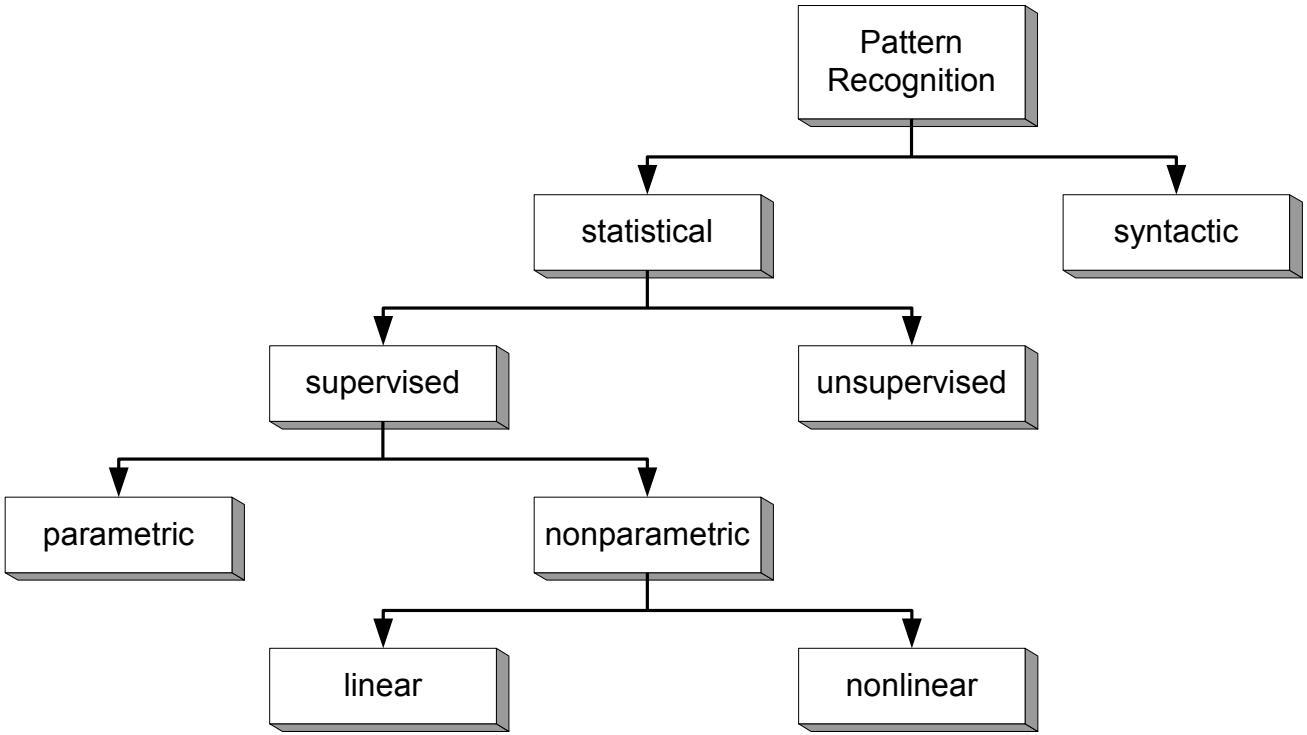
- ~ 100 billion neurons
- ~ 1000 connections each
- ~ can fire up to 10-1000 times a second
- small (1400cm^3)
- low powered (20W)
- fast reaction

- push a button on audio (160ms) or visual (190ms) signal
- reading a word / identifying a simple image (300-700ms)
- 71000 neuron and 71 million connections per mm^3 of brain tissue
- Our biggest net at 33 million weights represents less than ½ mm^3 of brain tissue or 0.04% of the whole brain
- WER: 1-4%

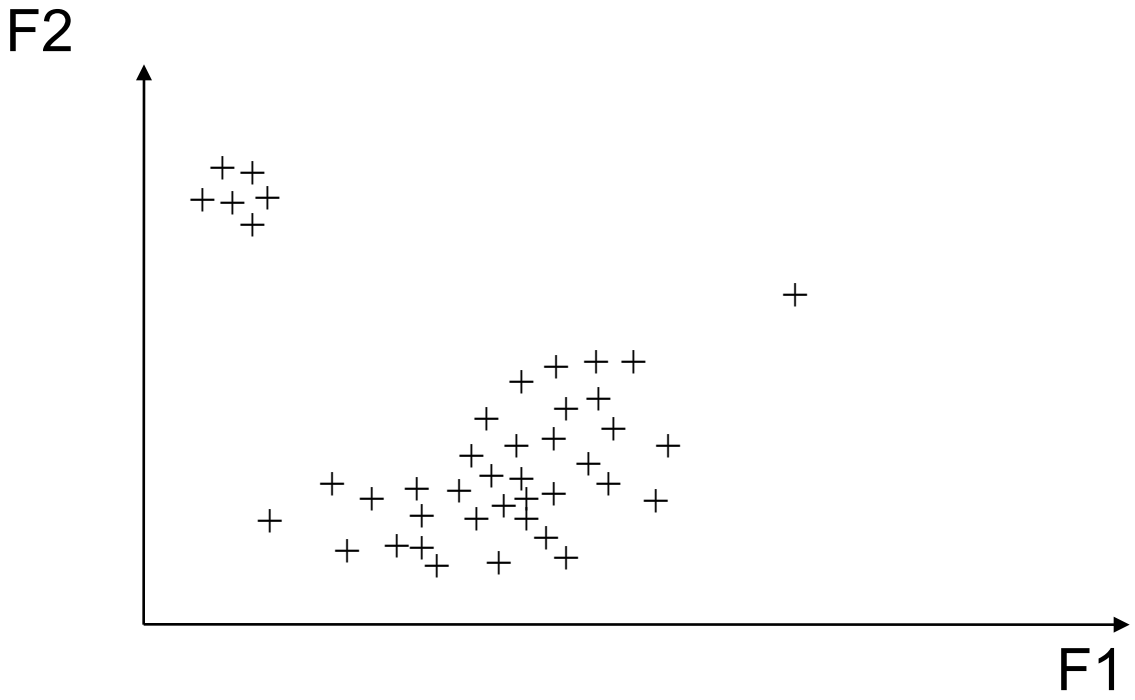
Unsupervised Learning

Alex Waibel

Pattern Recognition

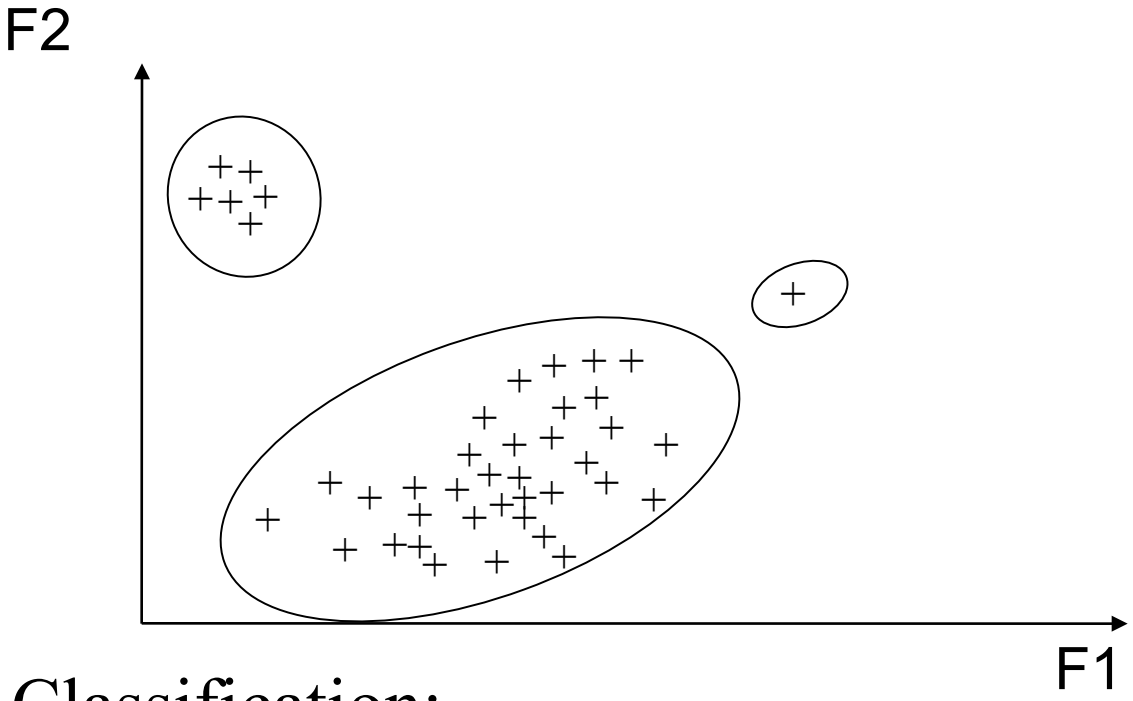


Unsupervised Classification



- Classification:
  - Classes Not Known: Find Structure

Unsupervised Classification



- Classification:
  - Classes Not Known: Find Structure
  - Clustering
  - How? How many?

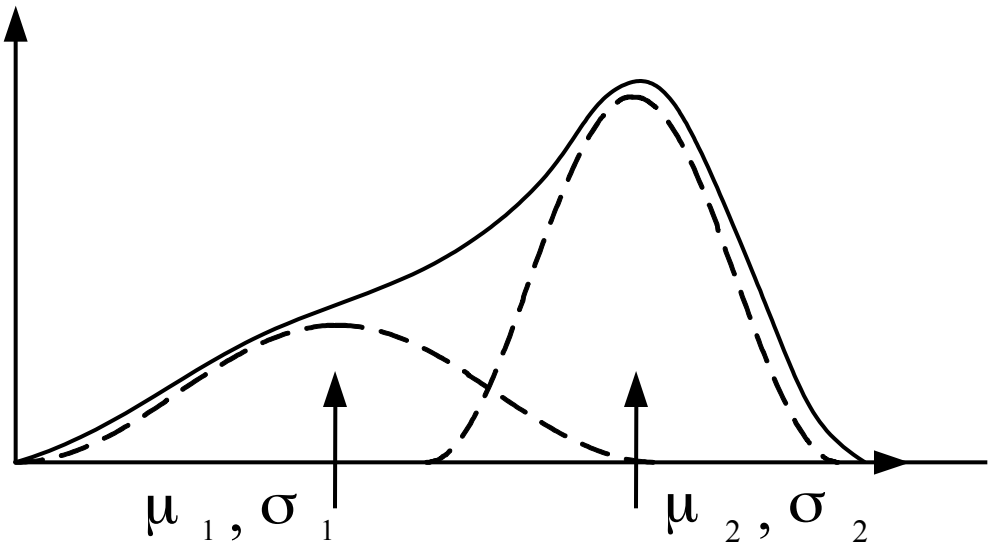
Unsupervised Learning

- Data collection and labeling costly and time consuming
- Characteristics of patterns can change over time
- May not have insight into nature or structure of data

→ **Classes NOT known**

Mixture Densities

- Samples come from c classes
- A priori probability $P(\omega_j)$
- Assume: The forms for the class-conditional PDF's $P(X / \omega_j, \Theta_j)$ are known (usually normal)
- Unknown parameter vector $\Theta_1 \ldots \Theta_c$



Mixture Densities

Problem: Hairy Math

Simplification/ Approximation:

Look only for means -> Isodata

1. Choose initial $\mu_1 \ldots \mu_c$
2. Classify n samples to closest mean
3. Recompute means from samples in class
4. Means changed? Goto step 2, else stop

Mixture Densities

Isodata, problems:

- Choosing initial means μ
- Knowing number of classes
- Assuming distribution
- What is "closest"?

Clustering

- Similarity
- Criterion function
- Samples in same class should extremize criterion function, that measures cluster quality
- Example: sum of error criterion

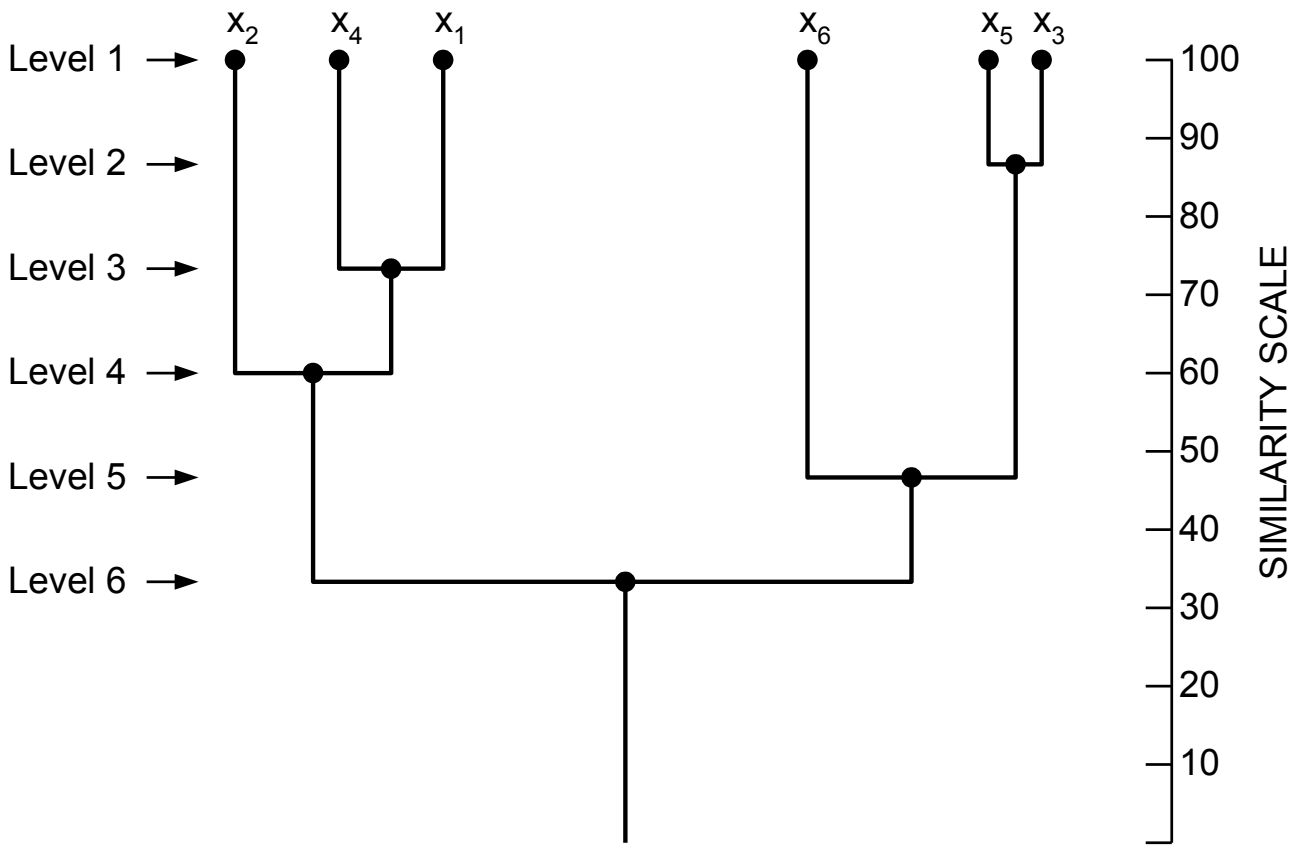
$$J = \sum_{i=1}^{c} \sum_{max} \|x - m_i\|^2$$

Hierarchical Clustering

- Need not determine c
- Need not guess initial means

1. Initialize c := n
2. Find nearest pair of distinct clusters χ_i and χ_j
3. Merge them and decrement c
4. If $c \leq C_{stop}$ stop, otherwise goto step 2

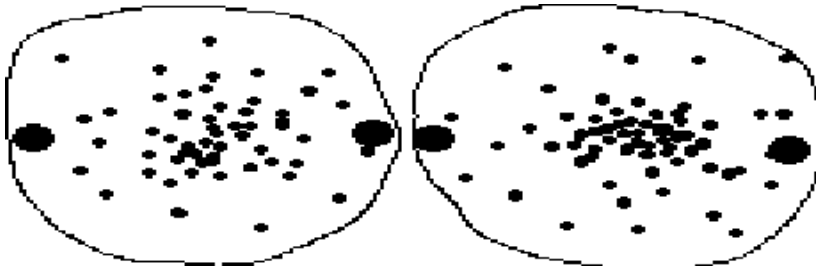
Dendrogram for hierarchical clustering



Similarity

What constitutes "nearest cluster"?

- D_{min}
- D_{ave}
- D_{max}

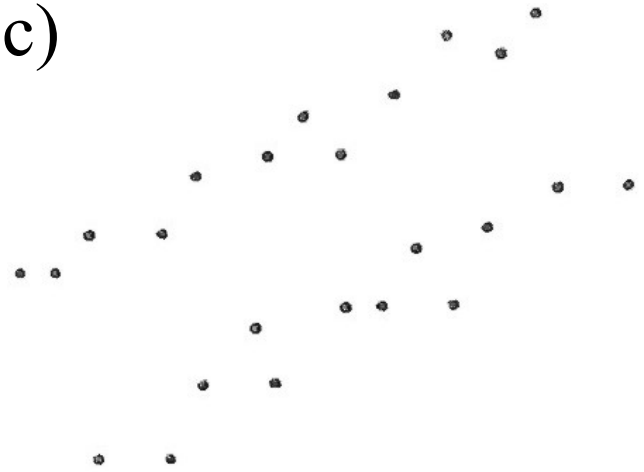


Three illustrative examples

a)

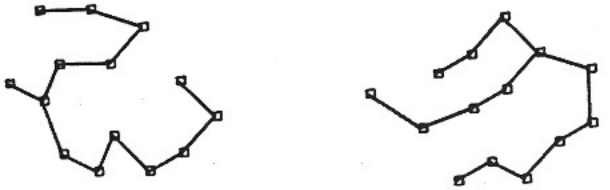
b)

c)



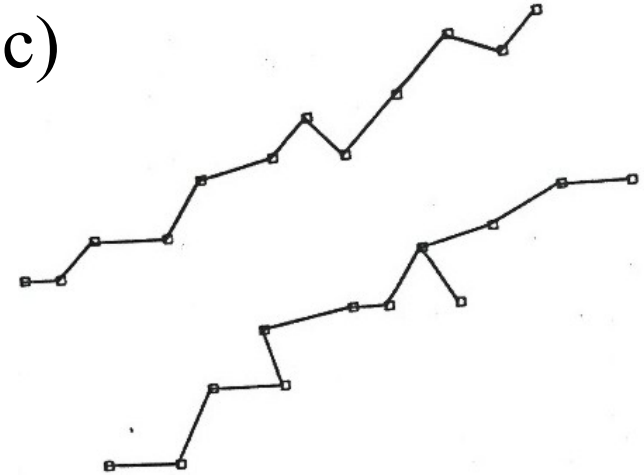
Results of the nearest-neighbor algorithm

a)



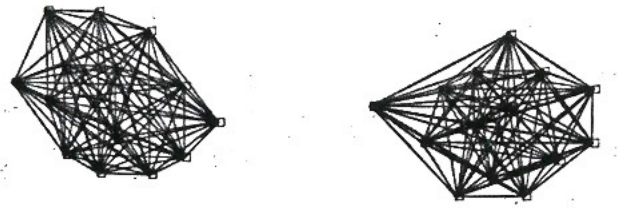
b)

c)

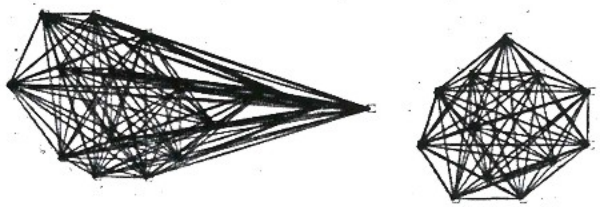


Results of the furthest-neighbor algorithm

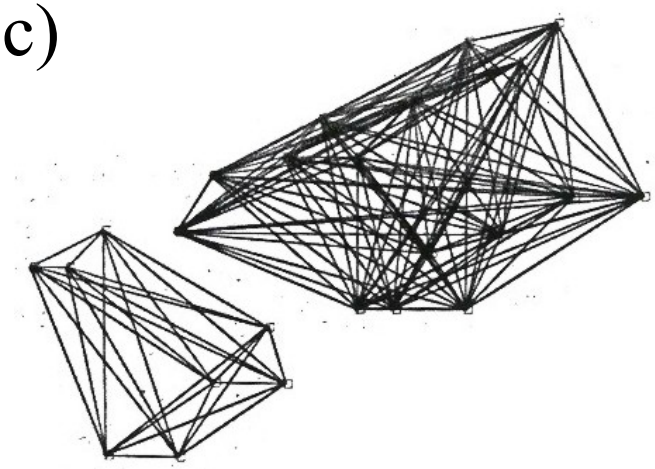
a)



b)



c)



Facebook - Europe

Rechneraustattung

- Rechencluster mit:
  33 Knoten, 1096 Kernen, insgesamt 6,8 TB RAM
- Davon 12 Knoten mit 512 GB RAM
- 8 M2095 GPUs
- 6 K20 GPUs
- 150 TB Plattenplatz