Data Mining

Lecture 7

Associative Networks and Recurrent Classification

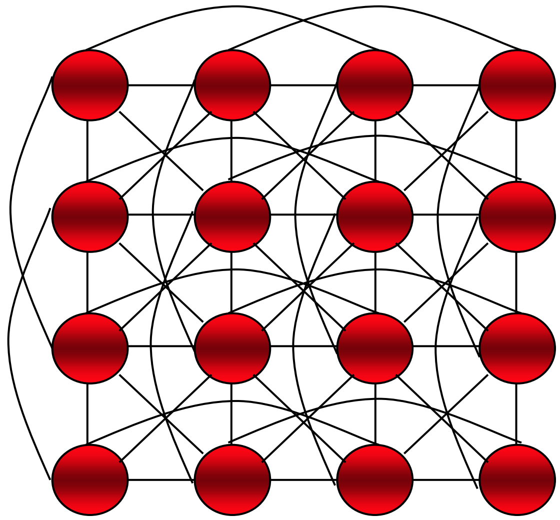
http://www.informatik.uni-hamburg.de/WTM/

Associative Memory: simple Form of Neural Learning

- We link patterns in our brains
- We generalise from patterns we have seen
  - **Example**: recognize letters in different fonts
- This is ***Context-Addressable*** or ***Associative*** Memory

The Hopfield Network as Associative Memory

- An example of a network of McCulloch and Pitts networks
  - Perceptrons
  - Binary Threshold Units

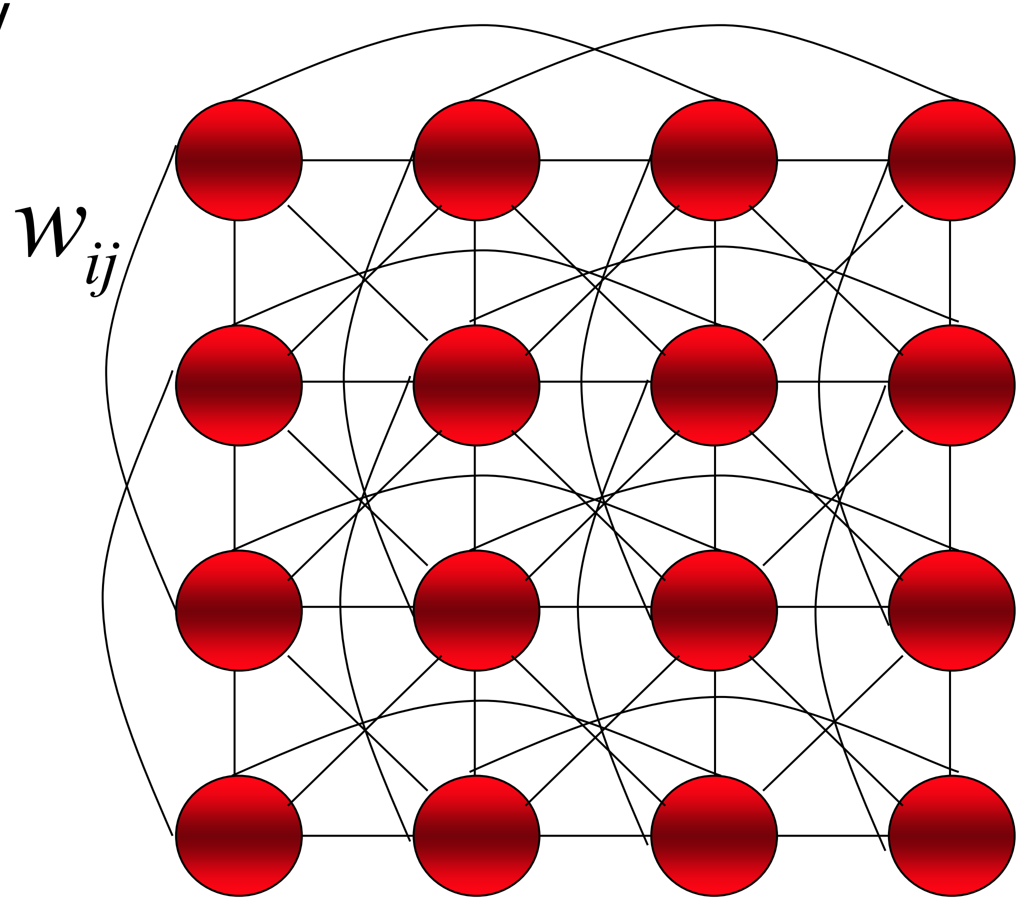


- Symmetric weights
- No self connection
- Invented by J. Hopfield

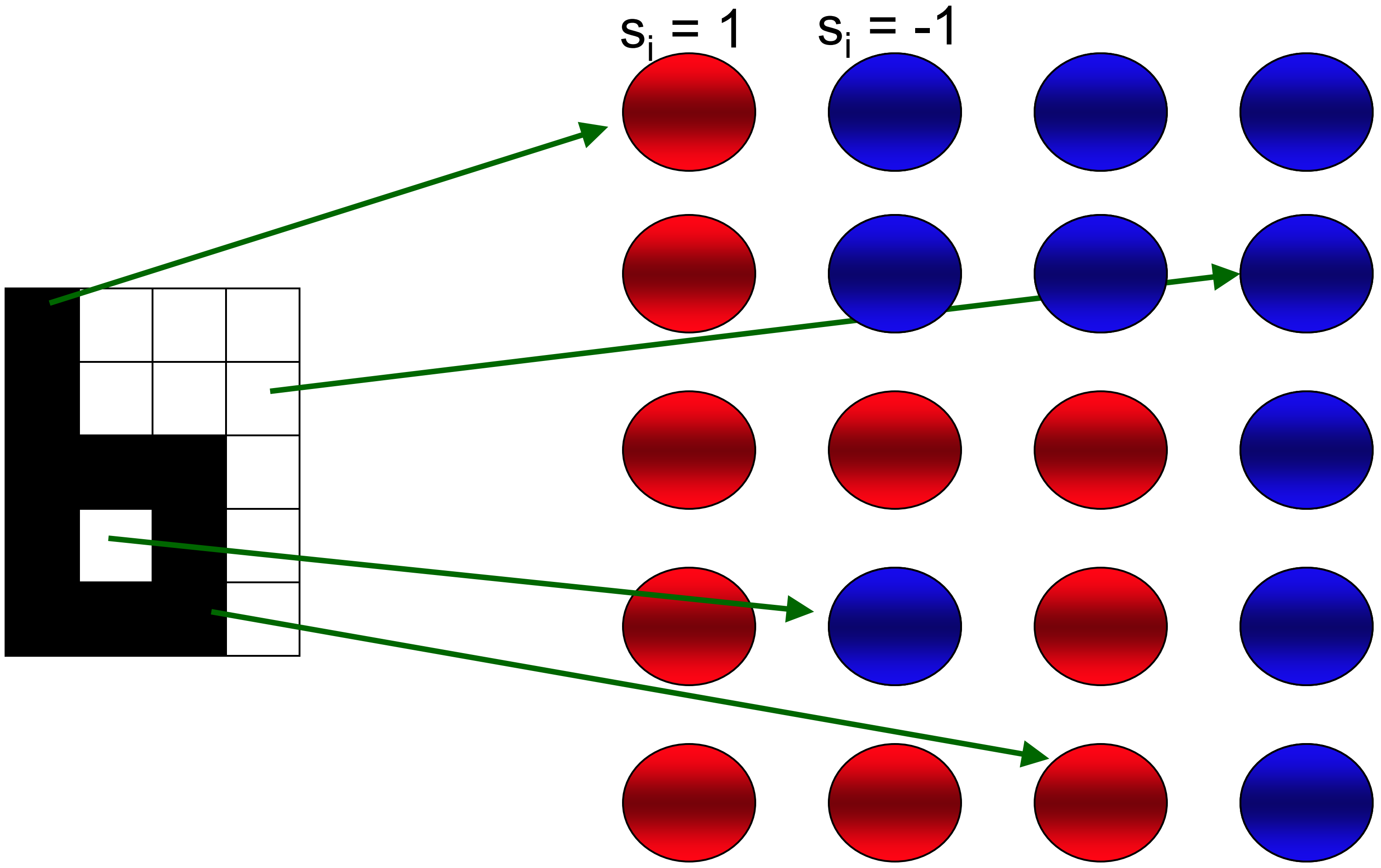
The Hopfield Network: A simple Attractor Network and Associative Memory

- All connected to every other neuron
- Synchronous or random update

$$s_i = \text{sign}\left(\sum_{j=1}^{n} w_{ij} s_j\right)$$



Representing Images



Learning the Pattern

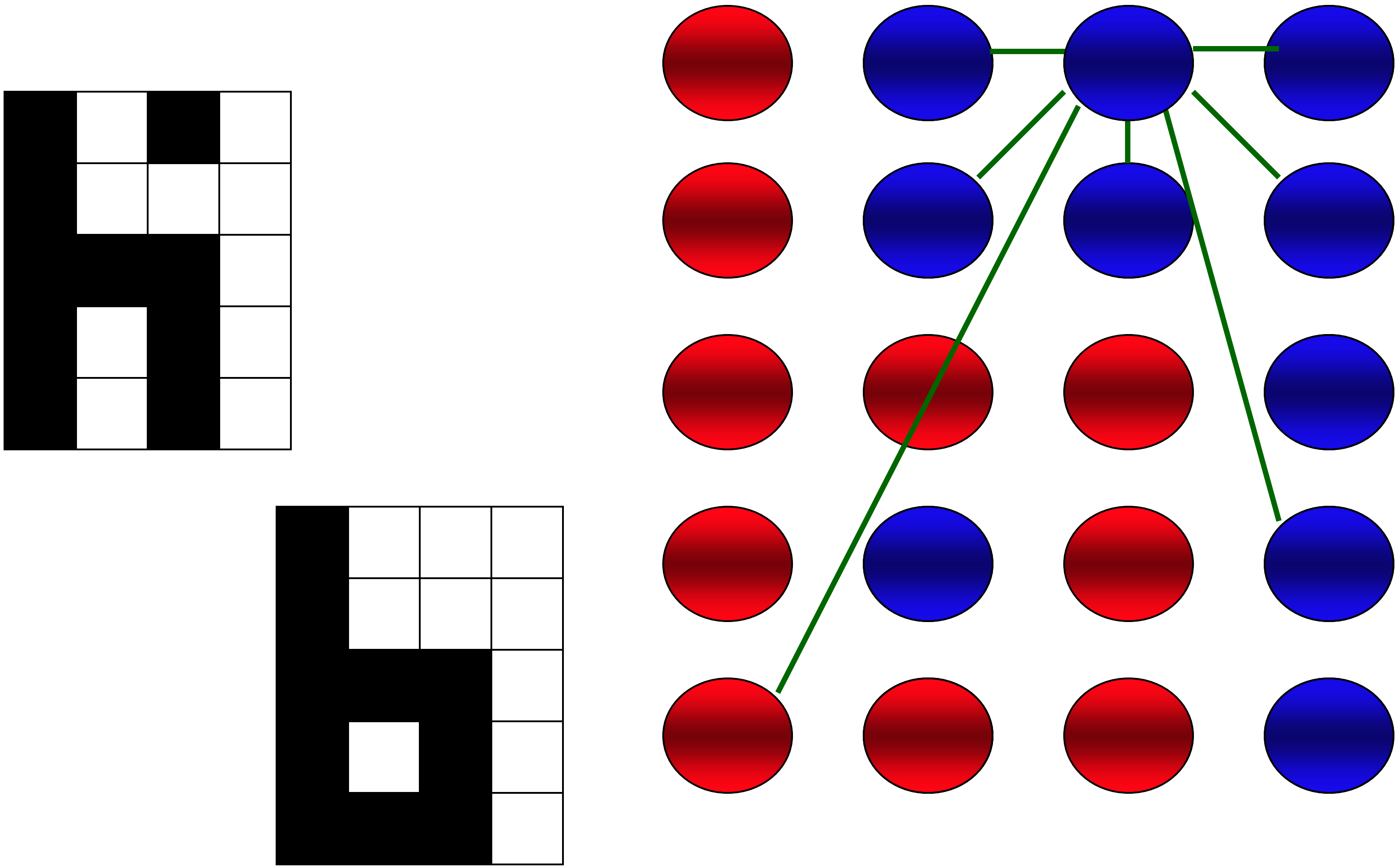
- Setting the weights:

$$s_i = \text{sign}\left(\sum_{j=1}^{n} w_{ij} s_j\right)$$

$$\Rightarrow w_{ij} = \frac{1}{N} s_i s_j$$

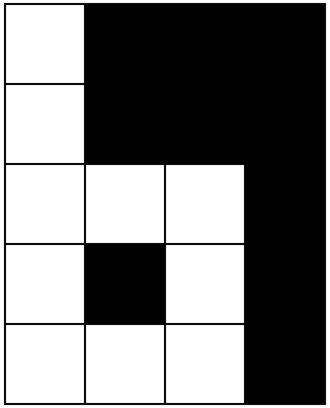
- only positive weight if both units are1 or -1

Using the Memory



Attractors

- The correct pattern is an ***attractor***
- Useful for constraint satisfaction
- Useful for noisy preclassification
- Useful for known attractors



More Patterns in the Memory

- Need to learn many patterns not just one
- Set the weights to the average for all input patterns p:

$$w_{ij} = \frac{1}{N} \sum_{p} s_i^p s_j^p$$

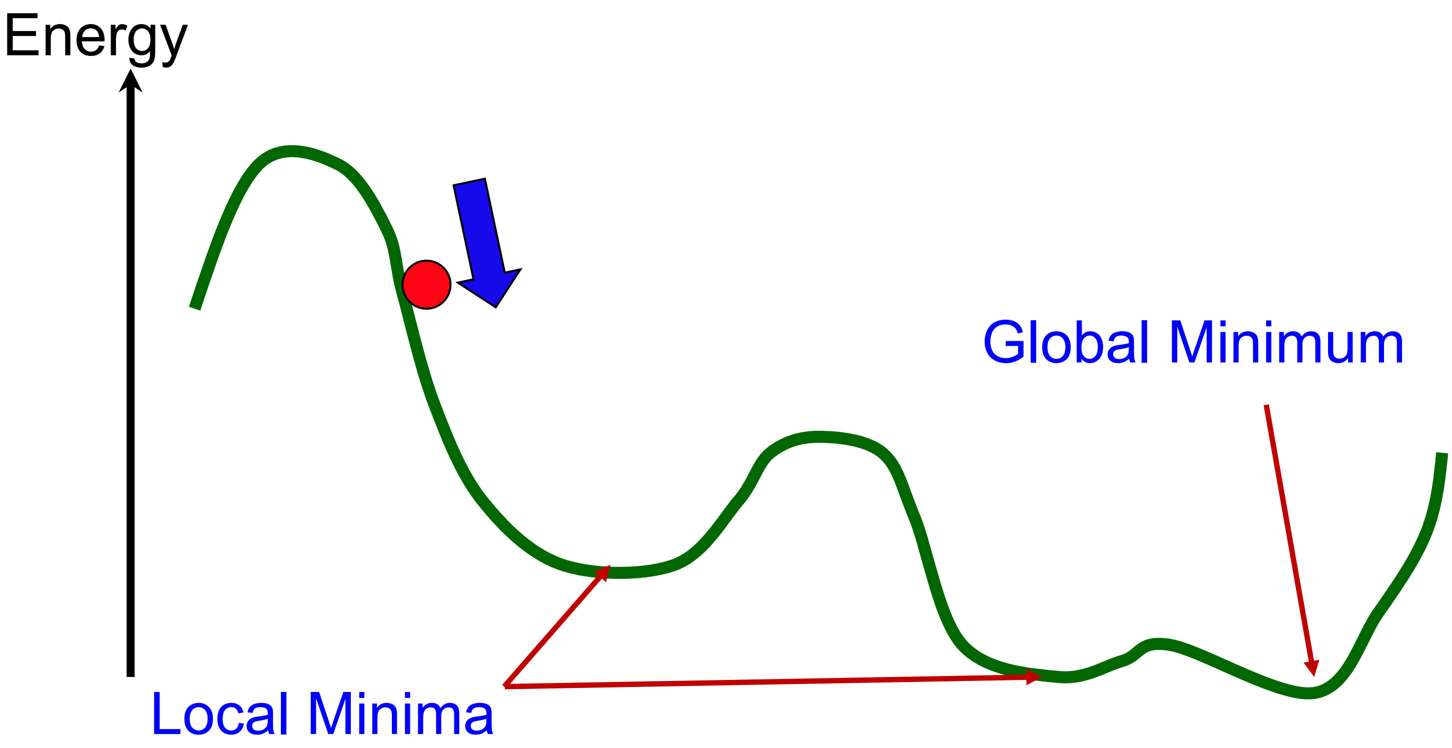
- For any input, network will ***converge*** to the closest trained pattern

“Energy” in the Network

$$H = -\frac{1}{2} \sum_{i} \sum_{j} w_{ij} s_i s_j$$

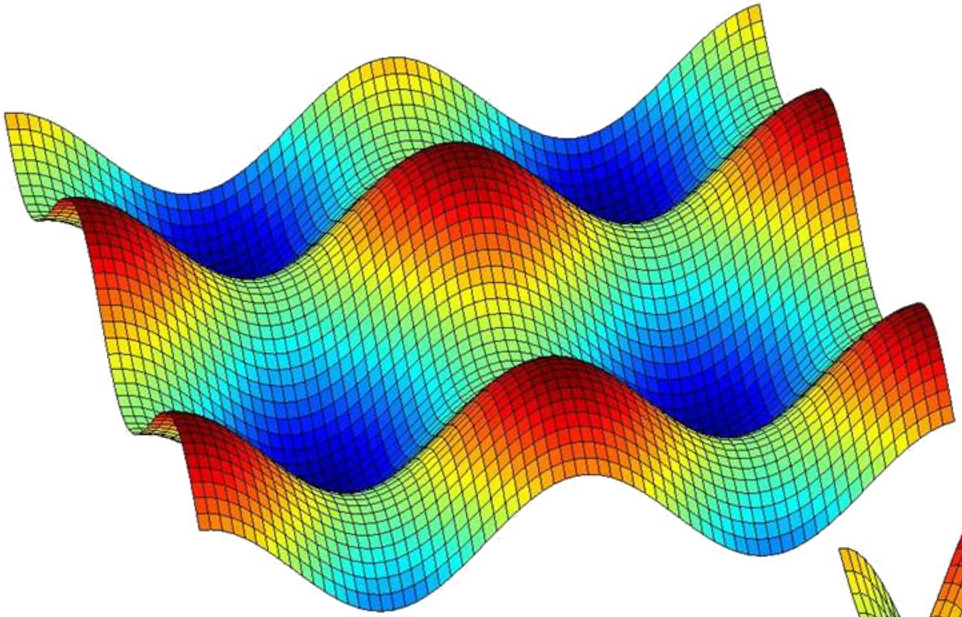
- This represents how much energy is in the network
- It decreases as the network stabilises
- The attractors are local minima of the energy function

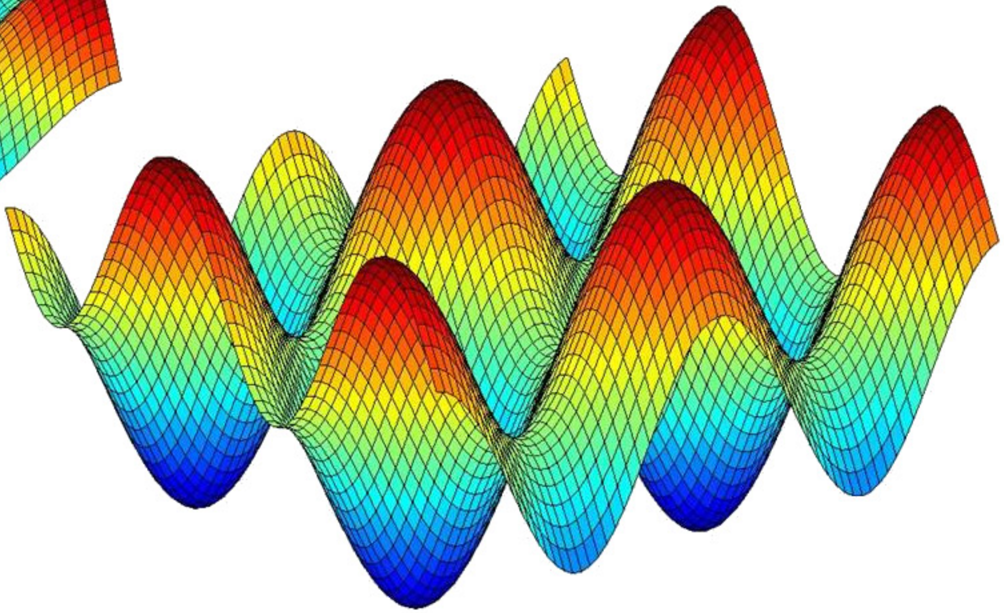
Energy Landscapes



Similar as in the context of Error function minimization

Energy Landscapes





Example and Data Mining Application: Hopfield Neural Network for Face Detection (1)

Reconstruct a learned pattern from ***noisy input***:

- Nodes are binary perceptrons p with:

$$p(x) = \begin{cases} 1 & \sum w_{ij} \cdot x_i > \theta \\ -1 & \text{otherwise} \end{cases}$$

- Every neuron is fully connected with all neurons except itself
- For HNNs with symmetric weights: Network converges to final ***stable state***
- Hebb-learning:

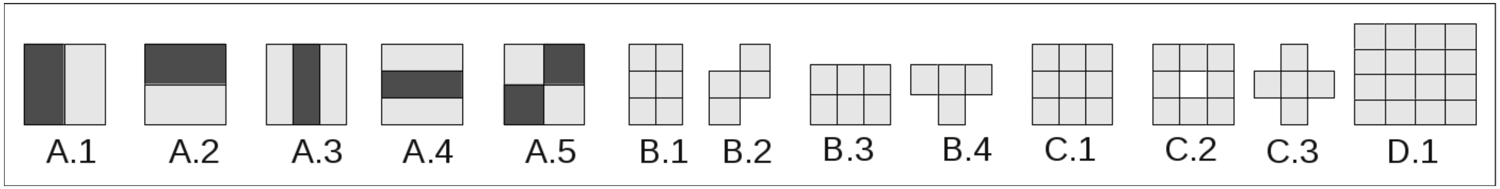
$$w_{ij} = \sum_{m=1}^{M} x_i^m \cdot x_j^m \quad \text{if } \; j \neq i,$$

$$w_{ij} = 0 \text{ otherwise}$$

M is the number of patterns

Hopfield Neural Network for Face Detection (2)

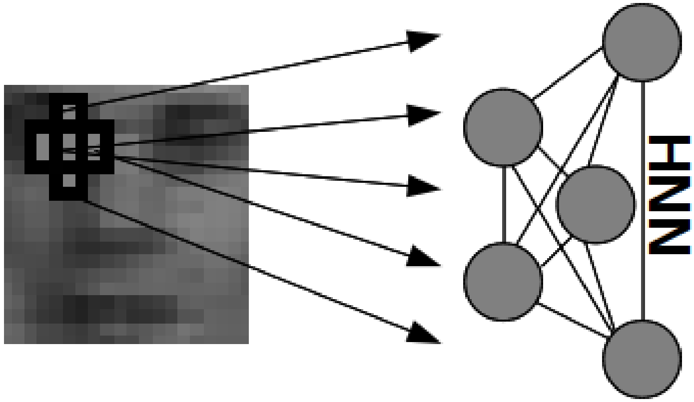
- Used “Haar-like” ***features***: Small sets of adjacent pixels



  - Efficient method for interesting aspects in images
  - Can be computed very fast
  - Examples of selected features:

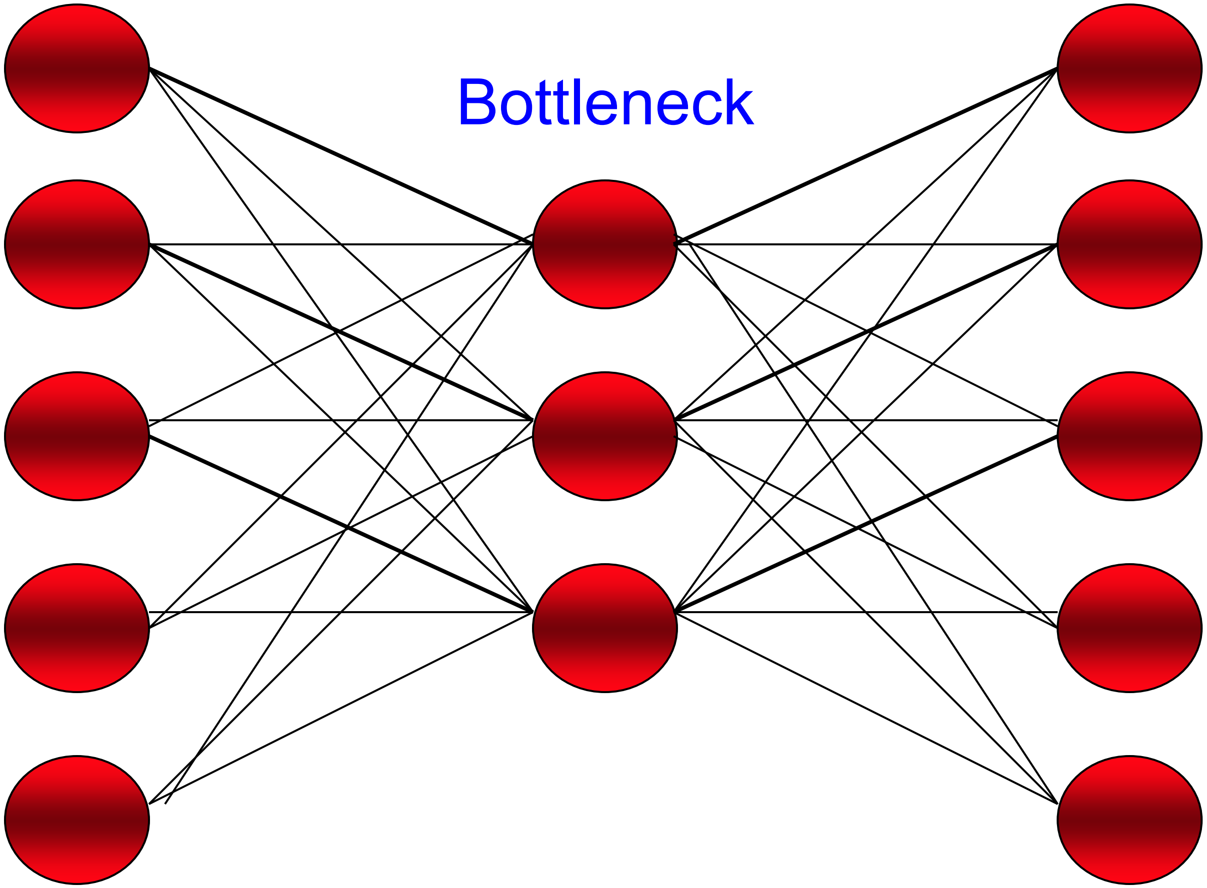
- Feed image patches (features) into the HNN

More details later in the lecture as well!



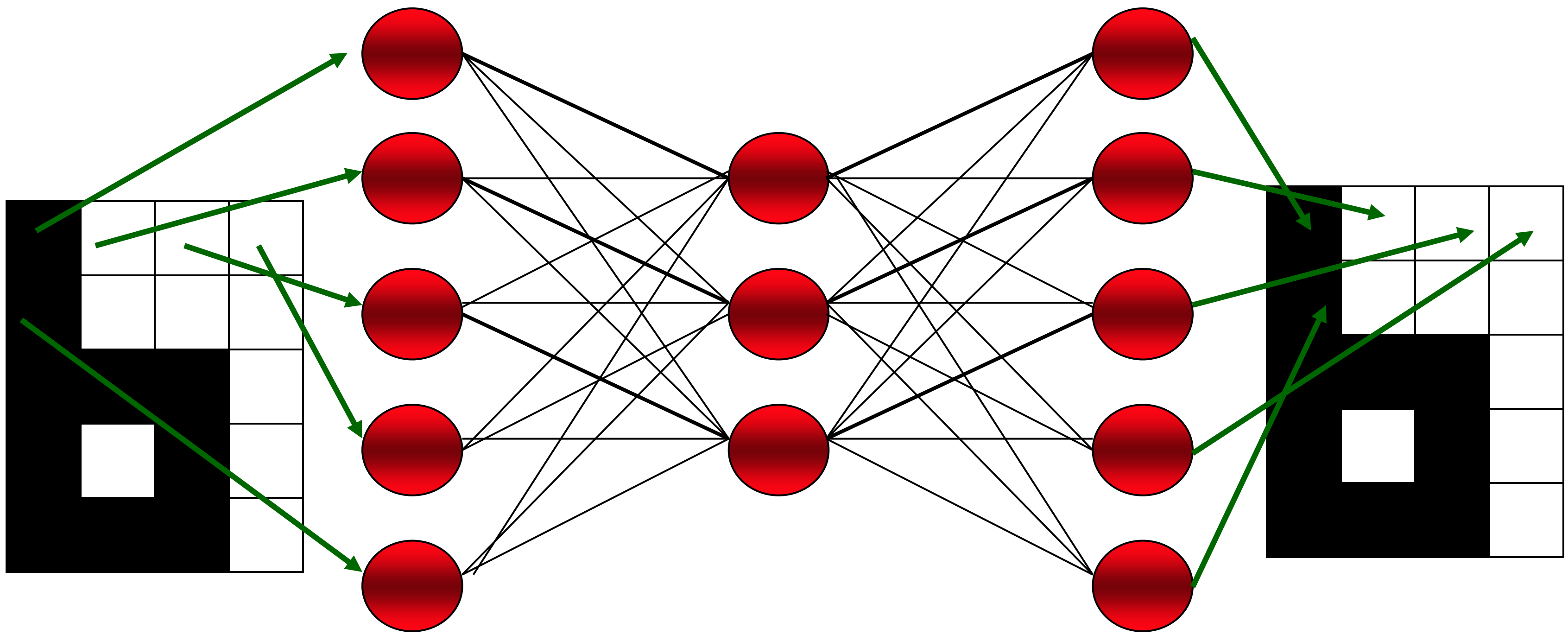
[Meins, Weber, Magg, Wermter 2012, 2013]

Multilayer Perceptron can be used as Autoassociative Network

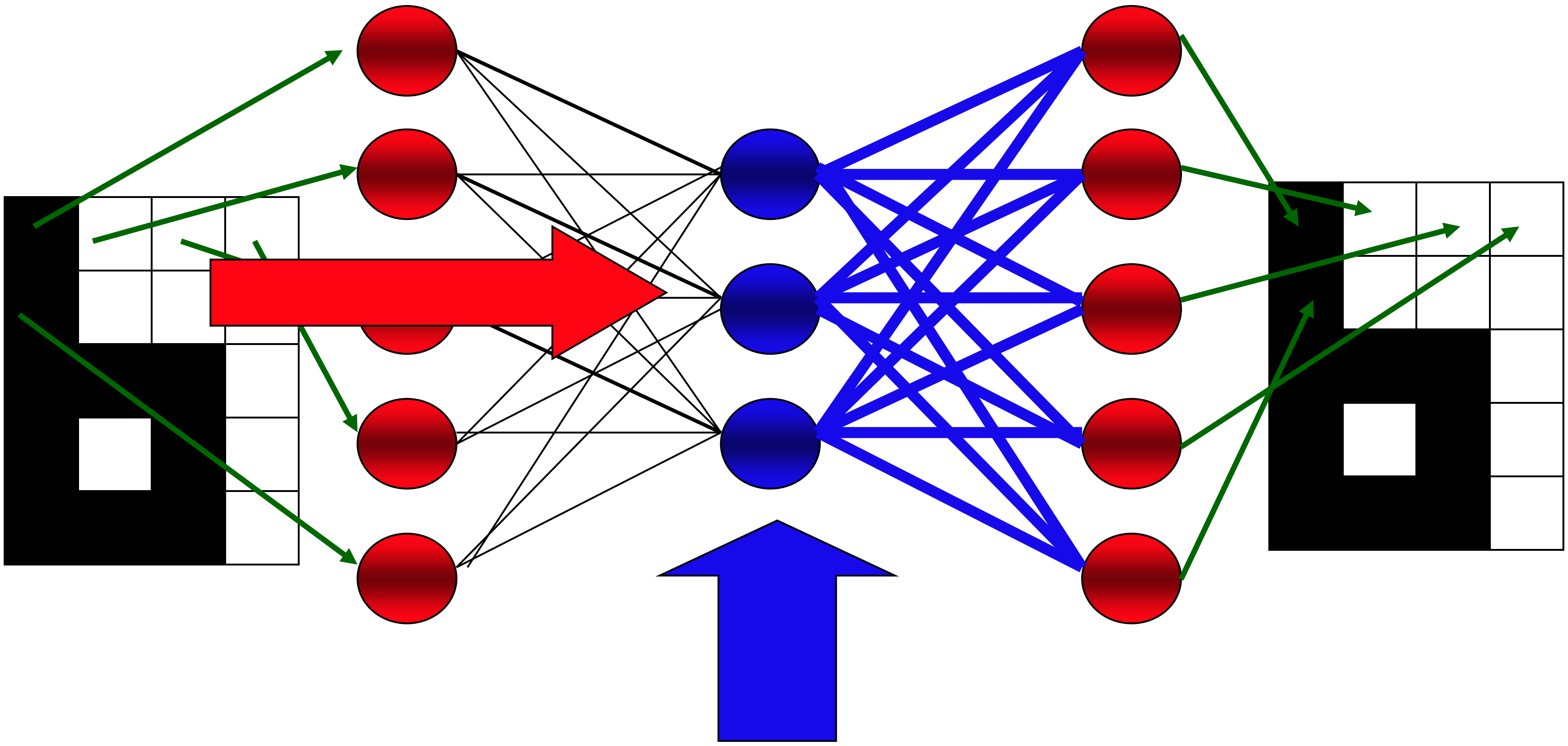


Autoassociative Network

Data compression
Use of less dimensions than the input
“Auto”: Same input and output

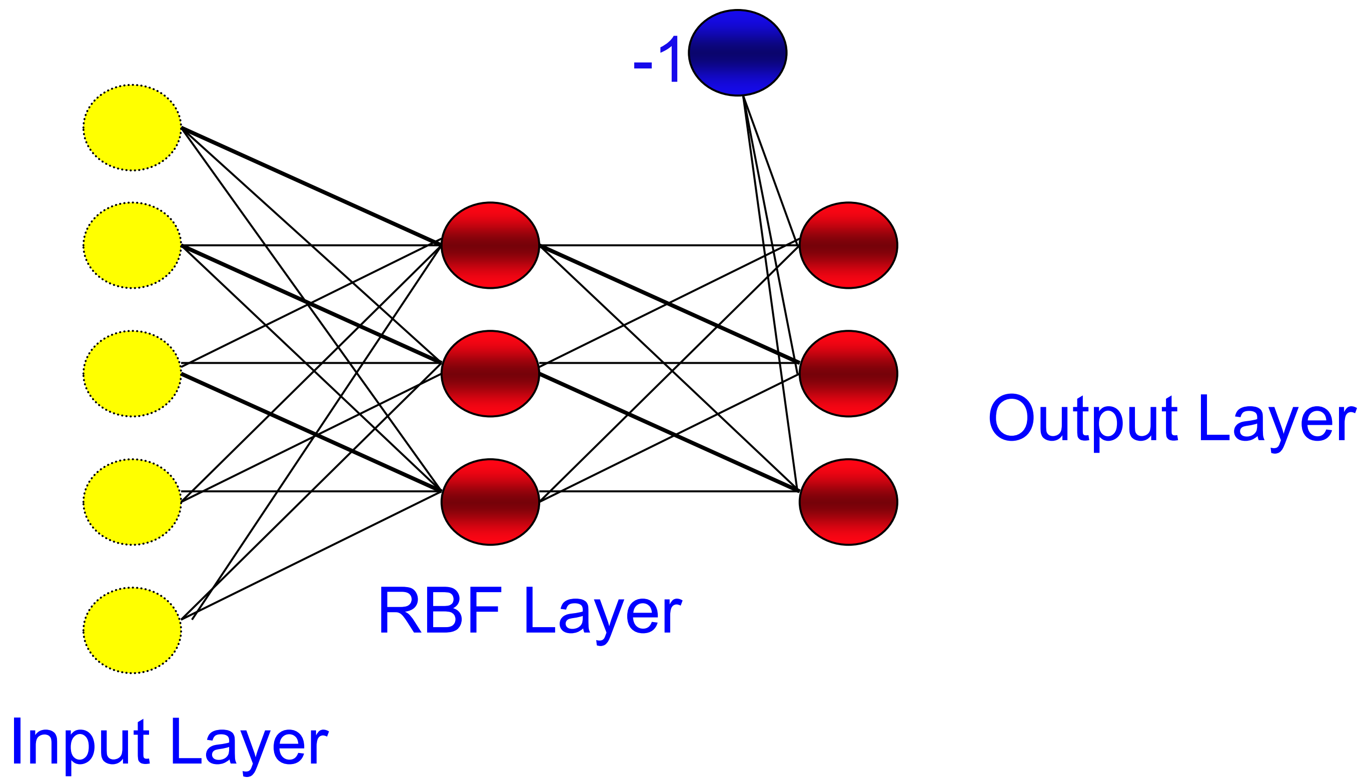


Autoassociative Network



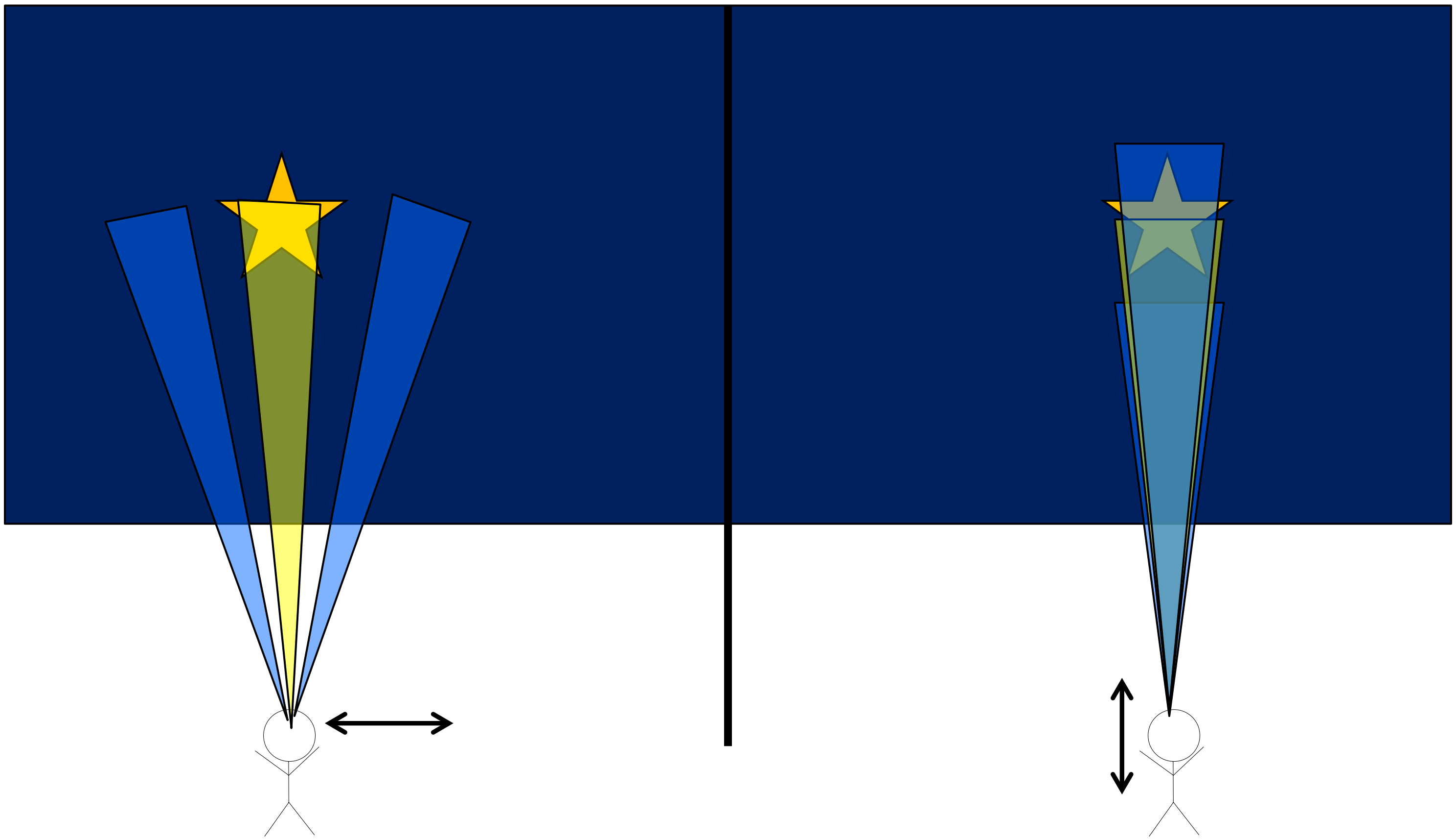
Store the second layer of weights and these activations

Moving from Memory to Function Approximation: Radial Basis Function Network



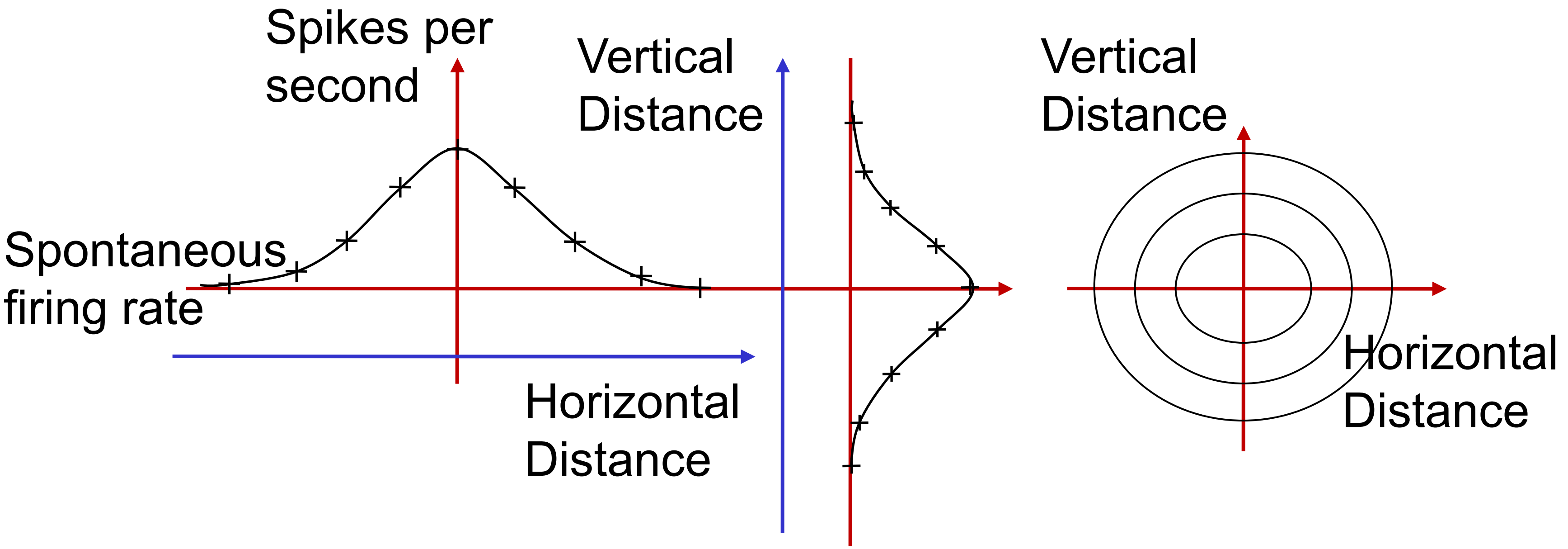
- RBF neurons fire proportionally to the distance between input and neuron in weight space
- No bias in hidden layer

Radial Basis Function Network: Idea (1)



Neurons firing more likely in certain horizontal or vertical positions

Radial Basis Function Network: Idea (2)

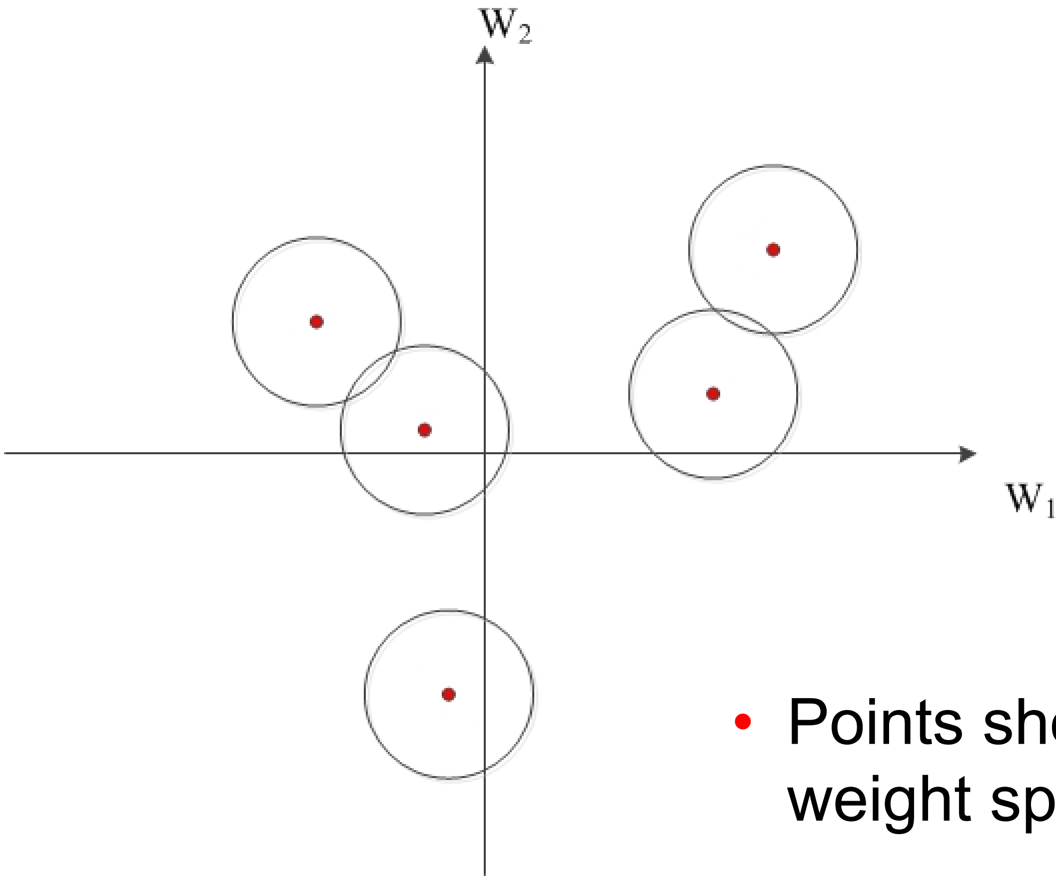


Count of the number of spikes per second as the distance of a rod from the light varies horizontally

The same case for vertical direction

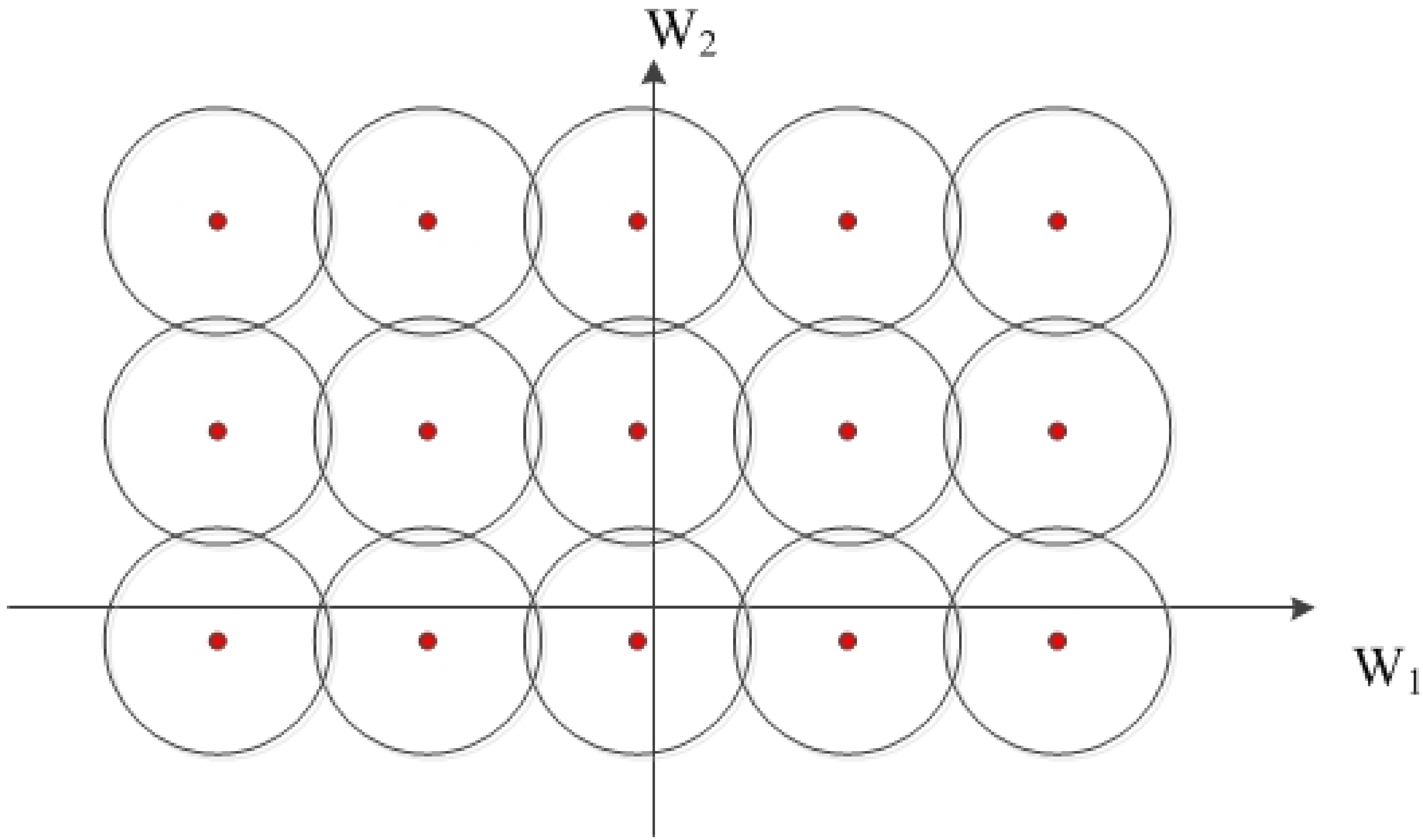
The combination of the two makes a set of circles

Nodes Representing Radial Bases in Weight Space



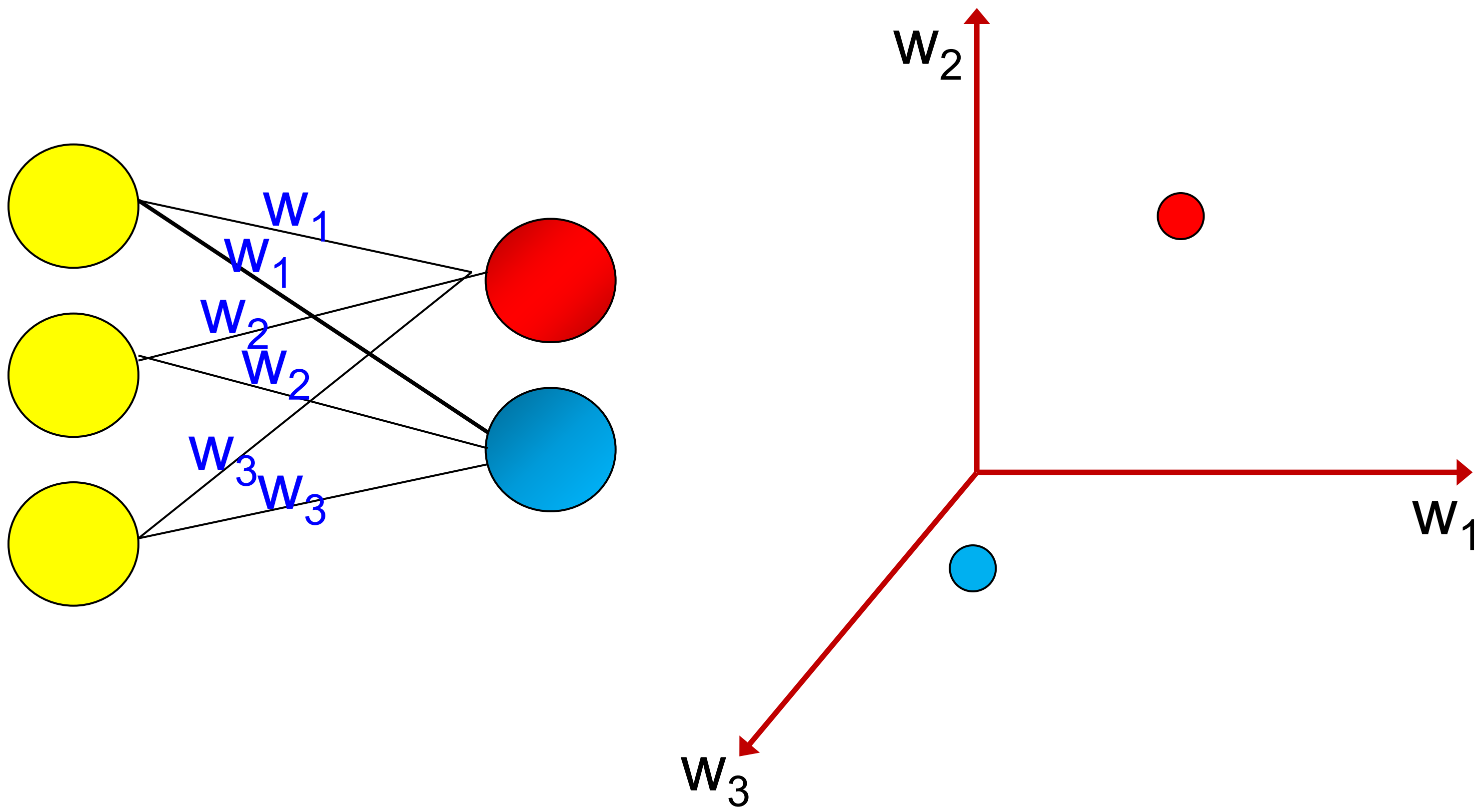
- Points show position of RBF in weight space
- Circle indicates receptive field

Using RBF as a Universal Approximator



RBF equally spaced nodes to cover the whole of weight space

Representing Receptive Fields with RBF



unit represented with its incoming weights to its inputs

The Radial Basis Function Algorithm

- Position the RBF ***centres*** by either:
  - Using the k-means algorithm to initialise the positions of the RBF centres **OR**
  - Setting the RBF centres to be randomly chosen datapoints

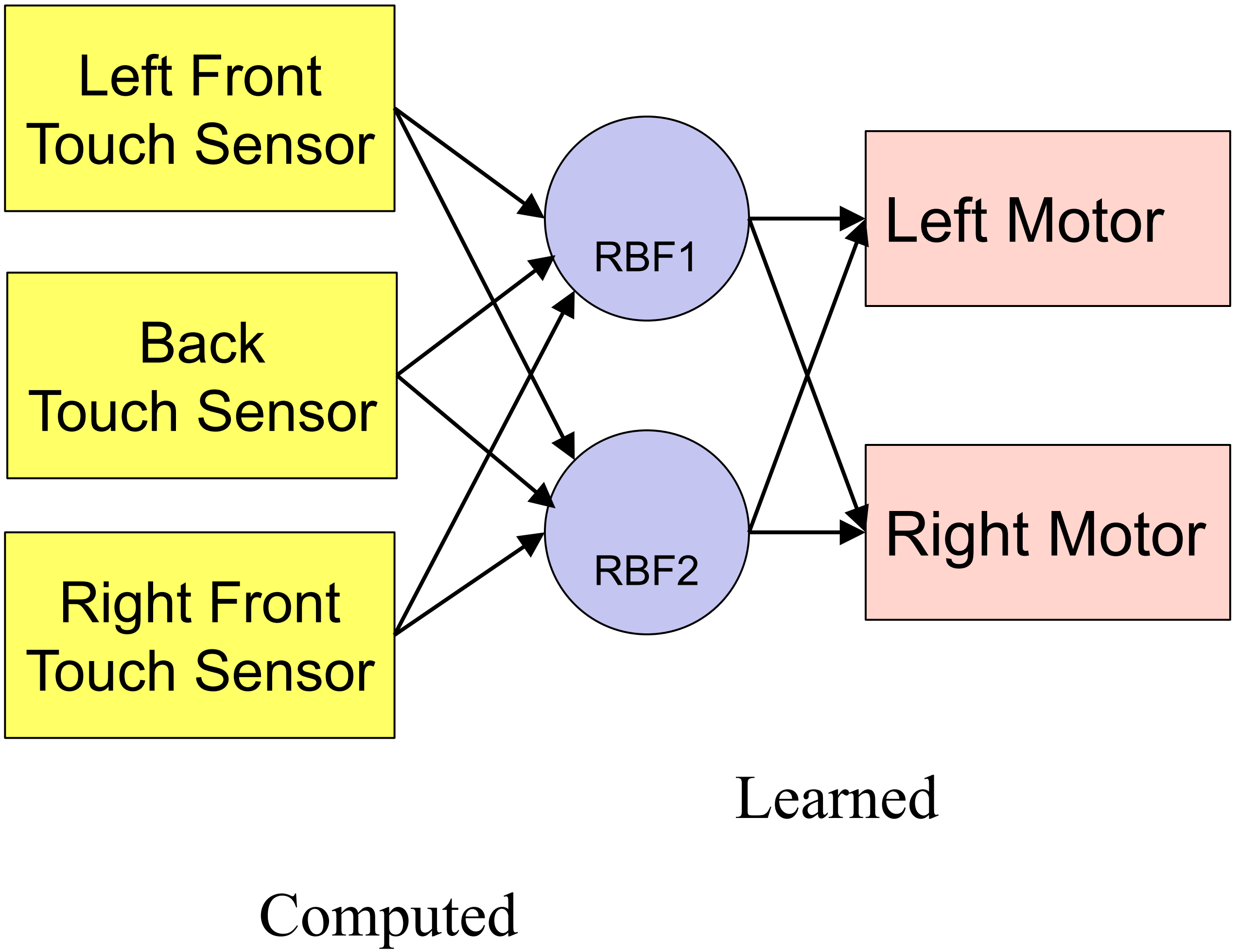
- Calculate the ***actions of the RBF nodes*** using the ***radial basis equation***

$$g(x, w, \sigma) = \exp\left(\frac{-\|x - w\|^2}{2\sigma^2} \right)$$

Controls width of Gaussian

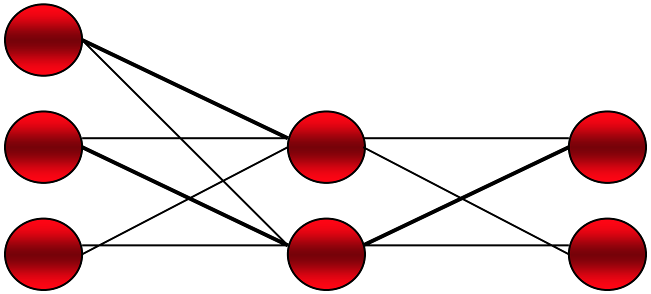
- Train the ***output weights*** e.g. by
  - Using the Perceptron

RBF example: Reactive Navigation Network



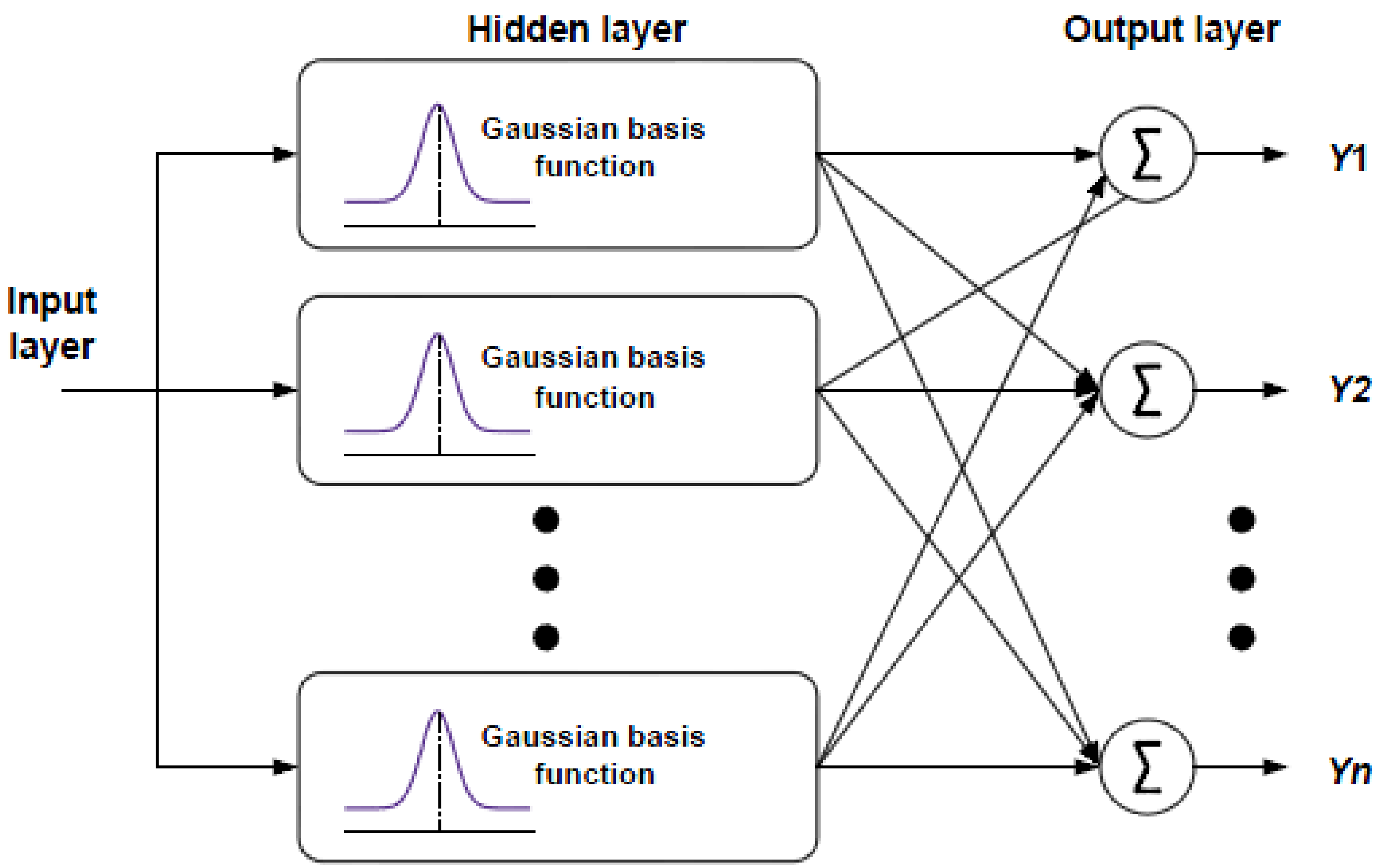
RBF example: Reactive Navigation Network

Input			Output	
Sensor Front Left	Sensor Back	Sensor Front Right	Motor Left	Motor Right
1	0	0	0	-1
0	1	0	1	1
0	0	1	-1	0
1	0	1	-1	-1
1	1	0	0	1
0	1	1	1	0
1	1	1	0	0
0	0	0	1	1



RBF example: Reactive Navigation Network

Rule-extraction and Rule Insertion with RBF Networks



Rule Extraction Algorithm

- Input:
  - Hidden weights η (centre position)
- Output:
  - One rule per output class
- Procedure:
  1. Train RBF network on data set
  2. Cluster hidden units by class
  3. For each class cluster
  4. For each η
  5. Get min value
  6. Get max value
  7. For each class
  8. Write out rule by:
  9. For each η = [min-max] interval
  10. Join intervals with AND
  11. Add Class label
  12. Write rule to file

Extracted Rules from the IRIS Data Set

- Rule 1

  IF (SL ≥ 4.4 AND ≤ 5.7) ***AND***

  IF (SW ≥ 2.9 AND ≤ 4.4) ***AND***

  IF (PL ≥ 1.3 AND ≤ 1.5) ***AND***

  IF (PW ≥ 0.2 AND ≤ 0.4)

  THEN..Setosa

- Rule 2

  IF (SL ≥ 4.9 AND ≤ 6.9) ***AND***

  IF (SW ≥ 2.0 AND ≤ 3.1) ***AND***

  IF (PL ≥ 3.5 AND ≤ 5.0) ***AND***

  IF (PW ≥ 1.0 AND ≤ 1.7)

  THEN..Versicolor

- Rule 3

  IF (SL ≥ 5.8 AND ≤ 7.2) ***AND***

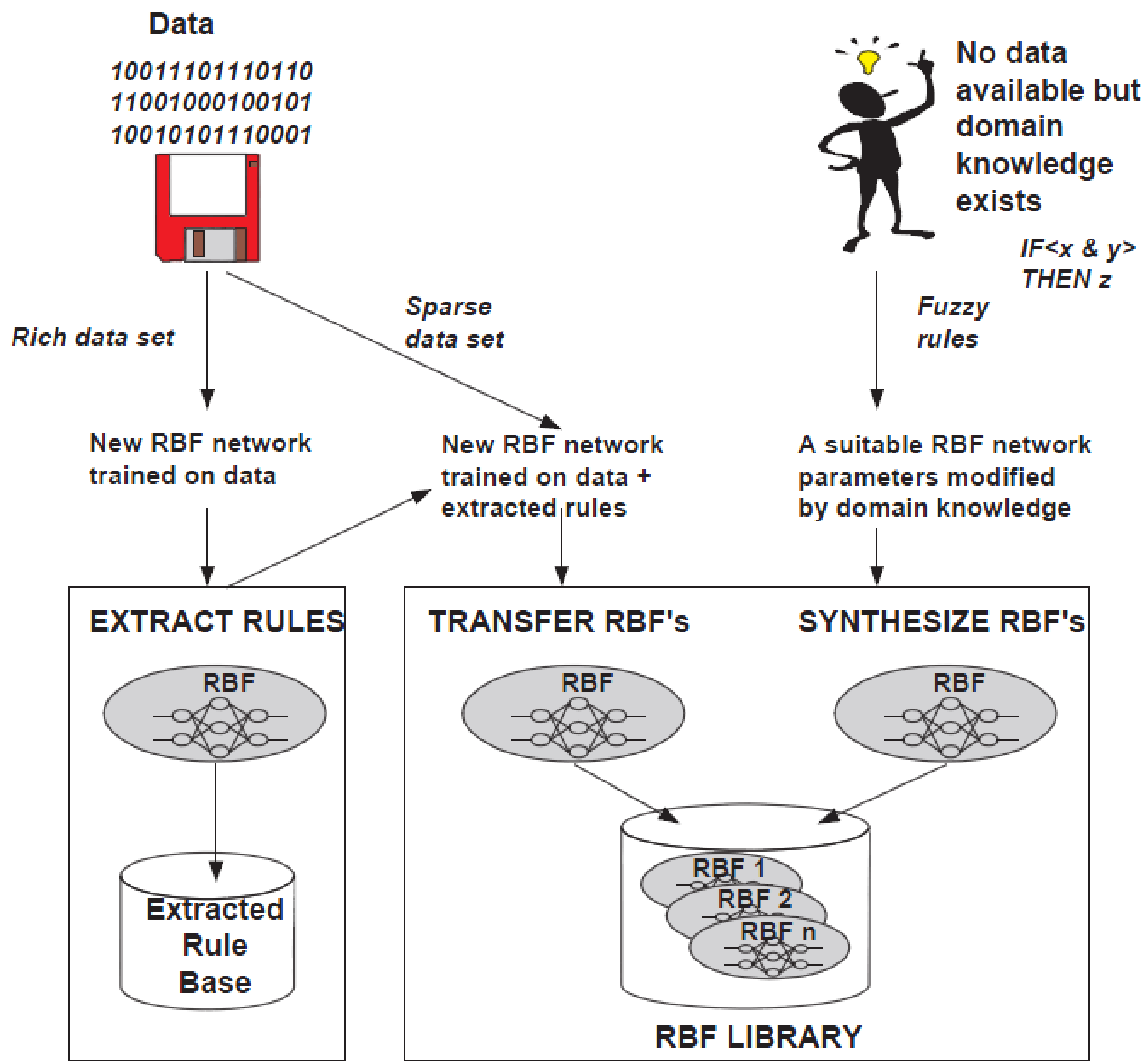
  IF (SW ≥ 2.8 AND ≤ 3.1) ***AND***

  IF (PL ≥ 4.5 AND ≤ 5.8) ***AND***

  IF (PW ≥ 1.5 AND ≤ 2.4)

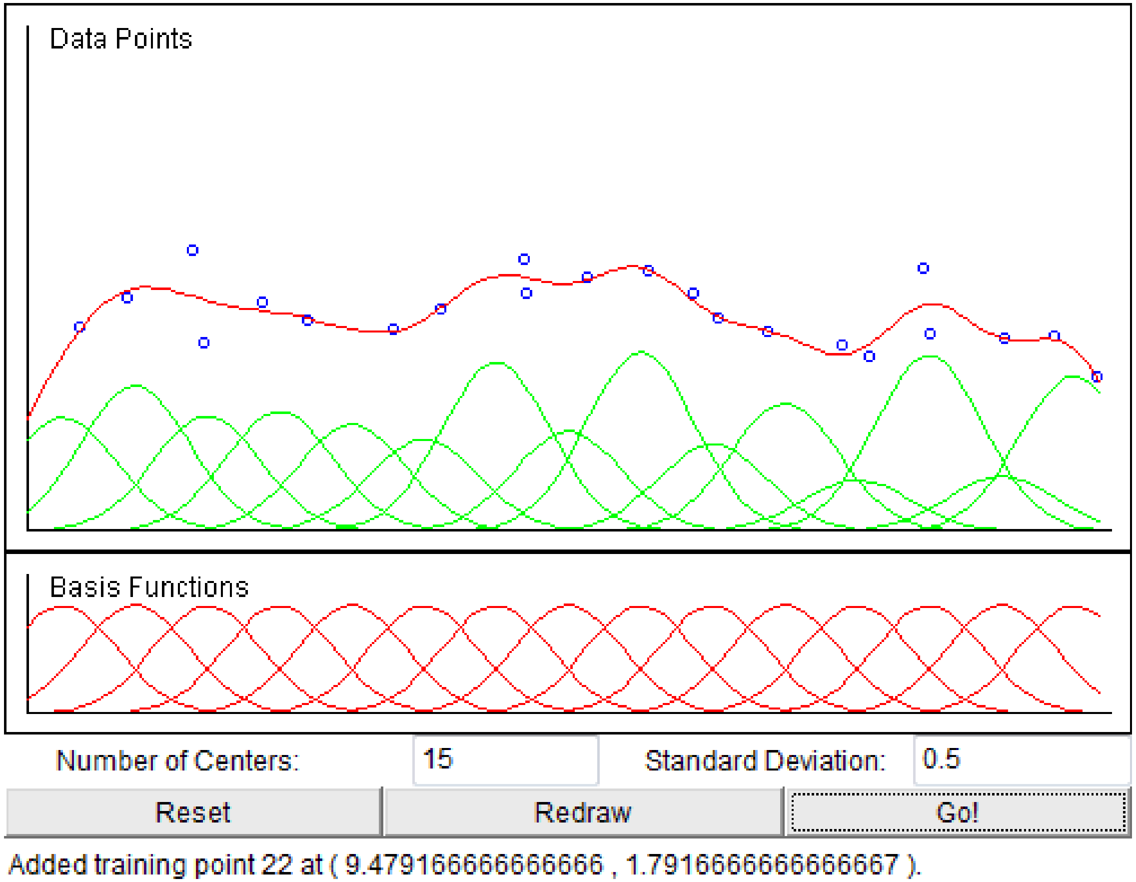
  THEN..Virginica

Training without Data:
Knowledge Insertion into RBF Networks



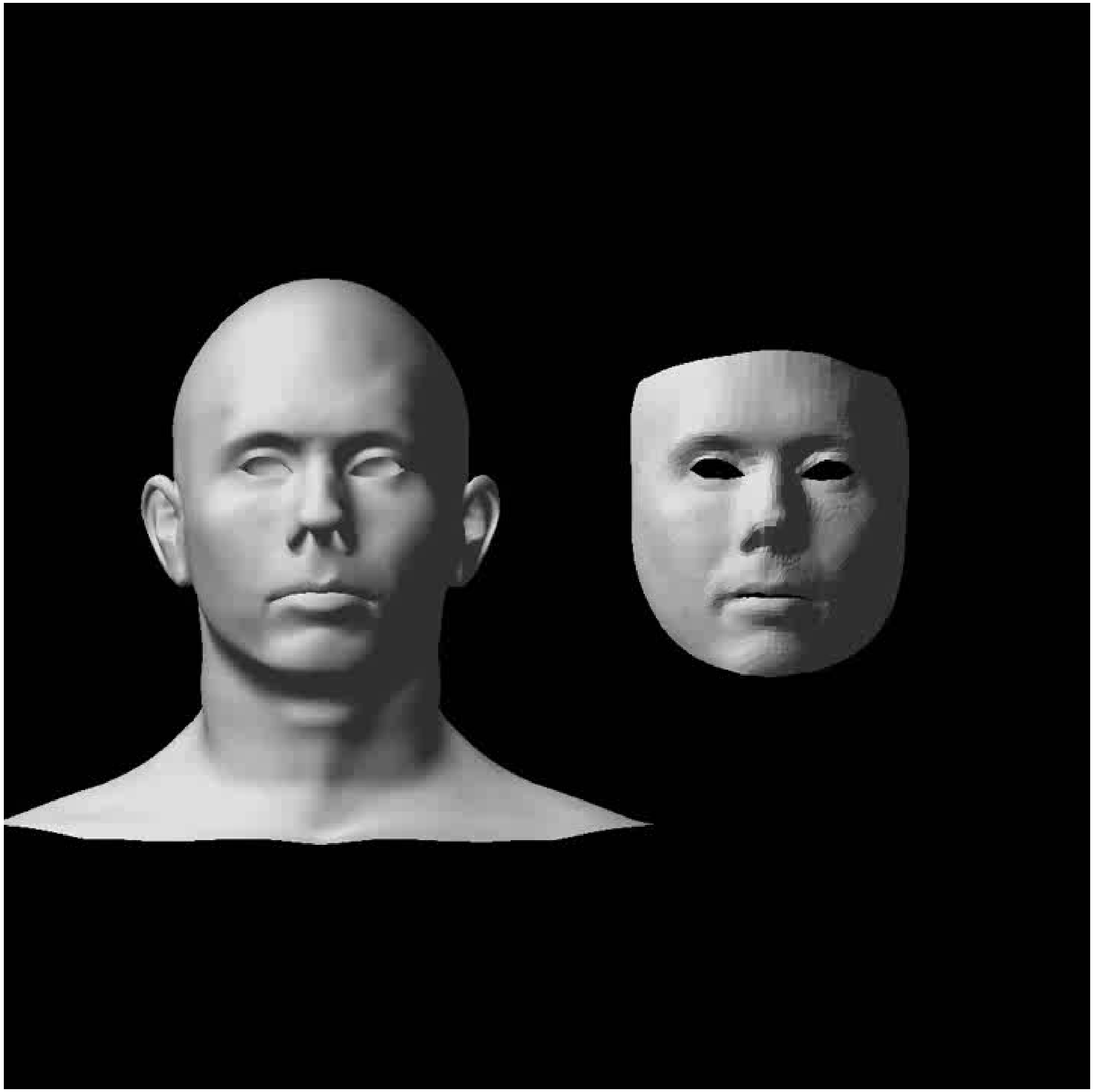
Example of Approximation Capability of Radial Basis Function Network - Script online!

http://lcn.epfl.ch/tutorial/english/rbf/html/index.html



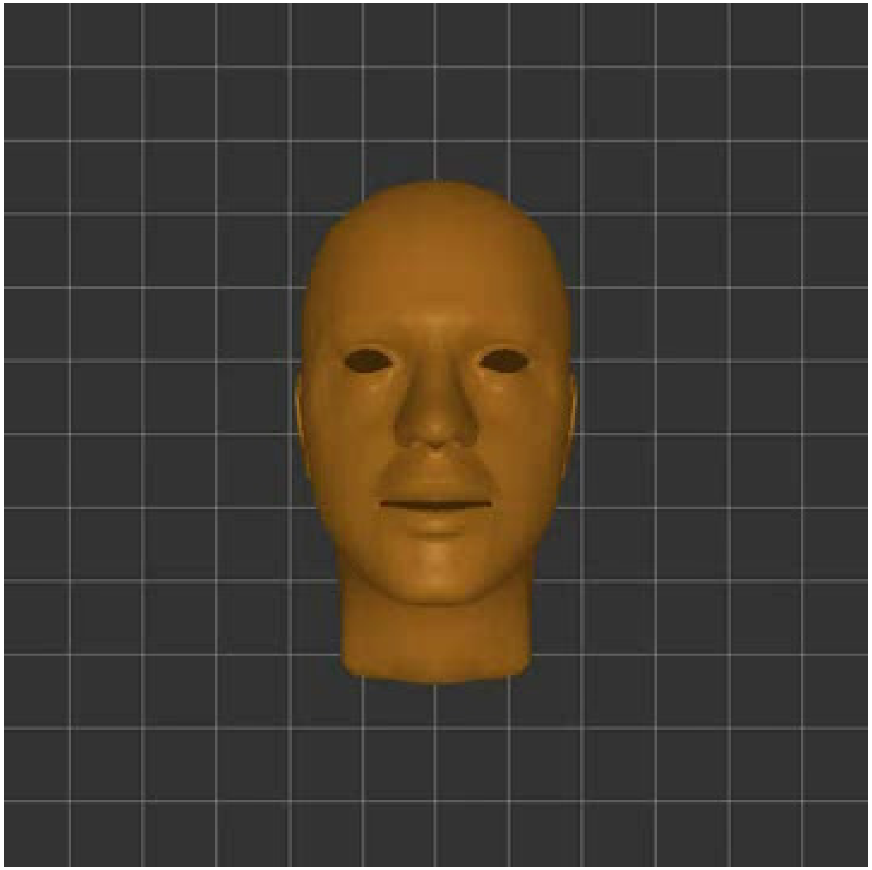
Test it online!

Example: Mesh Skinning for Face Animation with Radial Basis Functions



[James D. Edge, University of Surrey]

Example: Modeling a Face Mesh with RBFs



[http://cg.alexandra.dk/]

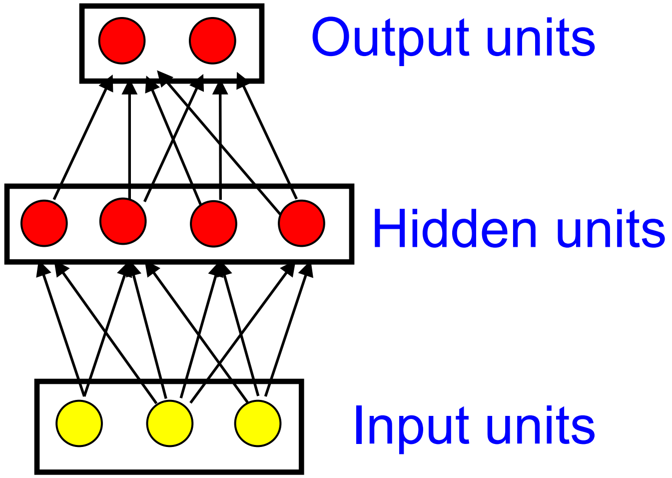
Intermediate Recap and Summary

- MLP can ***approximate*** and learn ***continuous*** functions
- Due to ***nonlinear*** activation function and multiple layers more powerful than perceptron learning
- RBFs introduce the concept of ***locality*** into a network

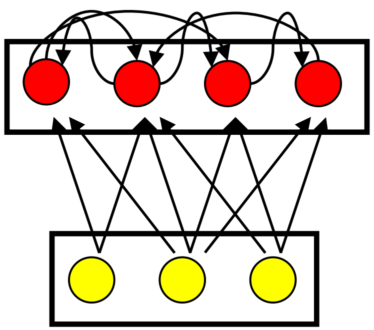
- Reading and experiments
  - Marsland chapter 3 on MLP, chapter 5 on RBFs
  - Experiment with the code!

Introduction to Recurrent Neural Networks: Types of Connectivity

- Feedforward networks
  - These compute a series of transformations.
  - Typically, the first layer is the input and the last layer is the output.



- Recurrent networks
  - These have directed cycles in their connection graph. They can have complicated dynamics.
  - More biologically realistic.



Simple Recurrent Network (SRN)

- Problem with Time

  [0 1 1 1 0 0 0 0 0]

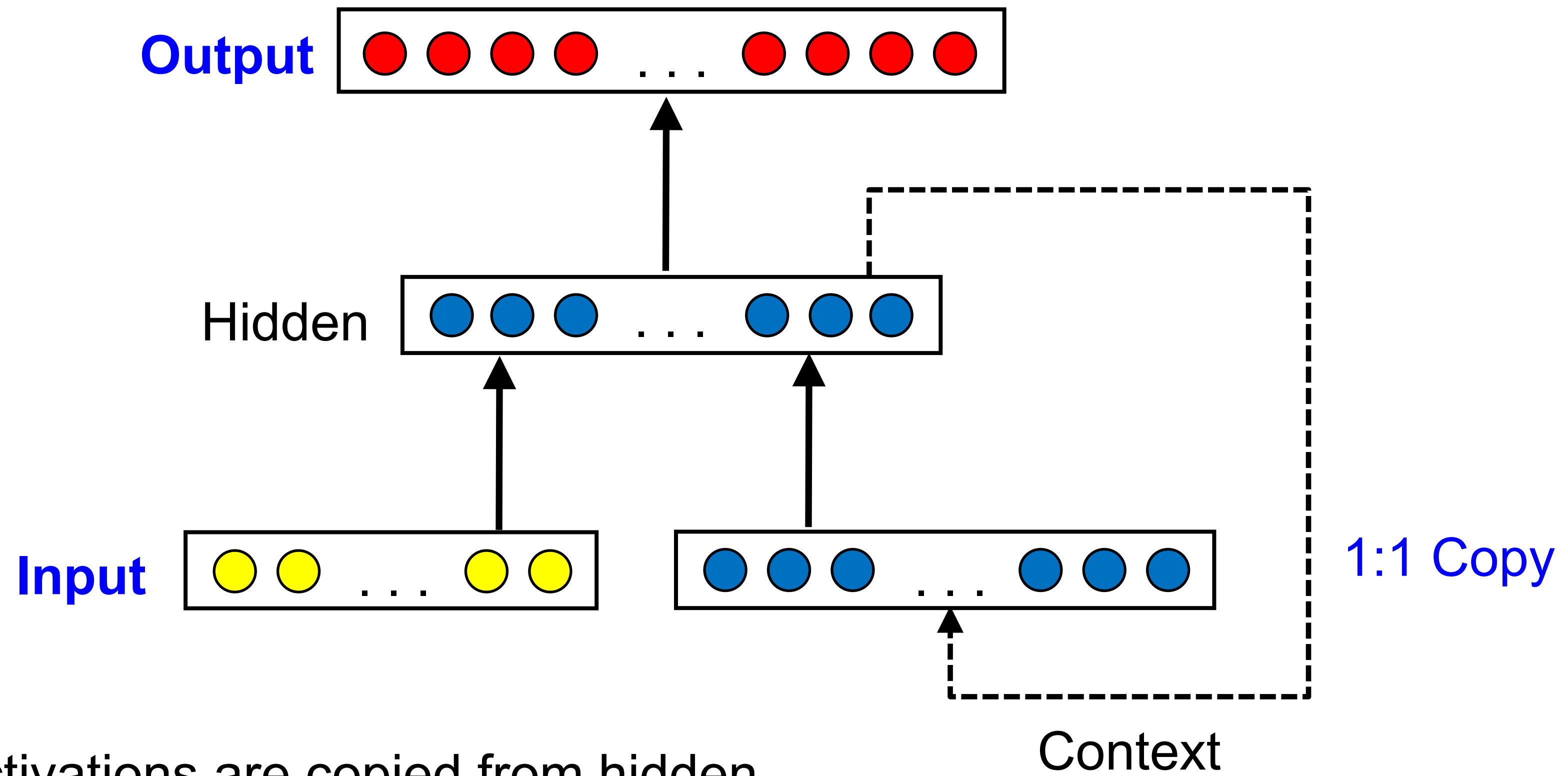
  [0 0 0 1 1 1 0 0 0]

- Two vectors appear to be instances of the ***same*** basic ***pattern***, but displaced in space
- Relative temporal structure should be ***preserved*** in the face of absolute temporal displacements

Simple Recurrent Network (SRN)

- Motivation for using ***internal state*** information
- Copy the internal hidden layer for next input
- Paper: Elman J., Finding Structure in Time, Cognitive Science 14, 1990

Simple recurrent network (SRN)



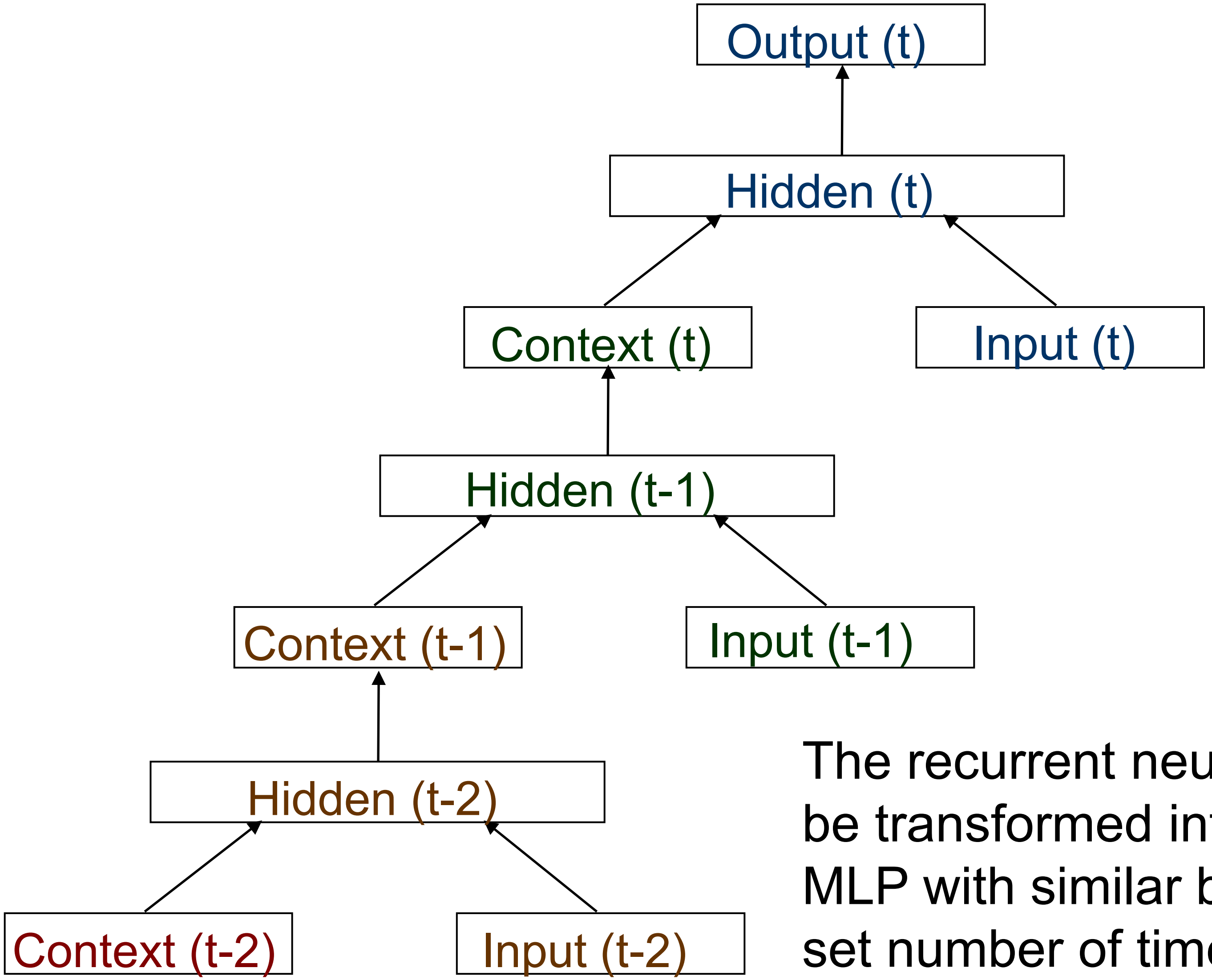
- Activations are copied from hidden layer to context layer on a one-for-one basis, with fixed weight of 1.0.
- Straight lines represent trainable connections.

Example Prediction

Input: $w_1\ w_2\ w_3\ ....\ w_n$

Output: $w_2\ w_3\ w_4\ ....\ w_{n+1}$

Backpropagation Through Time (e.g. 3 steps)



The recurrent neural network can be transformed into a feed-forward MLP with similar behaviour over a set number of time steps.

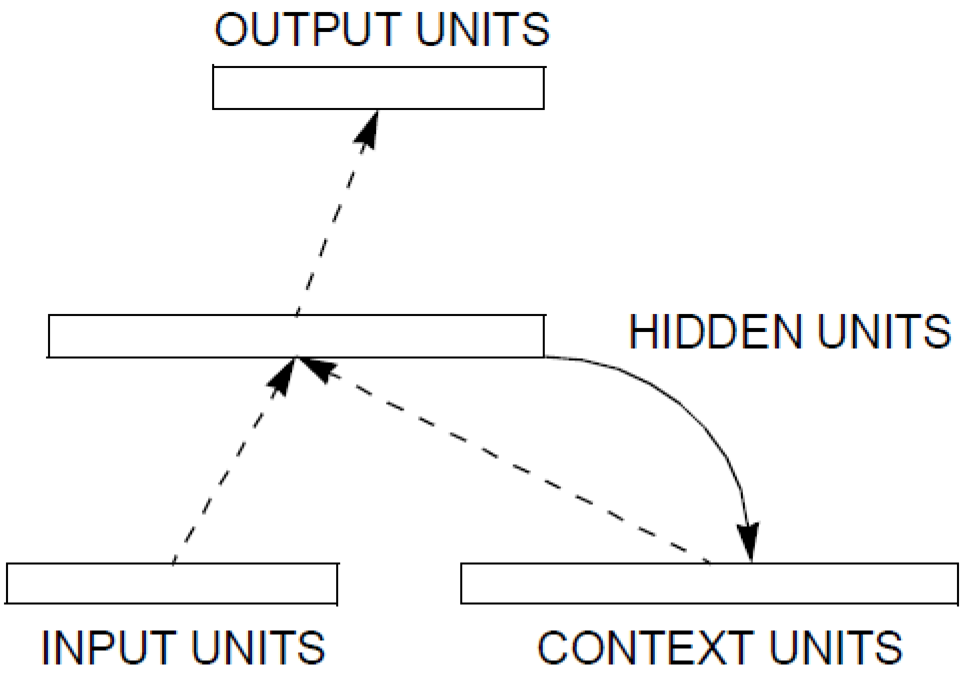
Learning Structure in Letter Sequences

- Multi-bit inputs with temporal extent and longer sequences
- 3 consonants (**b**, **d**, **g**) combined in random order to obtain 1000-letter sequence. Then each consonant replaced using rules
  - b->ba
  - d->dii
  - g->guuu
- **Example**: dbgbddg... into diibaguuubadiidiiguuu
- Task: predict next letter (uncertainty: sometimes clear, sometimes not)

Vector Definitions of Alphabet

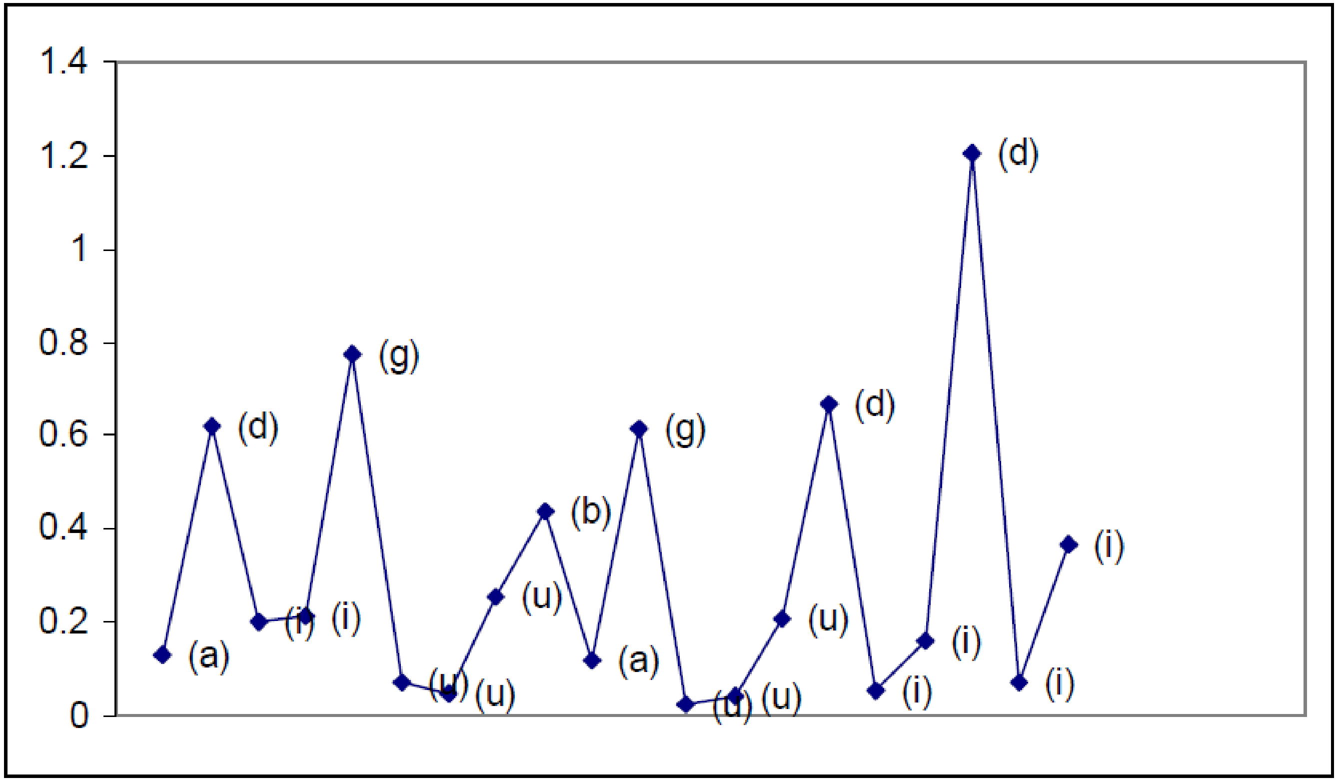
	Consonant	Vowel	Interrupted	High	Back	Voiced
b	[1	0	1	0	0	1]
d	[1	0	1	1	0	1]
g	[1	0	1	0	1	1]
a	[0	1	0	0	1	1]
i	[0	1	0	1	0	1]
u	[0	1	0	1	1	1]

SRN for Letter Sequences



6 input units, 20 hidden units,
6 output units, and 20 context units

Root Mean Squared Error in Letter Prediction Task



- Labels indicate the correct output prediction at each point in time
- Once network has consonant as input, it can predict following vowel
- After a “d”, “g”, “b” the prediction error drops, then it rises

Learning Lexical Classes from Word Order

- Order of words is constraint
- Can a network learn ***structure*** from order?
- Sentence generator based on categories of lexical items
- Each word represented by random 31 bit vector
- Each word represented by a different bit which is on if word present
- 27,354 word vectors in the 10,000 sentences were concatenated

Categories of Lexical Items

Category	Examples
NOUN-HUM	man, woman
NOUN-ANIM	cat, mouse
NOUN-INANIM	book, rock
NOUN-AGRESS	dragon, monster
NOUN-FRAG	glass, plate
NOUN-FOOD	cookie, sandwich
VERB-INTRAN	think, sleep
VERB-TRAN	see, chase
VERB-AGPA	move, break
VERB-PERCEPT	smell, see
VERB-DESTROY	break, smash
VERB-EA	eat

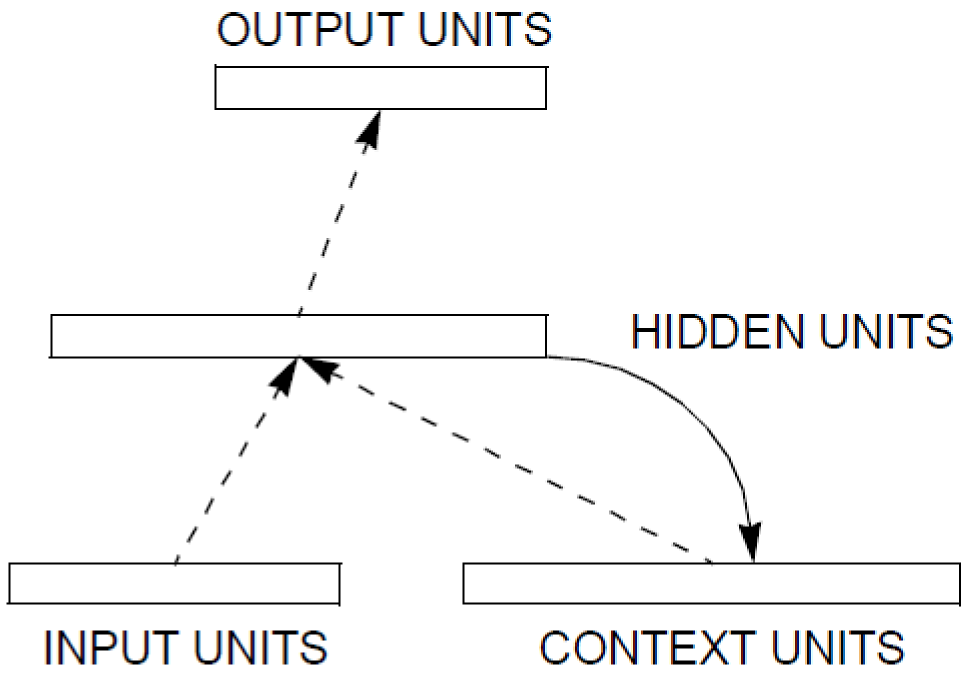
Templates for Sentence Generator

WORD 1	WORDS	WORD 3
NOUN-HUM	VERB-EAT	NOUN-FOOD
NOUN-HUM	VERB-PERCEPT	NOUN-INANIM
NOUN-HUM	VERB-DESTROY	NOUN-FRAG
NOUN-HUM	VERB-INTRAN	
NOUN-HUM	VERB-TRAN	NOUN-HUM
NOUN-HUM	VERB-AGPAT	NOUN-INANIM
NOUN-HUM	VERB-AGPAT	
NOUN-ANIM	VERB-EAT	NOUN-FOOD
NOUN-ANIM	VERB-TRAN	NOUN-ANIM
NOUN-ANIM	VERB-AGPAT	NOUN-INANIM
NOUN-ANIM	VERB-AGPAT	
NOUN-INANIM	VERB-AGPAT	
NOUN-AGRESS	VERB-DESTORY	NOUN-FRAG
NOUN-AGRESS	VERB-EAT	NOUN-HUM
NOUN-AGRESS	VERB-EAT	NOUN-ANIM
NOUN-AGRESS	VERB-EAT	NOUN-FOOD

Learning successive words

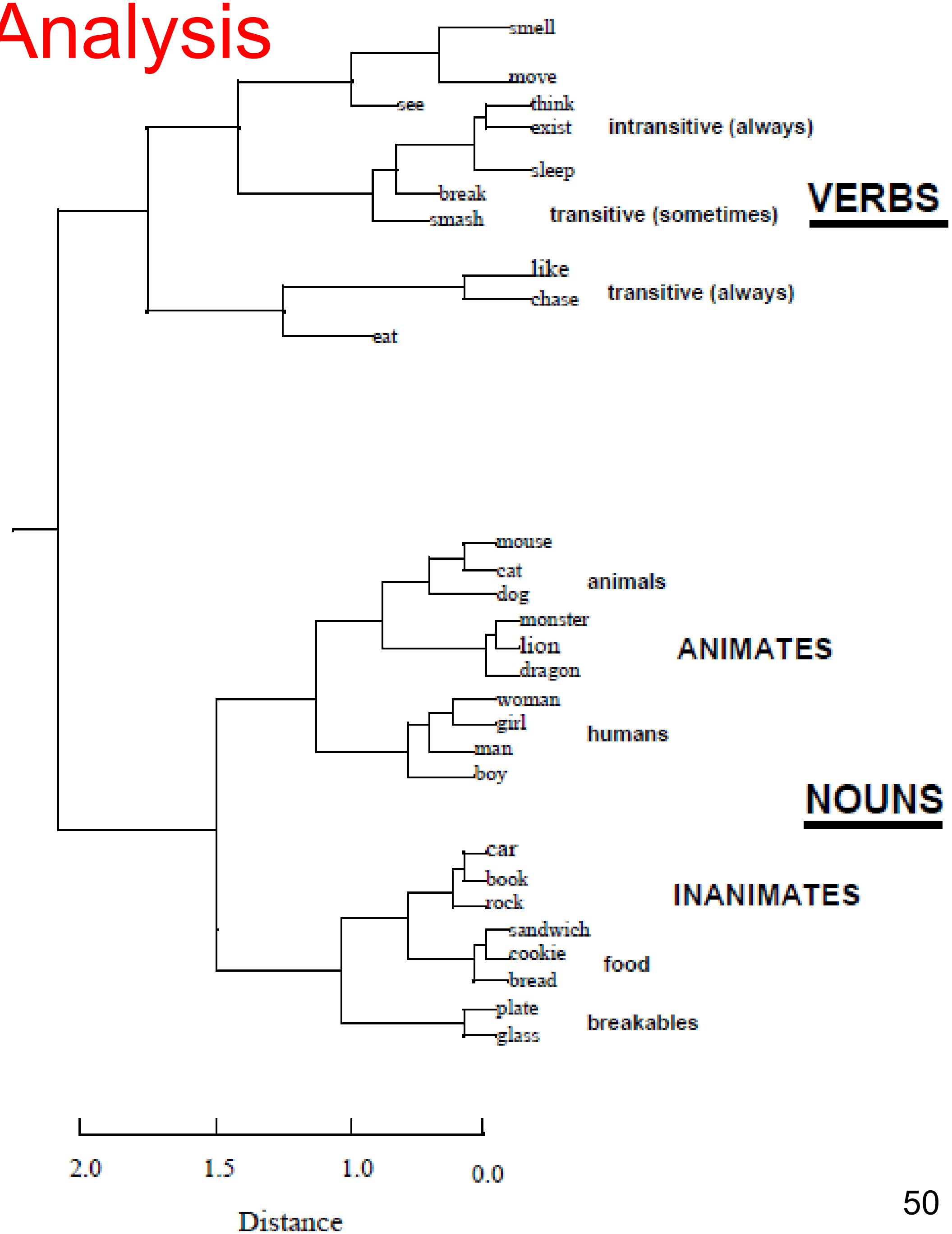
INPUT		OUTPUT	
0000000000000000000000000000010	(woman)	0000000000000000000000000010000	(smash)
0000000000000000000000000010000	(smash)	0000000000000000000001000000000	(plate)
0000000000000000000001000000000	(plate)	0000010000000000000000000000000	(cat)
0000010000000000000000000000000	(cat)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000000000100000000000000	(man)
0000000000000000100000000000000	(man)	0001000000000000000000000000000	(break)
0001000000000000000000000000000	(break)	0000100000000000000000000000000	(car)
0000100000000000000000000000000	(car)	0100000000000000000000000000000	(boy)
0100000000000000000000000000000	(boy)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000001000000000000000000	(girl)
0000000000001000000000000000000	(girl)	0000000000100000000000000000000	(eat)
0000000000100000000000000000000	(eat)	0010000000000000000000000000000	(bread)
0010000000000000000000000000000	(bread)	0000000010000000000000000000000	(dog)
0000000010000000000000000000000	(dog)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000000000001000000000000	(mouse)
0000000000000000001000000000000	(mouse)	0000000000000000001000000000000	(mouse)
0000000000000000001000000000000	(mouse)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	1000000000000000000000000000000	(book)
1000000000000000000000000000000	(book)	0000000000000001000000000000000	(lion

SRN for Letter Sequences



31 input units, 150 hidden units,
31 output units, and 150 context units

Hierarchical Cluster Analysis of Hidden Layers



A CASE Study: Real-world Semantic Sequence Classification with Recurrent Networks

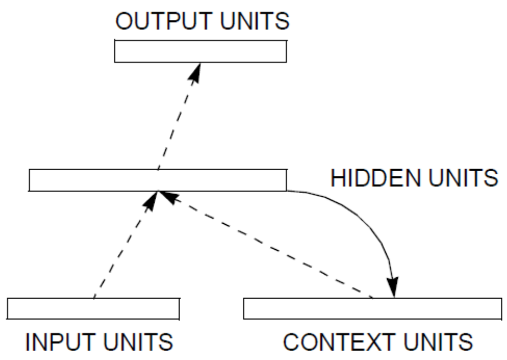
- How can we learn to classify sequences?
- Classification belongs to most important human tasks
- Neural agent for Reuters news corpus
- News titles belong to one of eight main categories, e.g. money, energy, shipping, interest, economic, currency, corporate, commodity
- Corpus size: 82,339 words
- Number of different words in titles 11,104


Bundesarchiv, B 145 Bild-F079071-0007
Foto: o.Ang. | Juni 1988

Neural News Routing Architectures

- Various hybrid architectures containing:
  - pre-processing from statistics and information retrieval
  - Different forms of simple recurrent networks and extensions (recurrent plausibility networks)
- Partially recurrent connections for ***incremental context*** of an ***unrestricted phrase***
- Titles presented as sequence of:
  - word input representations
  - category output representations
  - one pair for each word

Training and Evaluation of Neural News Routing



- At beginning of title, context layers initialized to *0*
- Each unit in output layer corresponds to particular semantic category
- Output units representing desired semantic categories set to *1*
- All other output units set to *0*
- Define as ***"classified correctly"*** when at end of sequence, value for output unit for desired category is > *0.5*

Neural News Routing (3)

- Average recall and precision values for each semantic category, and overall training and test sets
- Training regime forces network to assign desired category at beginning of title
- Training stopped when error over training set stops decreasing or use of validation set
- Typically, 700-900 epochs through whole training corpus

How to Get Appropriate Input Representations?

- Initial use of significance vectors to represent words
- Determined by frequency of words in different semantic categories

$$v(w, c_i) = \frac{Frequency\ of\ w\ in\ c_i}{\sum_j Frequency\ for\ w\ in\ c_j}\quad for\ j \in \{1, \ldots n\}$$

Each word w represented by vector $(c_1 c_2 \cdots c_n)$

c_i represents a certain semantic category

Examples of Significance Vectors

Word	MF	SH	IN	EC	CR	CO	CM	EN
a	,07	,02	,04	,17	,04	,28	,27	,10
and	,06	,02	,03	,15	,03	,25	,34	,09
bureau	,02	,00	,00	,38	,02	,02	,56	,01
mortgage	,01	,00	,30	,18	,00	,51	,00	,00
of	,07	,02	,03	,14	,03	,28	,31	,09
parker	,13	,00	,00	,00	,13	,73	,00	,00
the	,09	,03	,05	,18	,05	,20	,31	,09

MF: Money-fx
SH: Ship
IN: Interest
EC: Economic
CR: Currency
CO: Corporate
CM: Commodity
EN: Energy

Results using Significance Vectors

Category	Training Set		Test Titles	
	Recall	Precision	Recall	Precision
money-fx	84,36	84,62	84,74	69,56
ship	81,06	93,17	77,34	94,96
interest	77,93	82,71	85,42	83,45
economic	71,70	80,79	74,74	77,82
currency	85,75	91,46	85,36	87,16
corporate	88,81	92,31	94,87	95,10
commodity	86,27	94,77	86,47	88,31
energy	81,88	92,26	85,22	91,65
Total	85,15	86,99	**91,23**	**90,73**

Normalized Significance Vectors are better

- Significance Vectors represent plausibility of particular word occurring in a semantic category
- Should be Independent of number of examples observed in each category:

$$v(w, c_i) = \frac{Norm.\ freq.\ of\ w\ in\ c_i}{\sum_j Norm.\ freq.\ for\ w\ in\ c_j}\ for\ j \in \{1, \ldots n\}$$

where:

$$Norm.\ freq.\ of\ w\ in\ c_i = \frac{Freq.\ of\ w\ in\ c_i}{Number\ of\ titles\ in\ c_i}$$

- We call normalized significant vectors ***Semantic Vectors***

Why are Semantic Vectors (Normalised Significance Vectors) better?

Word	MF	SH	IN	EC	CR	CO	CM	EN
a	,13	,12	,11	,16	,16	,06	,11	,15
agency	,07	,23	,04	,17	,09	,03	,12	,25
and	,12	,12	,09	,16	,14	,05	,15	,15
bureau	,00	,00	,00	,50	,11	,00	,32	,01
money	,33	,01	,40	,14	,10	,00	,01	,00
mortgage	,02	,00	,72	,15	,02	,09	,00	,00
of	,13	,11	,11	,15	,15	,06	,14	,14
parker	,25	,00	,00	,00	,56	,17	,00	,00
the	,15	,13	,12	,15	,18	,04	,11	,12

Mortgage (Hypothek) has high semantic value for interest (Zinsen)

Even distribution for stop words

Some Improvement Using Semantic Vector Input

Category	Training Set		Test Titles	
	Recall	Precision	Recall	Precision
money-fx	87.78	88.60	84.07	69.59
ship	81.65	88.13	82.73	93.88
interest	85.33	86.97	88.25	88.19
economic	76.22	83.54	78.36	80.30
currency	87.76	92.59	89.64	89.86
corporate	89.16	91.96	95.90	95.98
commodity	86.27	90.52	86.20	87.22
energy	89.19	95.48	86.58	91.56
Total	88.57	88.59	**92.47**	**91.61**

Removing Insignificant Words

- Pre-processing strategy from information retrieval
- Emphasize significant ***domain-dependent*** words
- ***Remove insignificant stop words***
- Frequent, domain-independent words: determiners, prepositions & conjunctions
- 19 insignificant words removed using semantic vector representation

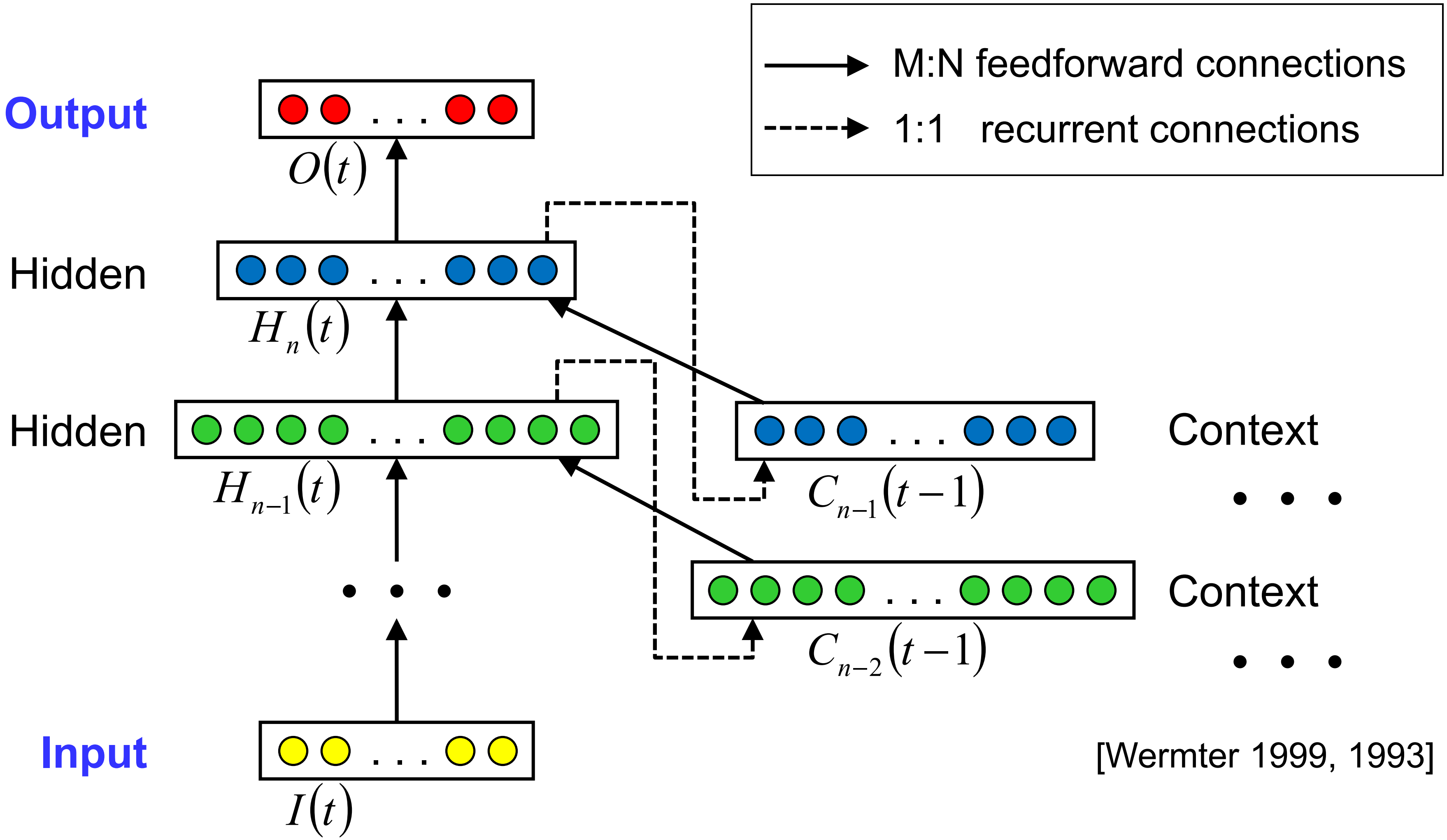
Stop Word Elimination Shows some Limited Improvement

Category	Training Set		Test Titles	
	Recall	Precision	Recall	Precision
money-fx	87.40	89.12	84.97	69.36
ship	79.14	88.85	78.42	93.53
interest	86.61	93.09	88.89	88.77
economic	79.12	87.80	79.28	82.64
currency	85.55	92.59	87.39	87.16
corporate	89.51	93.71	96.47	96.48
commodity	87.25	91.18	86.74	86.51
energy	85.32	94.19	83.86	90.54
Total	88.47	90.05	**92.88**	**91.92**

Can we do better if we give the Network more Sequential Memory? (Plausibility Networks)

- A plausibility networks is an extension of a simple recurrent network – adding more memory as hidden layers
- Recurrent plausibility network with 2 hidden & 2 context layers
- All other parameters same as experiment with semantic vector representation
- Some further improvement, especially for longer titles due to additional memory

Recurrent Plausibility Networks (RPN)



[Wermter 1999, 1993]

Results of Recurrent Plausibility Networks with Semantic Vectors (best results)

Category	Training Set		Test Titles	
	Recall	Precision	Recall	Precision
money-fx	87.34	89.47	86.03	76.70
ship	84.65	89.21	82.37	90.29
interest	85.24	87.77	88.19	86.05
economic	90.24	91.77	81.89	83.80
currency	88.89	91.36	89.64	89.86
corporate	92.31	92.66	95.55	95.43
commodity	92.81	93.14	88.84	90.29
energy	85.27	87.74	87.69	92.95
Total	89.05	90.24	**93.05**	**92.29**

Examples: Output Preferences for Sequences and Multiple Classes

.11	.22	.33	.44	.56	.67	.78	.89	1.00

Example (1)	MF	SH	IN	EC	CR	CO	CM	EN
BANK	.65		.48	.19	.36			
OF	.58		.40	.20	.30			
JAPAN	.74		.21	.23	.60			
INTERVENES	.97				.97			
SHORTLY	.98				.98			
AFTER	.96				.95			
TOKYO	.97				.96			
OPENS	.96				.95			

MF: Money-fx
IN: Interest
EC: Economic
CR: Currency

Output Representations over Sequences

.11	.22	.33	.44	.56	.67	.78	.89	1.00

Example (2)	MF	SH	IN	EC	CR	CO	CM	EN
BANK	.65		.48	.19	.36			
OF	.58		.40	.20	.30			
JAPAN	.74		.21	.23	.60			
DETERMINED	.15		.56			.18		
TO	.26		.50		.14			
KEEP	.22		.31					
EASY			.94					
MONEY	.17		.92					
POLICY	.43		.90					

MF: Money-fx
IN: Interest
EC: Economic
CR: Currency
CO: Corporate

Effects of Keywords on Classification

.11	.22	.33	.44	.56	.67	.78	.89	1.00

Example (3)	MF	SH	IN	EC	CR	CO	CM	EN
IRAN		.72				.16		.57
SOVIET		.90					.22	.35
UNION		.97					.14	.20
TO		.95					.20	.19
SWAP		.92					.34	.26
CRUDE		.69						.84
REFINED		.23						.98
PRODUCTS		.15						.96

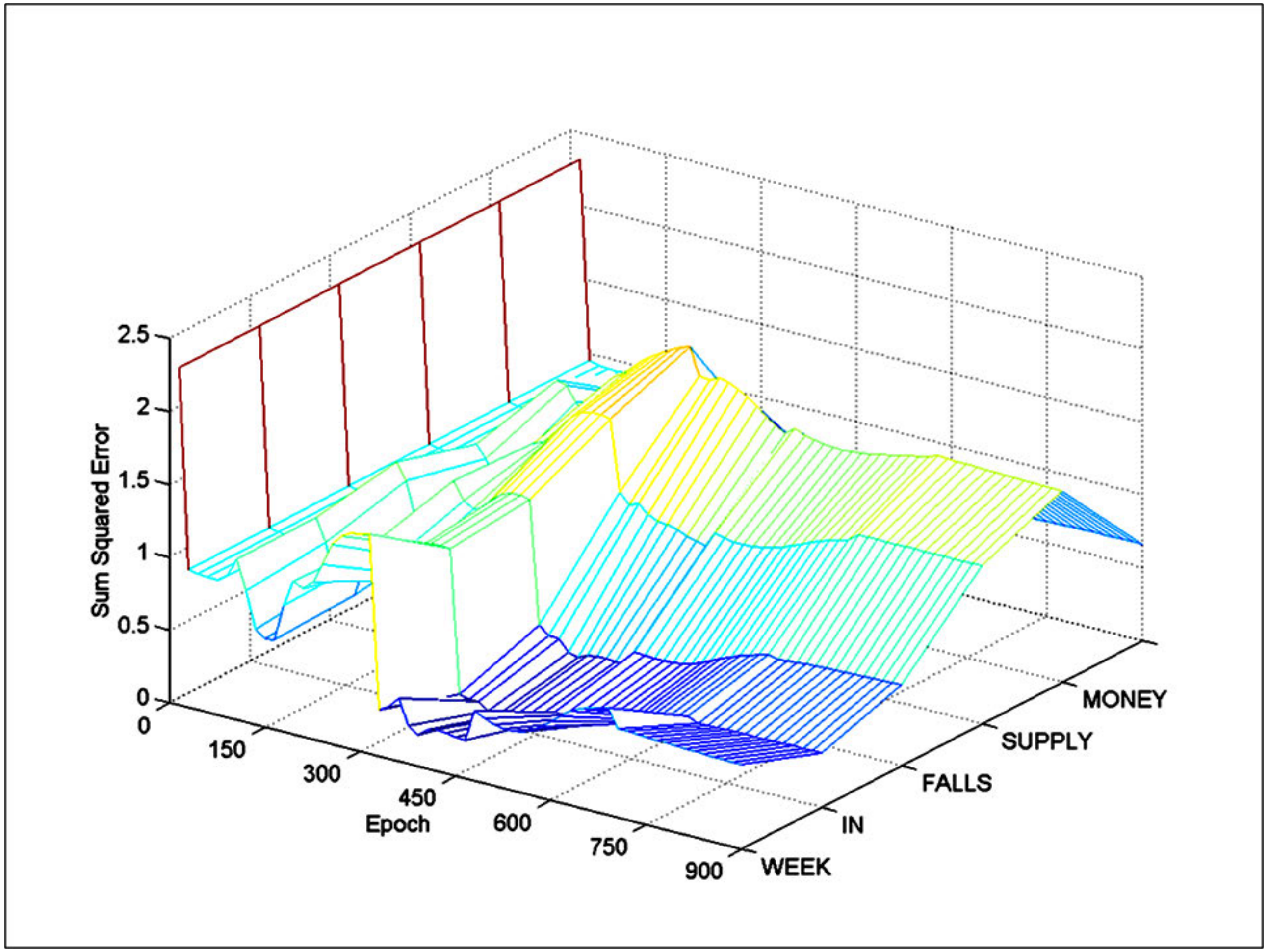
SH: Ship
CM: Commodity
CO: Corporate
EN: Energy

Sequence Learning Diagrams to show how the Network learns (1)

- Sequence learning diagrams to display performance of network learning over time for single sequences
- Based on ***sum squared error*** for <u>each sequence</u> for <u>each epoch</u>
- For analyzing sequences and learning in detail

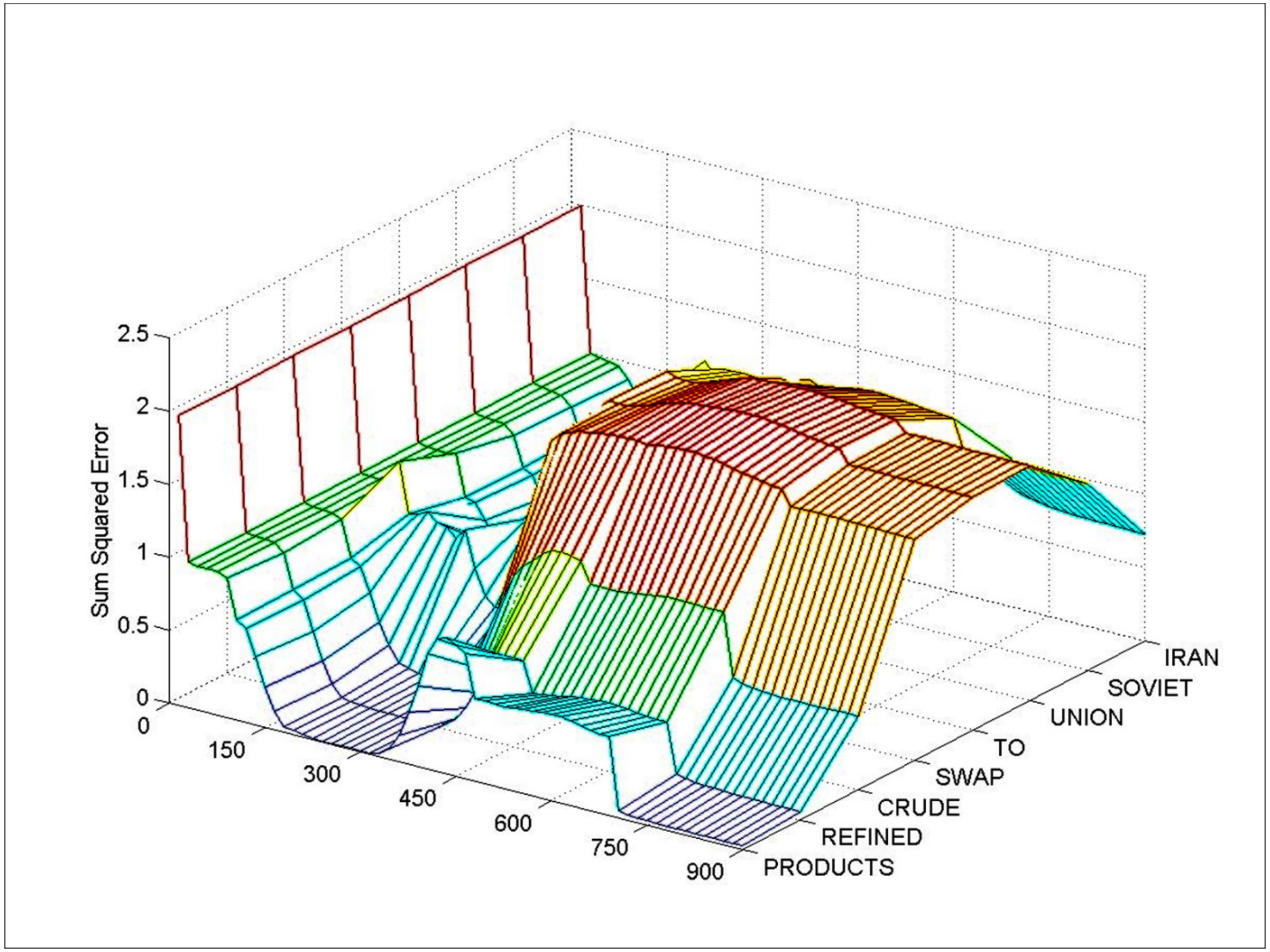
Sequence Learning Diagrams (2)

"Canadian money supply falls in week"



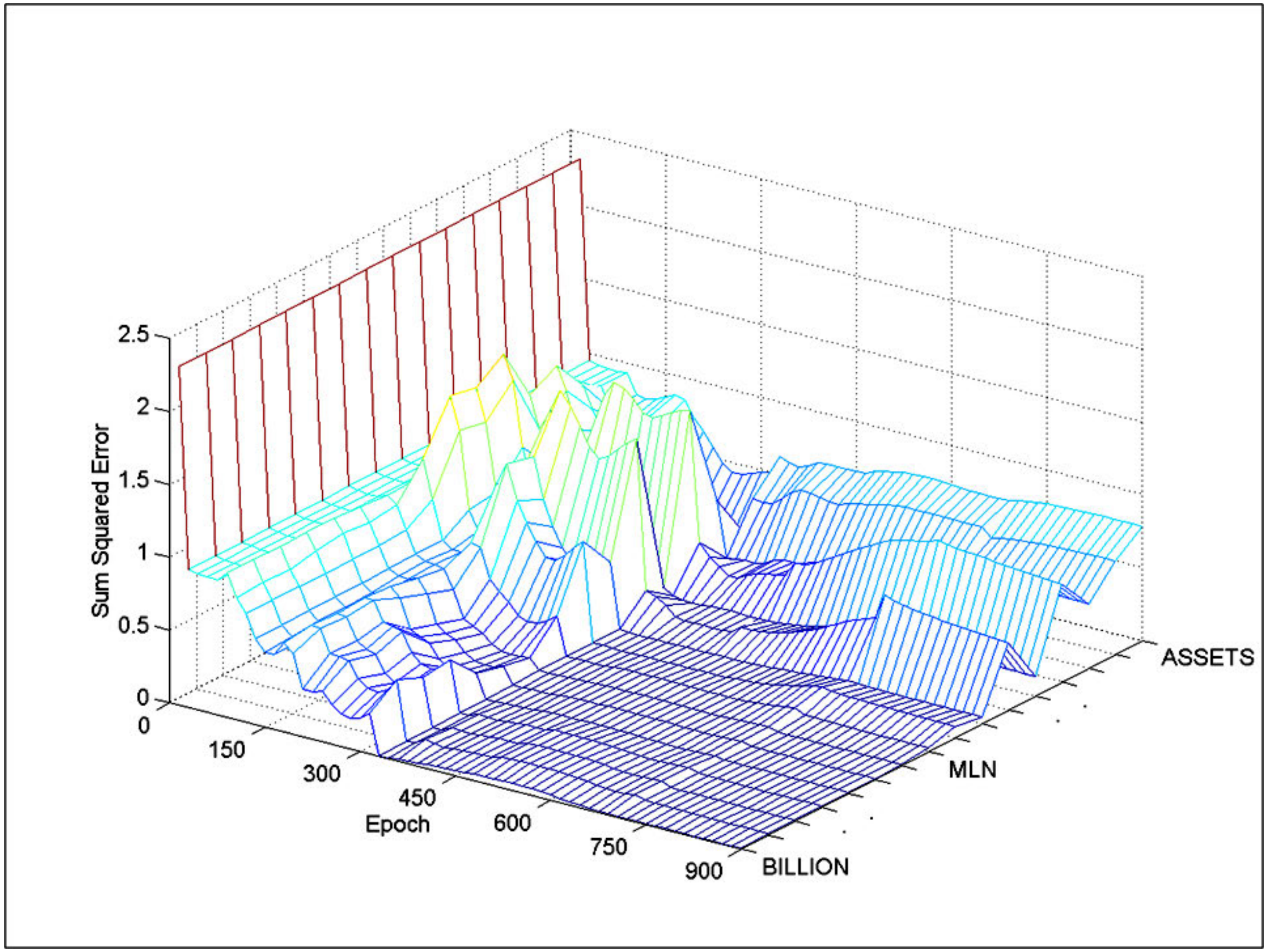
Sequence Learning Diagrams (2)

“Iran, Soviet Union to swap crude, refined products”



Sequence Learning Diagrams (3)

"Assets of money market mutual funds fell 35.3 mln dlrs in latest week to 237.43 billion"



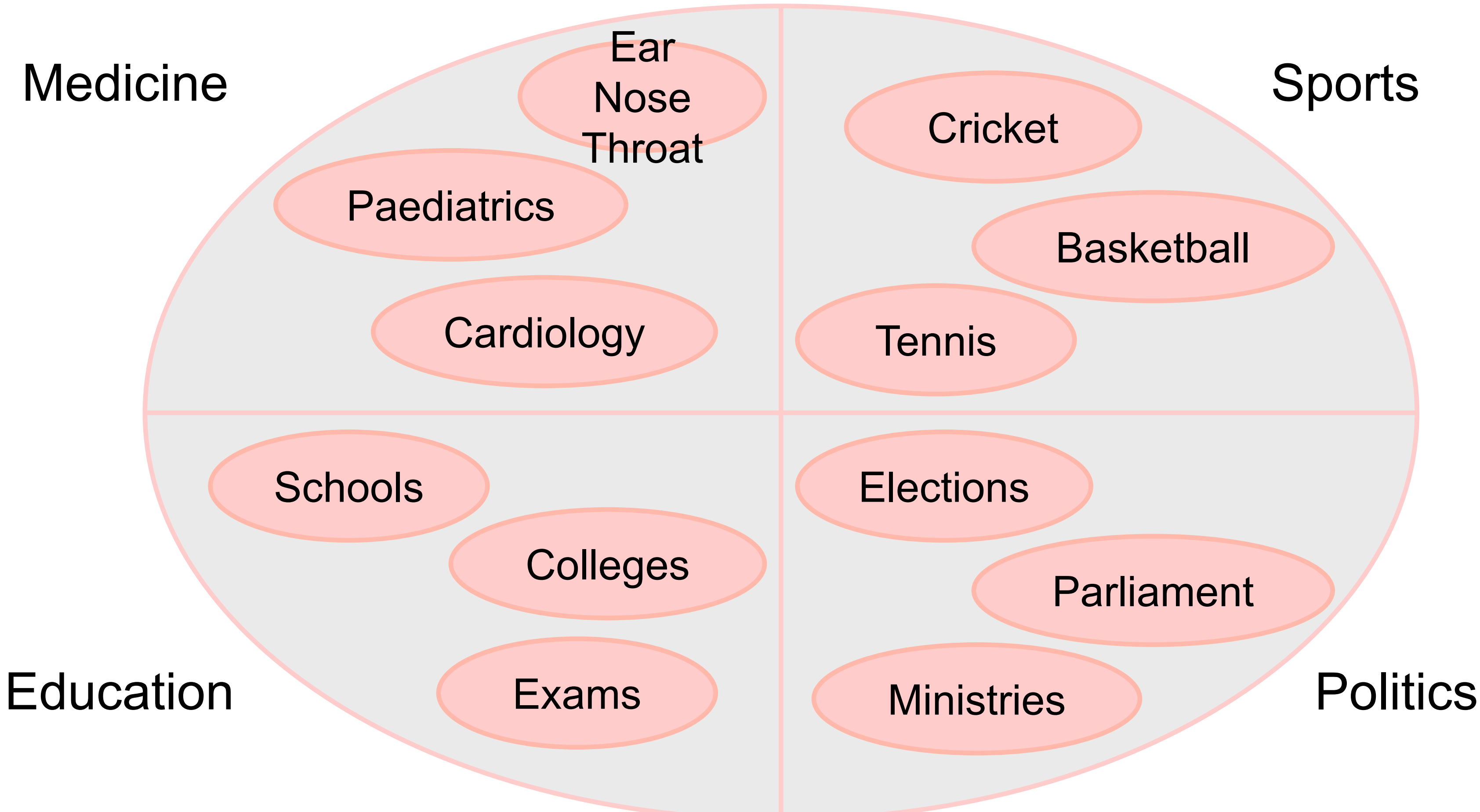
Current Research: Subspace Learning

- Need methods to point directly to main level 1 topic *without using any classification algorithm*
- This will increase search/categorization speeds
- Solution – *Subspace Learning*
- Hybrid parallel classifiers based on *semantic data subspaces* to improve two-level categorization of text documents
- Each *subspace can be handled separately* with a different classifier using only reduced dimensions.

Data Domains as Subspaces

- The web contains many *broad domains of data* which are quite distinct from each other e.g. medicine, education, sports and politics
- Each of these domains constitutes a subspace of the data within which the documents are *similar to each other* but quite distinct from the documents in another subspace
- The data within these domains is frequently further divided into many *subcategories*

Document Subspace Clusters



Document Vector Representation

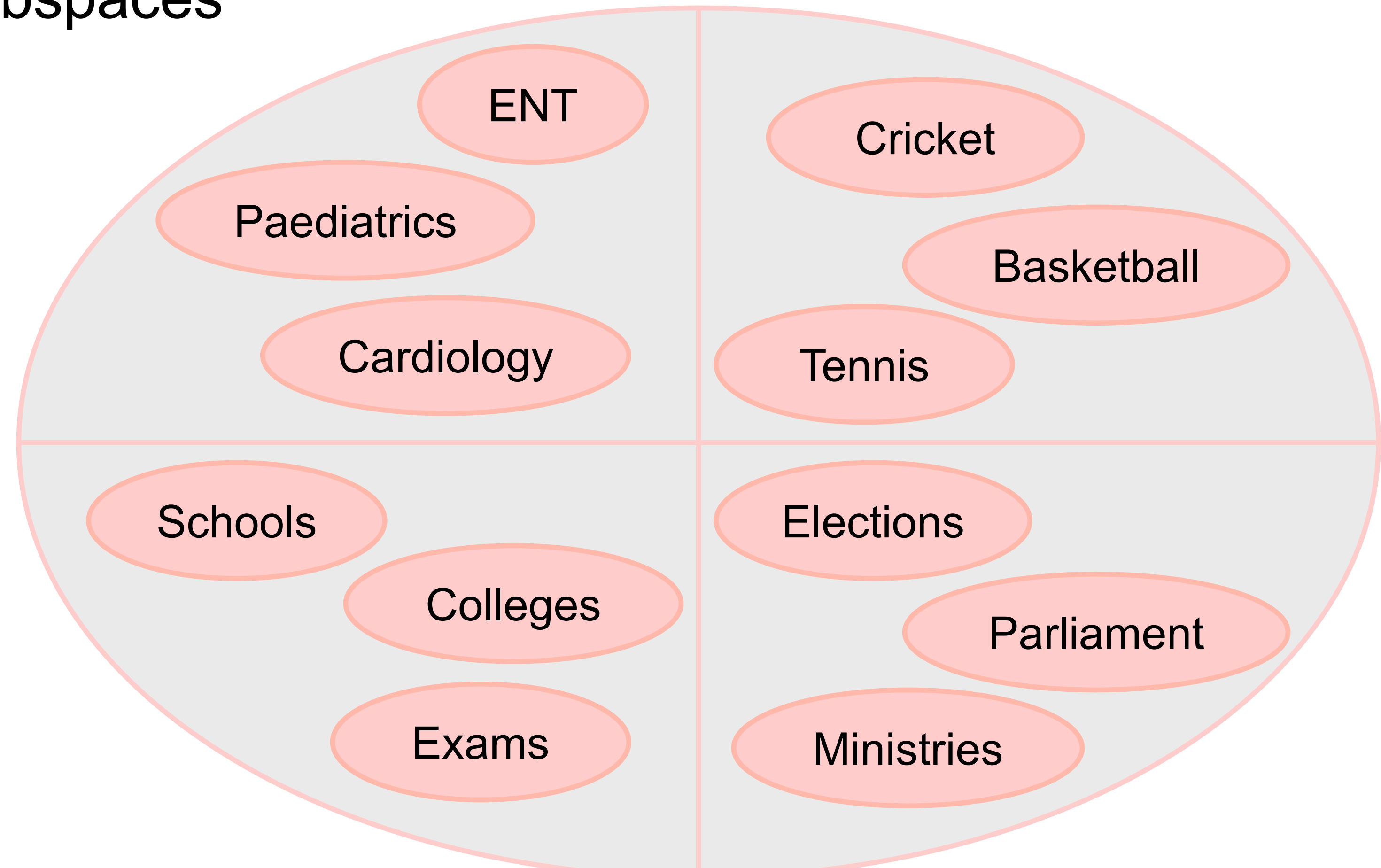
- D=(∑ word significance vectors)/p

  where p=number of words in the document

- Two variations:
  - Document Full Significance Vector (FSV)
  - Document Conditional Significance Vector (CSV)

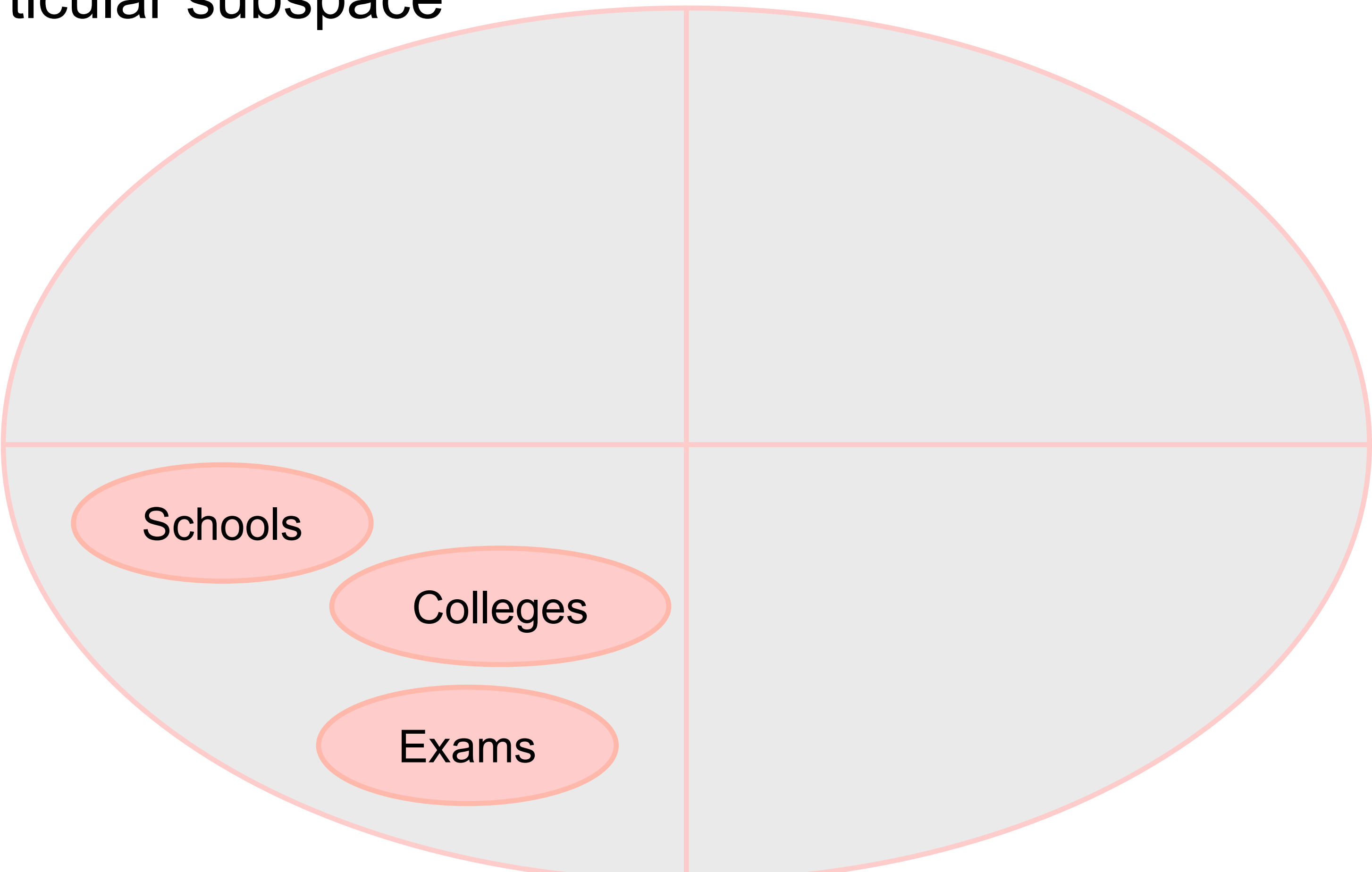
Full Significance

- Considers occurrence of a word in all subtopics of all subspaces



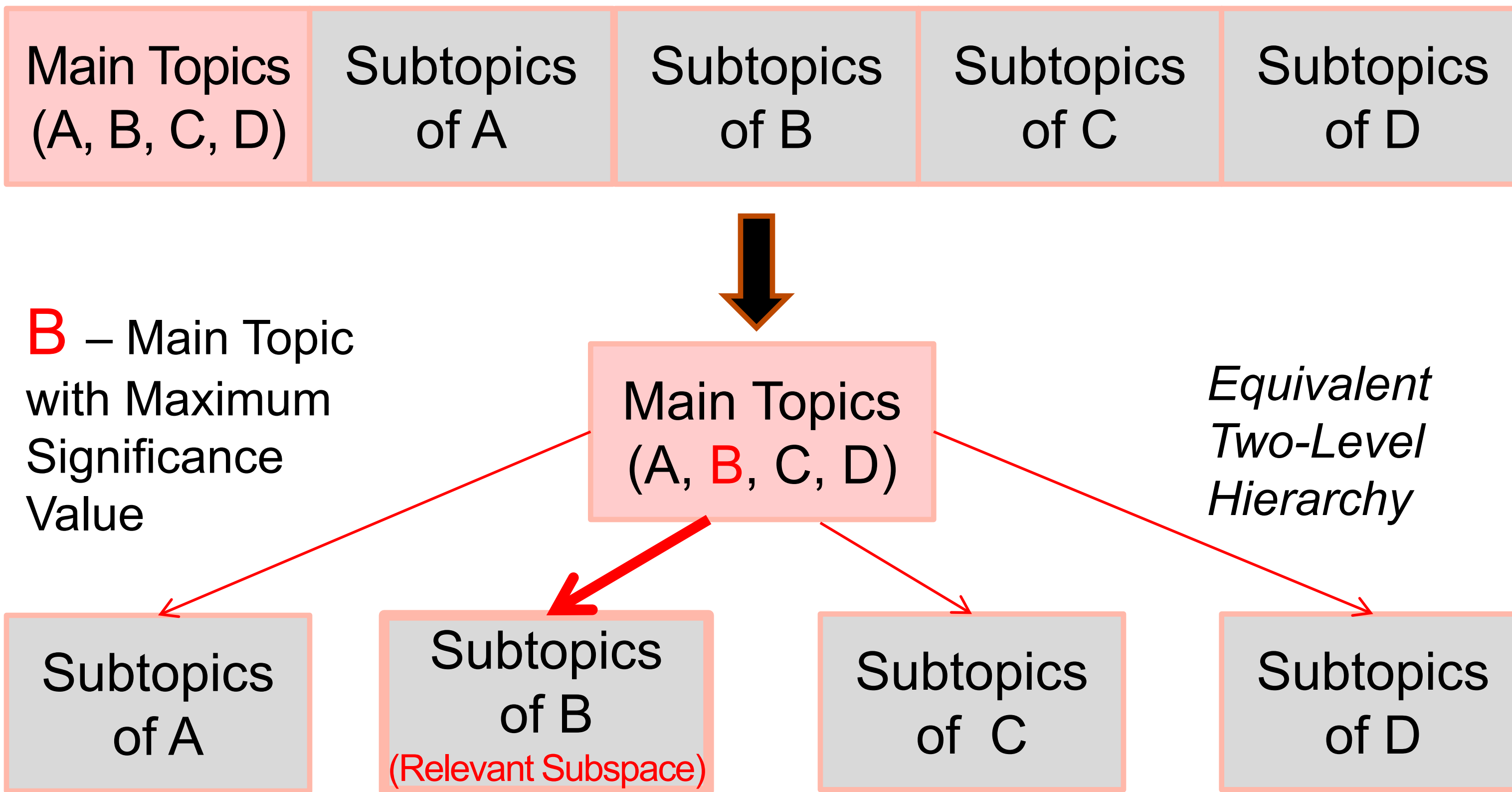
Conditional Significance

- Considers occurrence of a word in all subtopics of a particular subspace

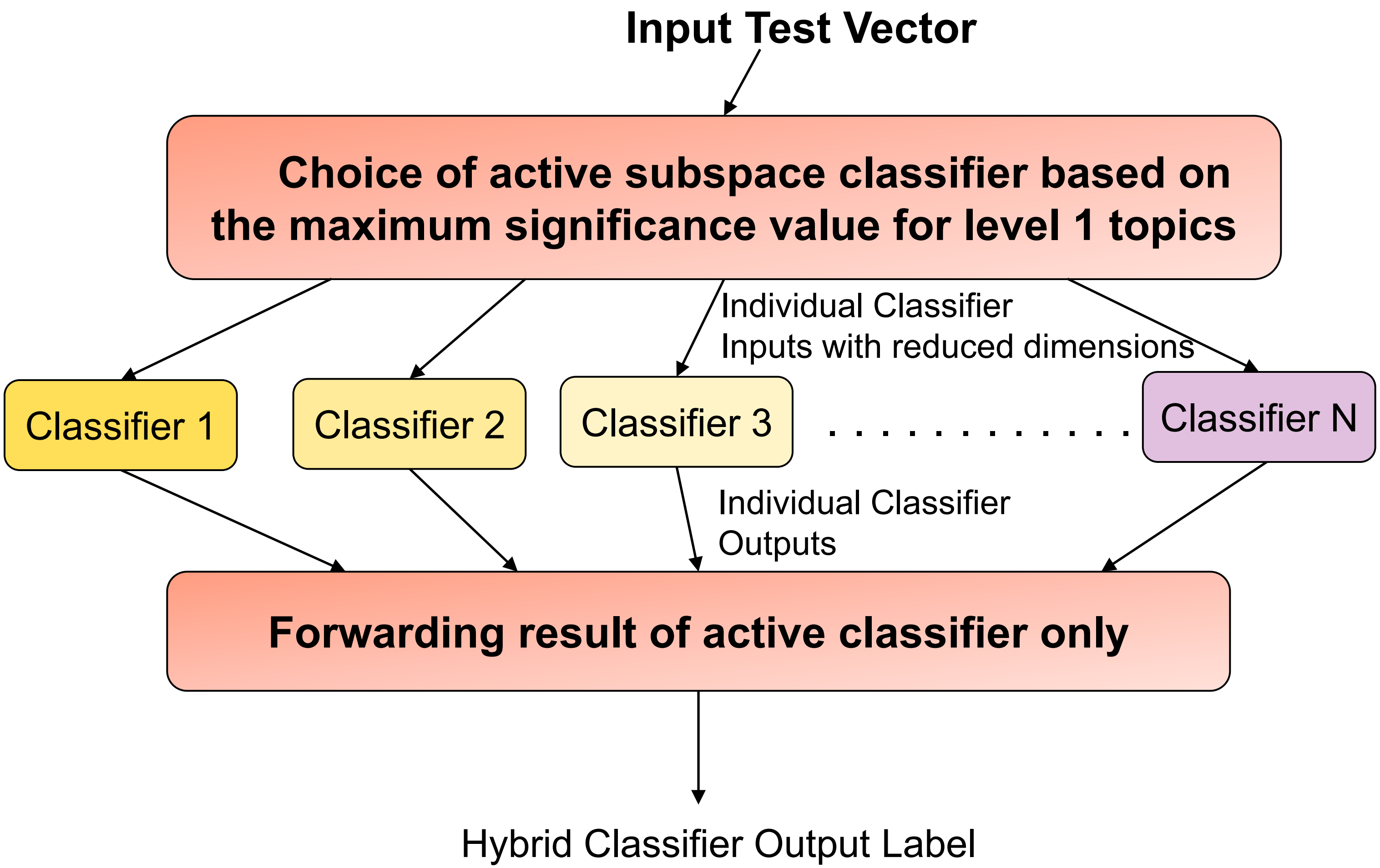


Conditional Significance Vector

Component Blocks of the Flat Conditional Significance Vector



Hybrid Parallel Classifier Architecture



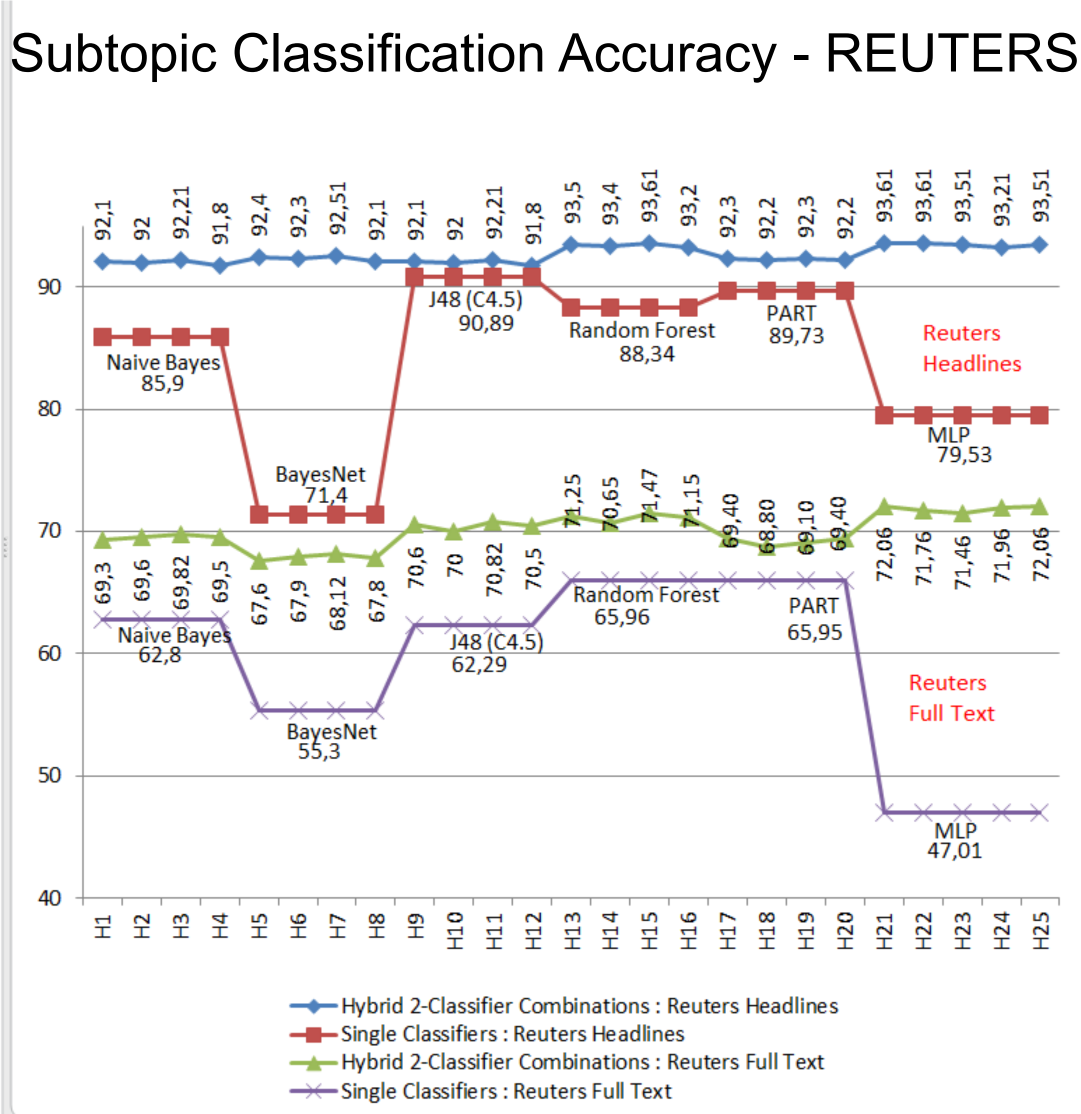
[Tripathi, Oakes, Wermter 2012, 2013]

CASE study for Hierarchical Classification

- Reuters RCV1 News Corpus
  - Reuters Headlines
  - Reuters Full Text (Headlines + Body Text)
  - 4 main topics, 50 subtopics

- LSHTC (Large Scale Hierarchical Text Classification) Corpus
  - From LSHTC challenge associated with the European Conference on Information Retrieval (ECIR), 2010
  - derived from the ODP (Open Directory Project) directory
  - data in the form of content vectors
  - 10 main topics, 158 subtopics

CASE study for Hierarchical Classification (2)



Hybrid 2-Classifier Combinations:

H1-NB/J48
H2-NB/RF
H3-NB/MLP
H4-NB/PART

H5-BN/J48
H6-BN/RF
H7-BN/MLP
H8-BN/PART

H9-J48/NB
H10-J48/BN
H11-J48/MLP
H12-J48/PART

H13-RF/NB
H14-RF/BN
H15-RF/MLP
H16-RF/PART

H17-PART/NB
H18-PART/BN
H19-PART/J48
H20-PART/RF

H21-MLP/NB
H22-MLP/J48
H23-MLP/BN
H24-MLP/PART
H25-MLP/RF

Summary

- Associative Memory and Radial Basis Networks
  - **Approximate** and learn **Continuous** Functions
  - Multiple layer for **nonlinearity**
- Recurrent Neural Networks for Prediction and Classification:
  - ***Efficiently applicable*** to
    - Sequence and stock market prediction
    - Handwriting and speech recognition
    - Attentive vision and keywords spotting... *and much more!*
  - Method to problem solving ***using*** some ***context***
  - ... that is still ***deterministic*** and can be analyzed
- Further reading:
  - Rojas R. Introduction to Neural Networks. Berlin: Springer, 1996. (Available online)
  - Marsland. Machine Learning. 2009, chapter 4