


Machine Learning - Lecture # 2

Bruce M. McLaren
Senior Researcher, DFKI

Slides courtesy of
Erica Melis, Russell/Norvig

Overview

Ensemble Learning

Neural Networks

- Brains
- Neural Networks
- Perceptrons
- Multi-layer Perceptrons
- Support Vector Machines
- Applications of Neural Networks

Learning from Text

Overview of a DFKI Machine Learning Project

Ensemble Learning

- Last lecture: We chose a single hypothesis to make a prediction
- Imagine: Running multiple decision tree learning algorithms in parallel, perhaps with different parameter settings
- Collect resulting, individual hypotheses in an ***ensemble*** and combine their predictions appropriately

Formalization and Motivation

Let $h_1, h_2, ..., h_n$ be the set of individual hypotheses and let e be an example.

Typical voting schemes for ensembles:

- Unanimous vote:

  $h(e) = 1$ iff $\sum h_k(e) = n$

- Majority vote:

  $h(e) = 1$ iff $\sum h_k(e) > n/2$, i.e if the majority is positive

- Weighted majority vote:

  $h(e) = 1$ iff $\sum w_k h_k(e) > \sum w_k(1 - h_k(e))$, where each hypothesis h_k has a weighting factor w_k

Motivation - An Example (1)

Advantage

To improve the quality of the overall hypothesis, for example:

- Collect the hypotheses of 5 learners in an ensemble
- Combine their hypotheses using a simple majority vote; to misclassify a new example, at least 3 out of 5 hypotheses have to misclassify it

Improvement under the assumptions

- Suppose each hypothesis h_k in the ensemble has an error of p; i.e. the probability that a randomly chosen example is misclassified by h_k is p
- Furthermore, suppose the errors made by the individual hypotheses are independent

$$p_M = (5/3)p^3(1-p)^2 + (5/4)p^4(1-p) + (5/5)p^5$$

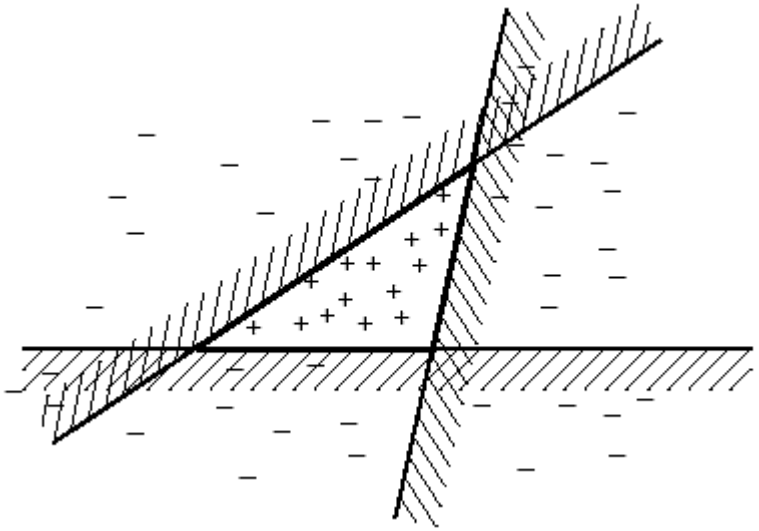
i.e. 1 in 100 instead of 1 in 10 !!

Motivation (2)

Advantage: enlarging the hypothesis space, expressive power

- an ensemble itself constitutes a hypothesis
- the new hypothesis space is the set of all possible ensembles constructible from hypotheses in the original hypothesis space of the individual learning algorithms
  - Example:

Three linear threshold hypotheses. Together, they create a hypothesis not expresible in the original space.



A Type of Ensemble Learning: Boosting

Improves the quality of the ensemble by focusing on and „boosting“ accuracy

Basics:

- A weighted training set; i.e. each training example has an associated weight
- The method respects the weights of the training examples, i.e. the higher the weight of an example the higher its importance during the learning phase

General Boosting Algorithm

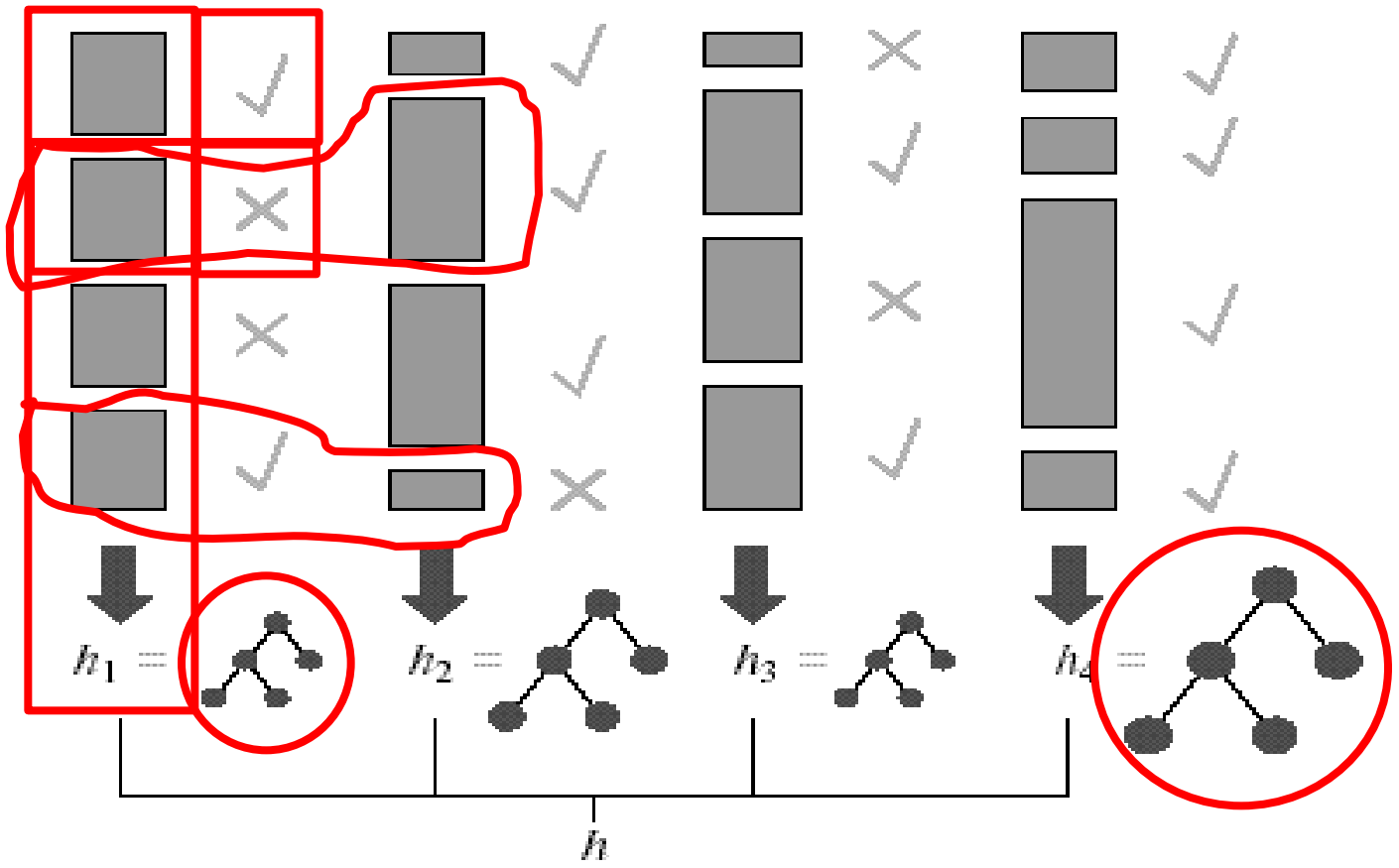
Initially, each training example has the fixed weight 1

The first round of learning – using learning method L – starts

n-th Round ($n \leq M$)

- run L on the given weighted training examples to generate hypothesis h_n
- change the weight of the training examples as follows:
  - decrease the weight, if the example is correctly classified by h_n
  - increase it, otherwise
- start the next round of learning

Intuition of Boosting



Intuition: As example weights are altered, we hope to improve our hypotheses and then give the greatest weight to the hypotheses in the ensemble that perform the best on the training data.

Boosting: Summary

- There are many variants of boosting with different ways of adjusting the weights and combining the hypotheses

- Some have very interesting properties
  e.g. AdaBoost; even if the learning method L is *weak* AdaBoost will return a hypothesis that classifies the training data perfectly, provided M is large enough

  weak?

  L always returns a hypothesis with a weighted error on the training set that is slightly better than random guessing

- See figure on page 667 of your book -- AdaBoost applied to decision stump (a single-node decision tree) is shown to improve performance on test set, as compared to decision stump without boosting

Overview

Ensemble Learning

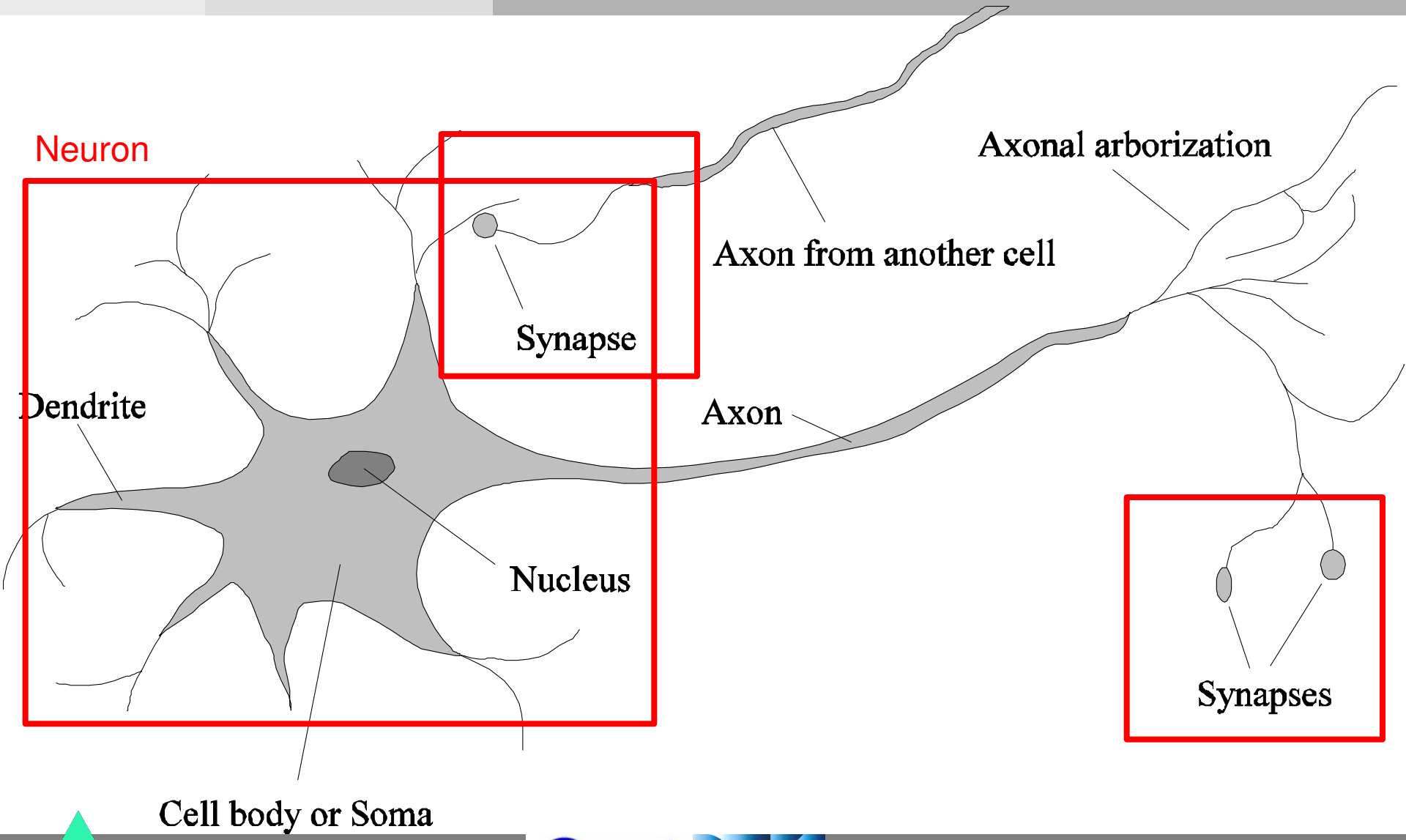
Neural Networks

- Brains
- Neural Networks
- Perceptrons
- Multi-layer Perceptrons
- Support Vector Machines
- Applications of Neural Networks

Learning from Text

Overview of a DFKI Machine Learning Project

Human Brain, Neuron



Human Brain - The Raw Numbers

10^{11} neurons

> 20 types of neurons

10^{14} synapses

1ms-10ms cycle time

From this ⇨ Scene recognition time ~ 0.1 second

Neural Networks - Mapping the Brain to the Computer

The function of the brain led AI researchers to investigate whether artificial neural networks could simulate brain activity and, thus, achieve “intelligence” and “learning”

Created field called Neural Networks (or Connectionism… or Parallel Distributed Processing …)

- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Highly parallel, distributed process
- Emphasis on tuning weights automatically

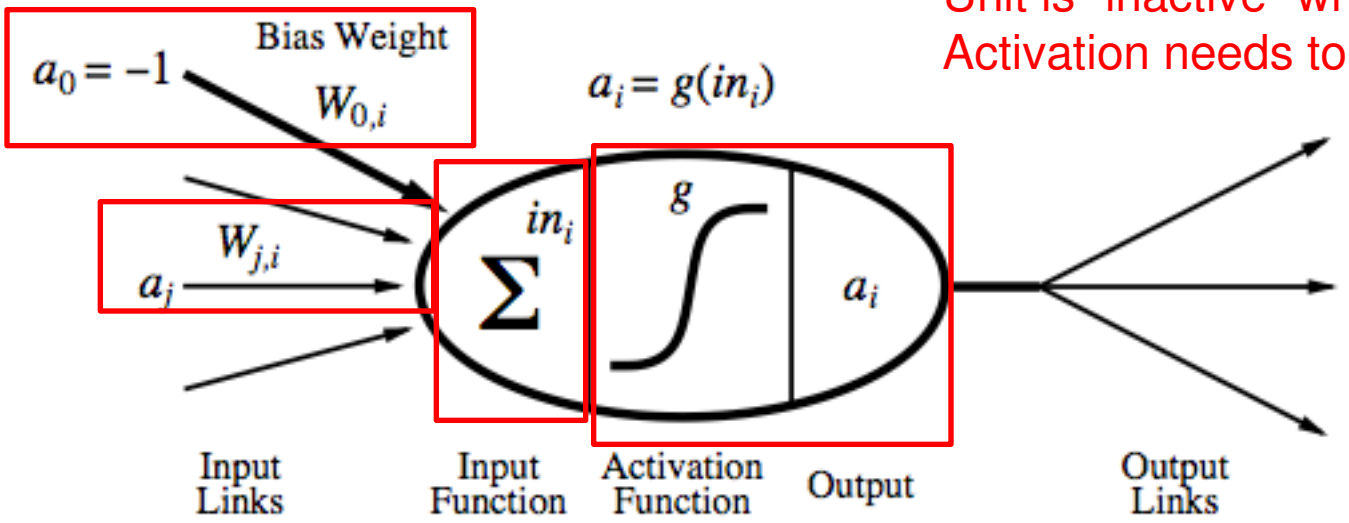
This is the key to NN machine learning

McCulloch-Pitts "Unit"

- Output is a linear function of units:

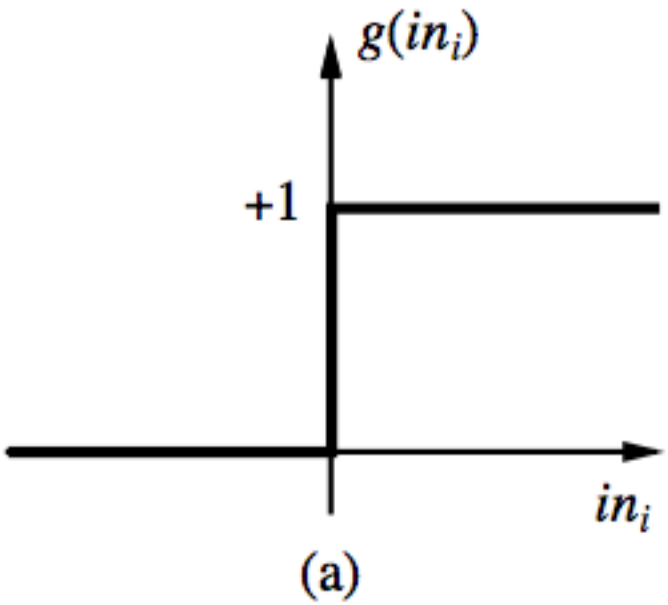
$$a_i \leftarrow g(in_i) = g\left(\sum_j W_{j,i} a_j\right)$$

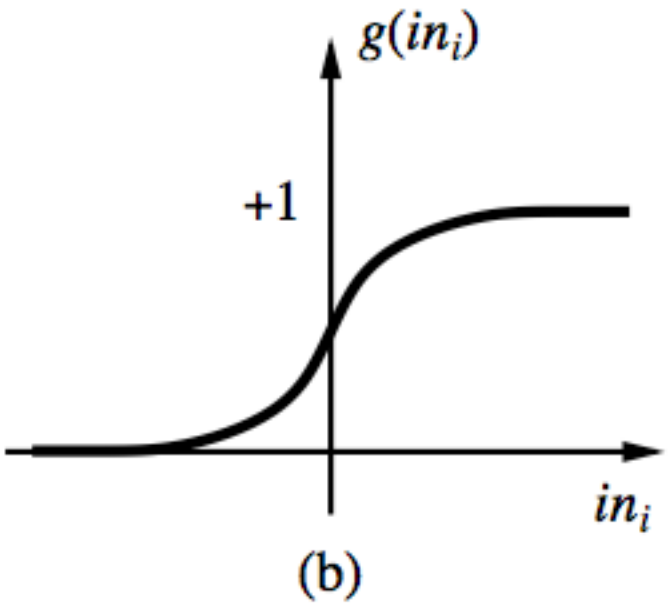
Unit is "active" when g ~= +1
Unit is "inactive" when g ~= 0
Activation needs to be nonlinear!



- This is a gross over-simplification of neurons, but its purpose is to develop understanding of what simple units can do

Activation Functions

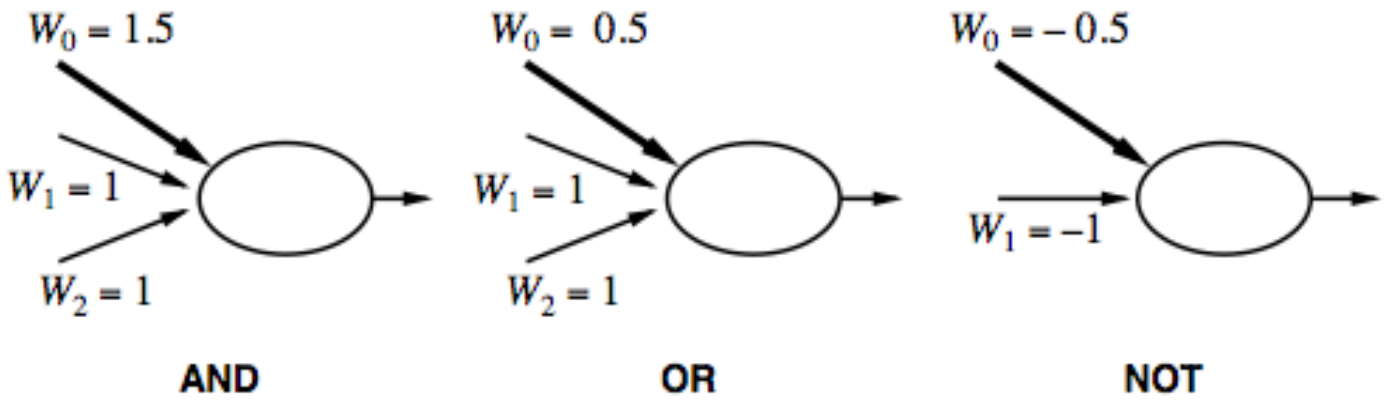




- is a step function or threshold function
- is a sigmoid function $1 / (1 + e^{-x})$

Changing the bias weight $W_{0,i}$ moves the threshold location

Implementing Logical Functions



-1 (1.5)

0 (1) → 0.5

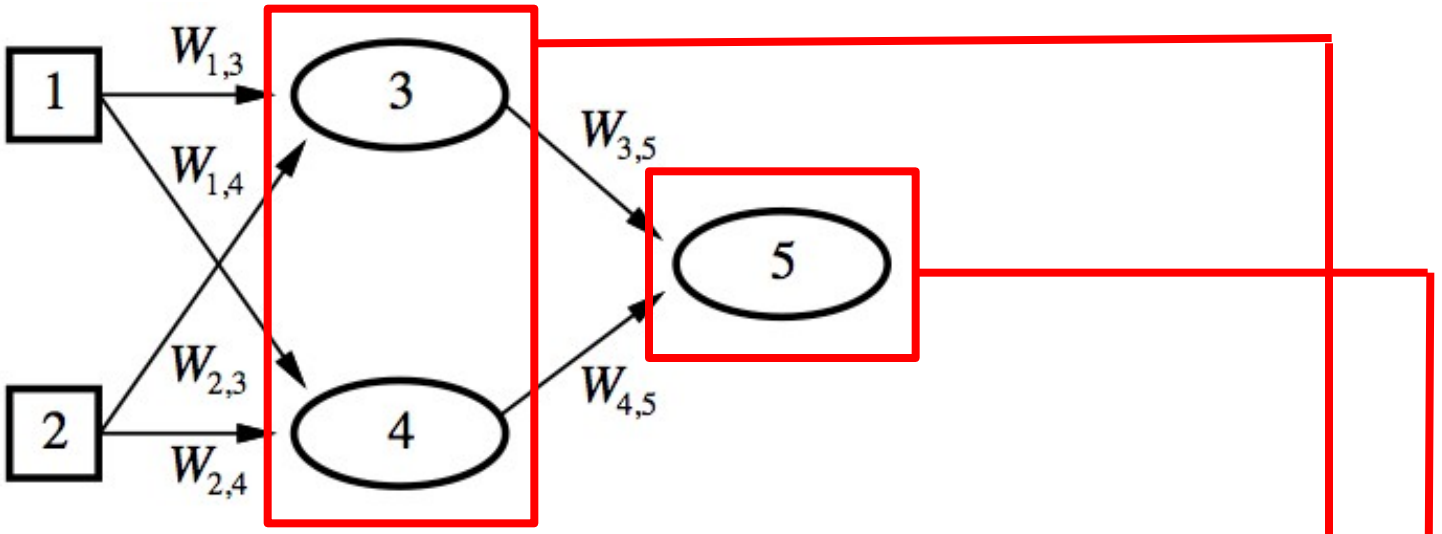
0 (1)

- McCulloch and Pitts: Every boolean function can be implemented

Network Structures

- Feed-forward networks:
  - Acyclic
  - Implements a function of its current inputs, no internal state
  - Single- and multi-layer perceptrons
- Recurrent networks:
  - Cyclic, feeds its output back into its inputs; can be chaotic
  - Depends on its initial state, can support “short-term memory”
  - More interesting as models of the brain, but much more complicated
  - A couple of quick examples …
    - Hopfield networks have symmetric weights ($W_{i,j} = W_{j,i}$)
      - Holographic associative memory
    - Boltzmann machine use stochastic activation functions

Feed-Forward Example

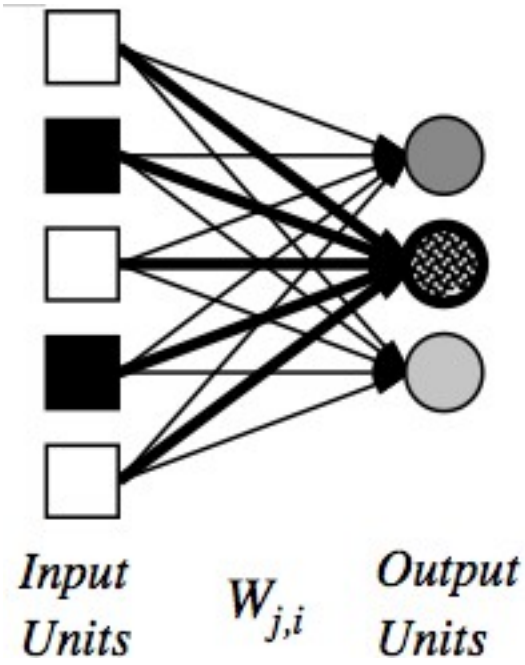


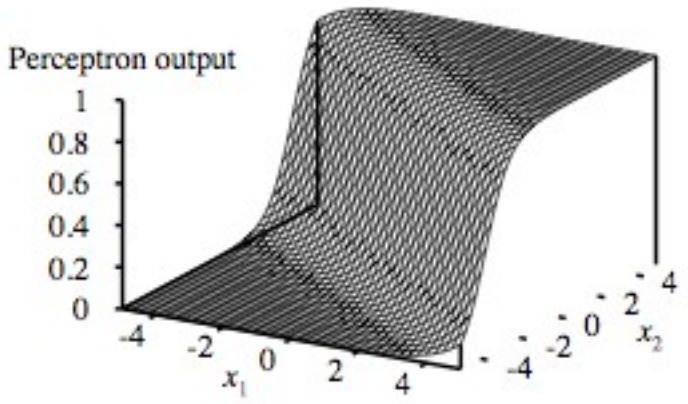
- Feed-forward network = a parameterized family of non-linear functions:

$$
\begin{aligned}
a_5 &= g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \\
&= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2))
\end{aligned}
$$

- Adjusting weights changes the function: this is how learning is done!

Single-Layer Perceptron



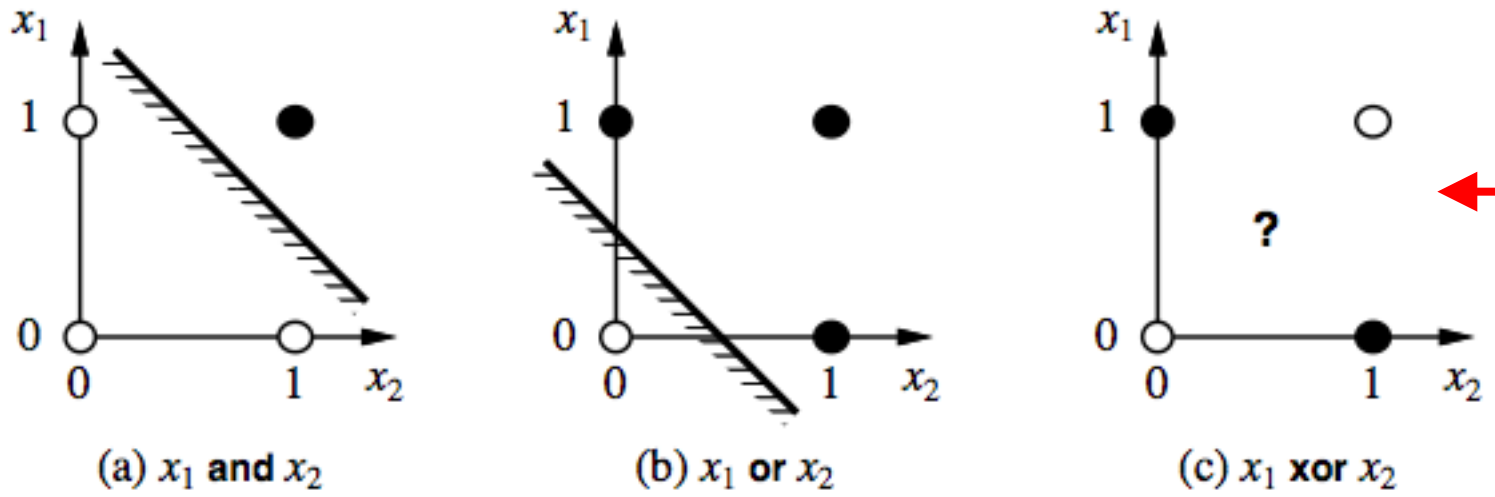


- Output units all operate separately - no shared weights
- Adjusting weights moves the location, orientation, and steepness of cliff

Expressiveness of Threshold Perceptrons

- Consider a perceptron with step activation function
- Can represent AND, OR, NOT, Majority Function (> 1/2 inputs = 1; much more compactly than decision trees -> $O(2^n)$ nodes), but not XOR
- Represents a linear separator in input space (hyperplane in 2-D)

$$\Sigma_j W_j x_j > 0 \quad \text{or} \quad \mathbf{W} \cdot \mathbf{x} > 0$$



- Minsky & Papert (1969) pricked the neural network balloon

Perceptron Learning - Some advantages

- Learn by adjusting weights to reduce error on training set
- The squared error for an example with input $\mathbf{x}$ and true output y is

$$E = \frac{1}{2}Err^2 \equiv \frac{1}{2}(y - h_{\mathbf{W}}(\mathbf{x}))^2 ,$$

- Perform optimization search by gradient descent:

$$\frac{\partial E}{\partial W_j} = Err \times \frac{\partial Err}{\partial W_j} = Err \times \frac{\partial}{\partial W_j}\left(y - g(\Sigma_{j=0}^{n} W_j x_j)\right)$$
$$= -Err \times g'(in) \times x_j$$

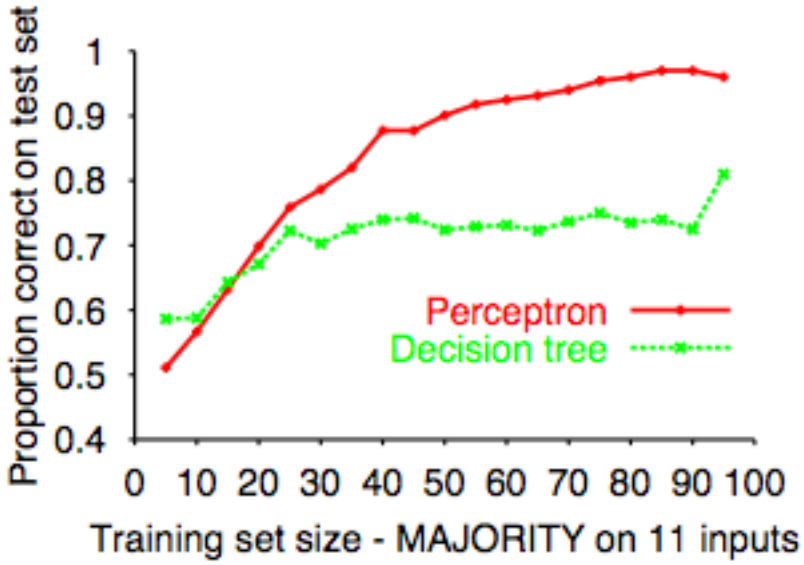
- Simple weight update rule:

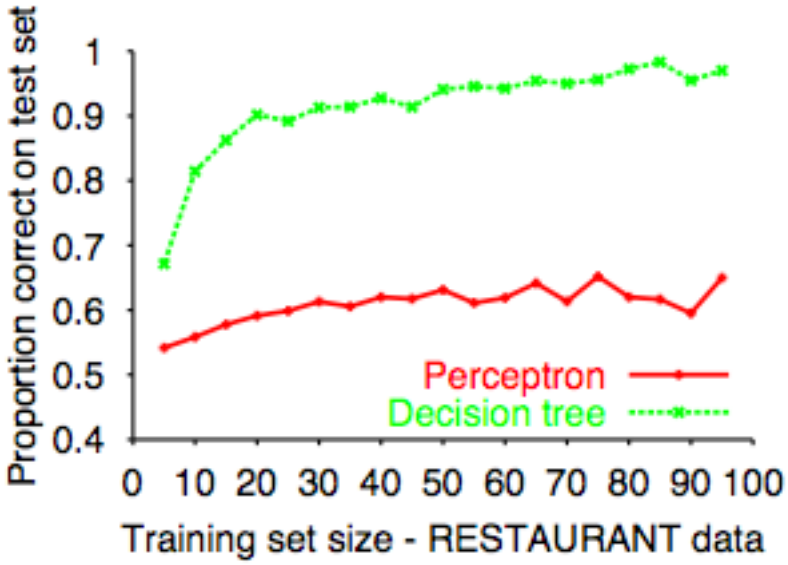
$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j$$

E.g., +ve error $\Rightarrow$ increase network output
$\Rightarrow$ increase weights on +ve inputs, decrease on -ve inputs

Perceptron Learning (cont.)

- Perceptron learning rule converges to a consistent function for any linearly separable data set

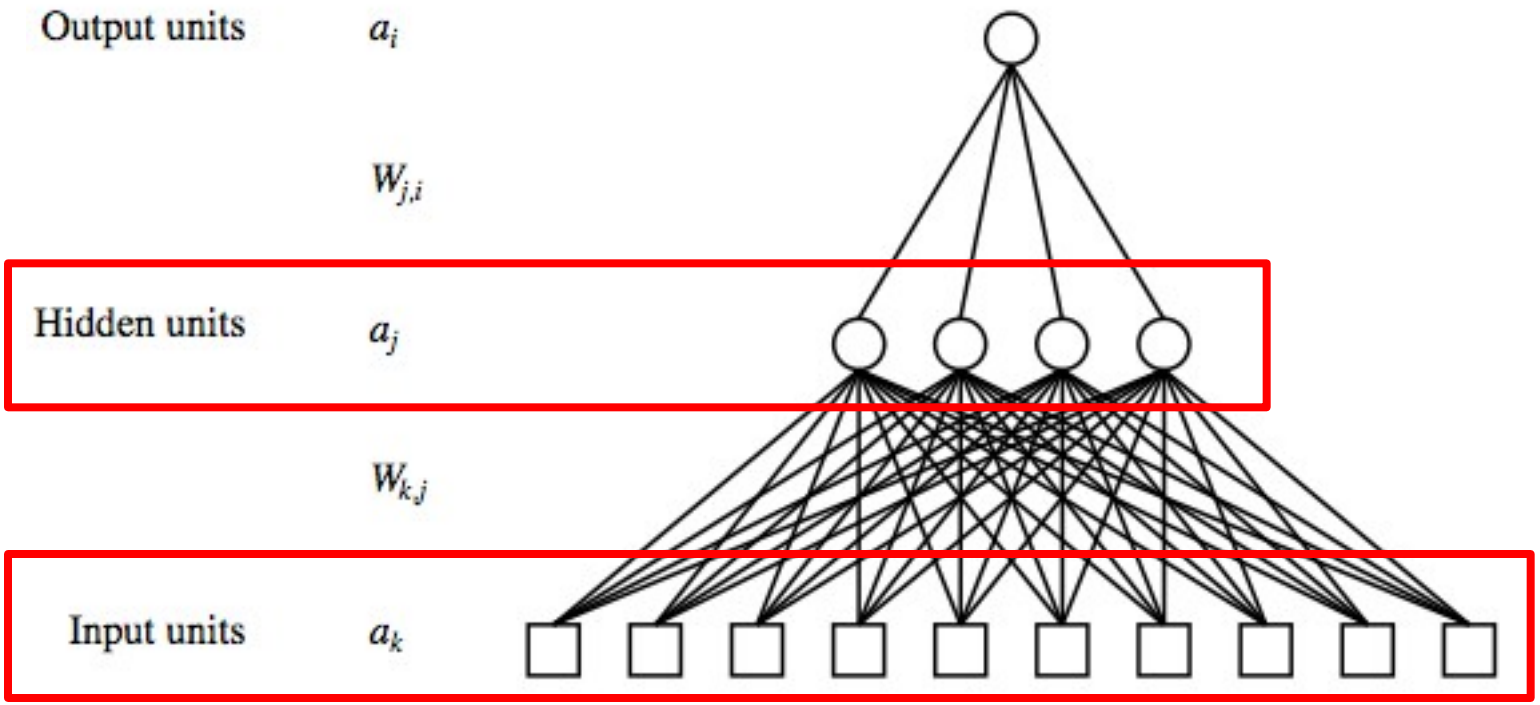




- Perceptron learns function easily, DTL is hopeless
- DTL learns restaurant function easily, perceptron cannot represent it
- Not necessarily deterministic - sigmoid perceptron could be a prob.

Multilayer Perceptrons

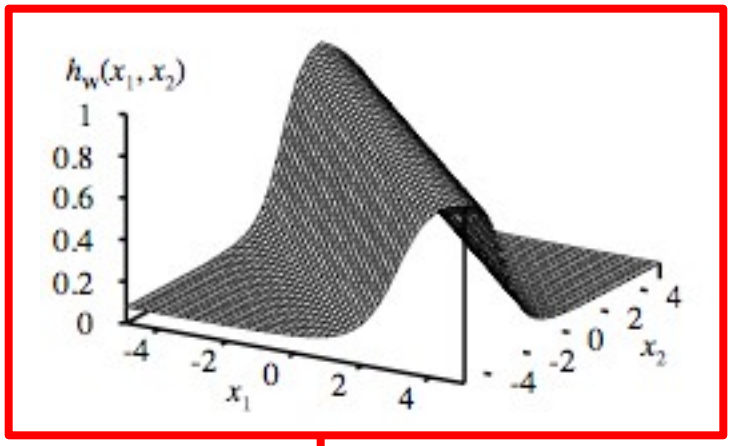
- Most common case is a perceptron with a single hidden layer

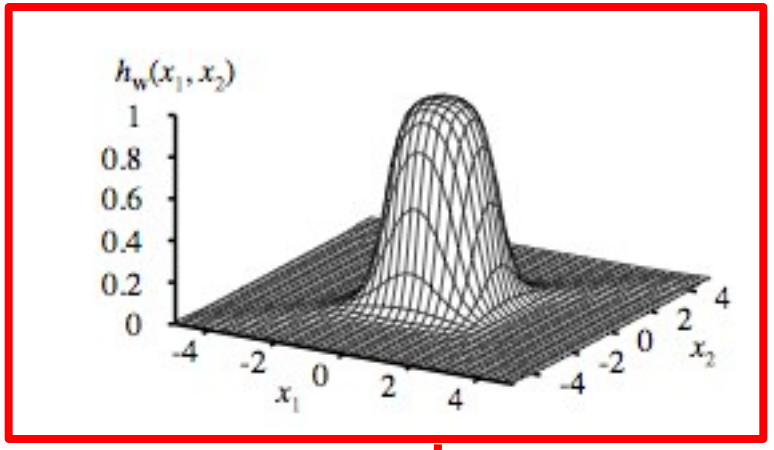


- How many hidden units needed? Not clear …

Expressiveness of Multilayer Perceptrons

- All continuous functions w/ 1 hidden layer, all functions w/ 2 layers





- Combine two opposite-facing threshold functions to make a ridge
- Combine two perpendicular ridges to make a bump
- Add bumps of various sizes and locations to fit any surface
- Proof requires exponentially many hidden units (cf DFL proof)

Back-propagation Learning

- Output layer: same as for single-layer perceptron,

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$$

- where $\Delta_i = Err_i \times g'(in_i)$
- Hidden layer: **back-propagate** the error from the output layer:

$$\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i \ .$$

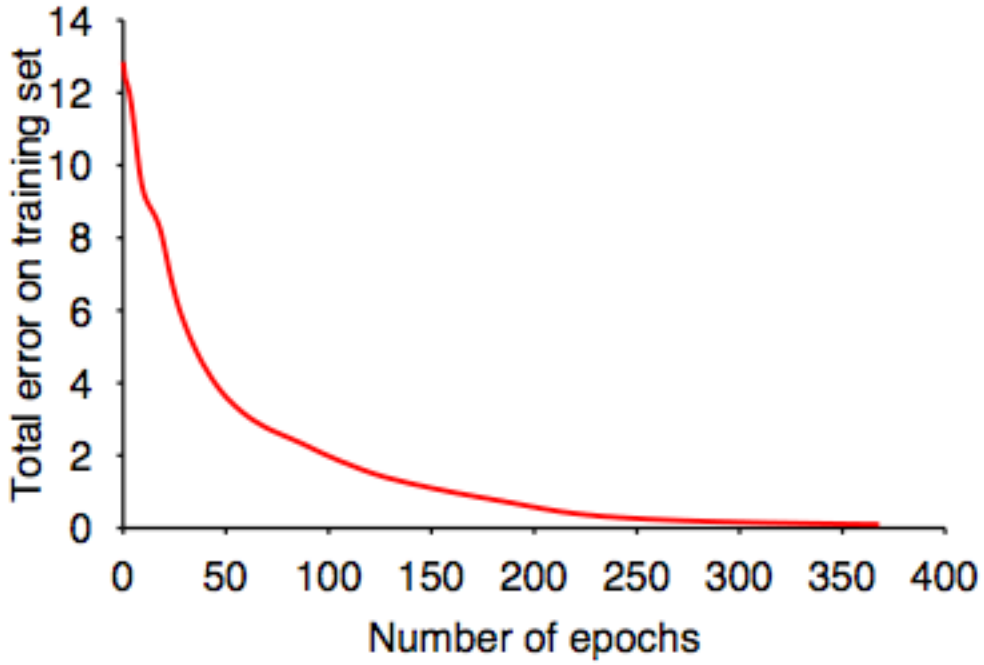
Update rule for weights in hidden layer:

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \times \Delta_j \ .$$

- (Most neuroscientists deny that back-propagation occurs in the brain)

Back-propagation Learning (cont.)

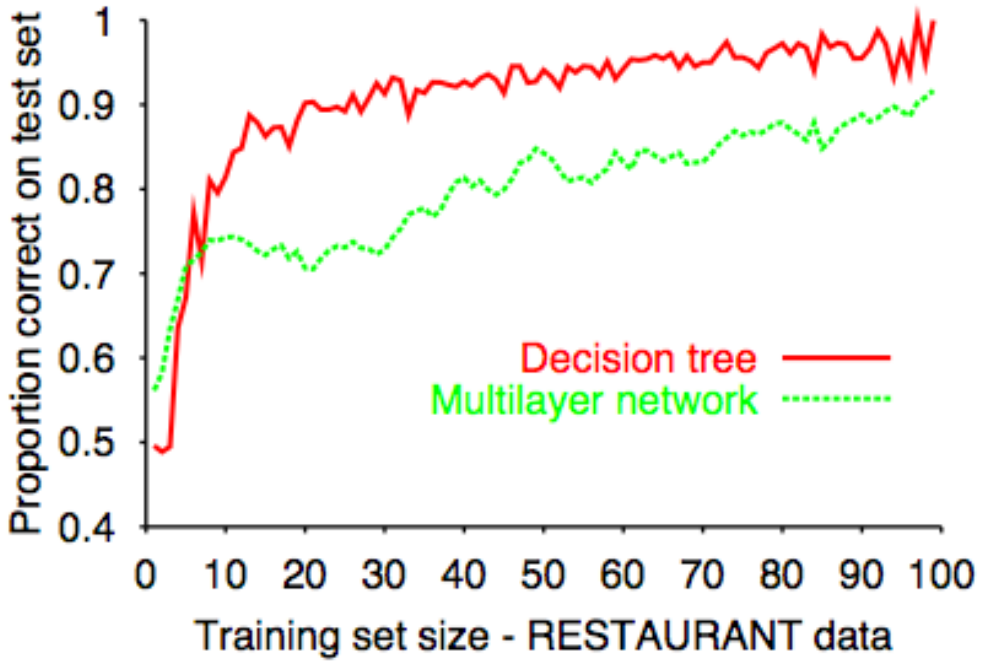
- An epoch is a cycle all the way through the 100 restaurant examples
- Demonstrates that the network converges to a perfect fit!



- Typical problems: slow convergence, lots of local minima

Back-propagation Learning (cont.)

- Learning curve for Multi-Layer Perceptron with 4 hidden units:



- MLPs are quite good for complex pattern recognition tasks, but resulting hypotheses not easily understood

Learning Network Structures

- We have focused on the problem of learning weights, but how does one decide on the network configuration?
  - In particular, need to choose hidden layers and sizes
- If too big, network memorizes training data but there is no generalization - overfitting!
- Approaches to this problem:
  - Try several and keep the best using X-Validation
  - “Optimal brain damage” algorithm
    - Start with a large, fully-connected network
    - Train the network
    - Use information-theoretic approach to drop connections
    - Retrain the resulting network
    - If performance has not decreased, return to step 3
      Otherwise, use the resulting network

Support Vector Machines (SVM)

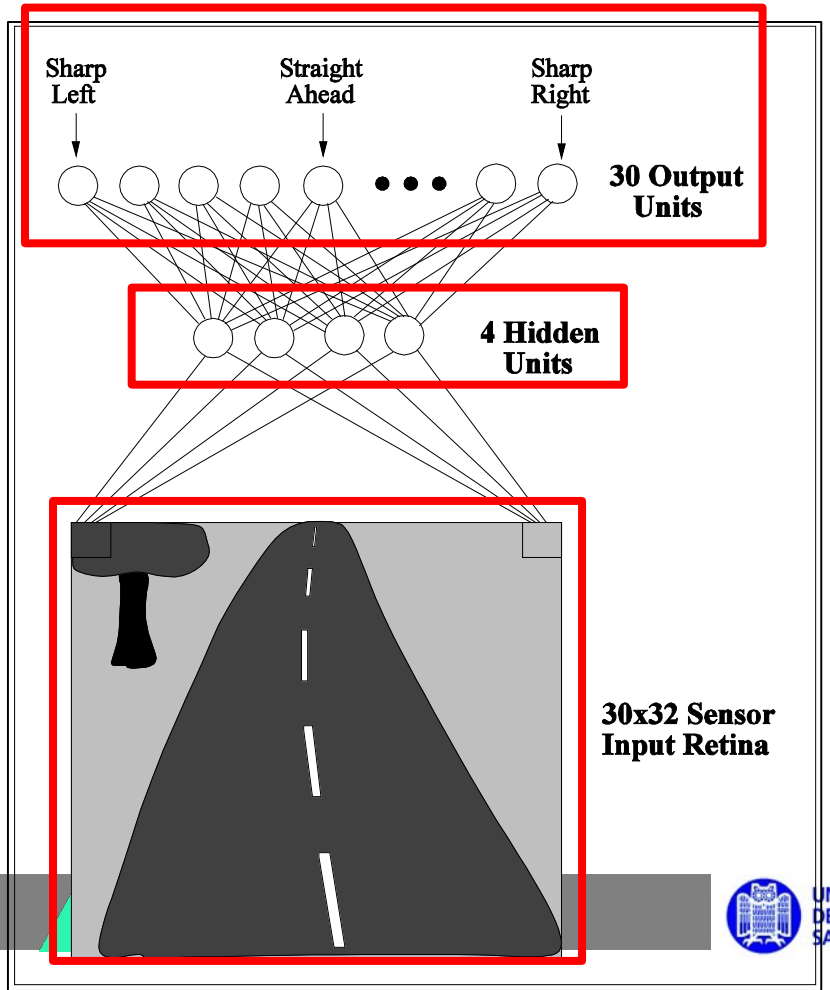
- Relatively new approach, also called “Kernel Machines”
- Goal: Get the best of single- and multi-layer perceptrons
  - Simple & efficient algorithm (like single-layer perceptron)
  - Represent complex functions (like multi-layer perceptron)
- General idea:
  - Re-express the input data using computed features
  - Do this so data is now linearly separable
  - If data are mapped into a space of sufficiently high dimension, then they will always be linearly separable
- Works well with handwritten digit recognition, many input features, dot products of pairs of points

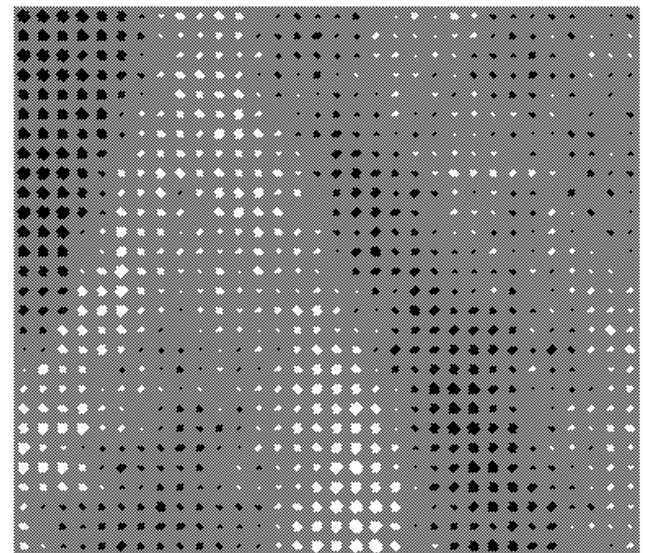
Neural Net Application Example (1)

ALVINN is a perception system which learns to control the NAVLAB vehicles by "watching" a person drive. ALVINN's architecture consists of a single hidden layer back-propagation network.
ALVINN drives 70 mph on highways

Carnegie Mellon

Left Turn!



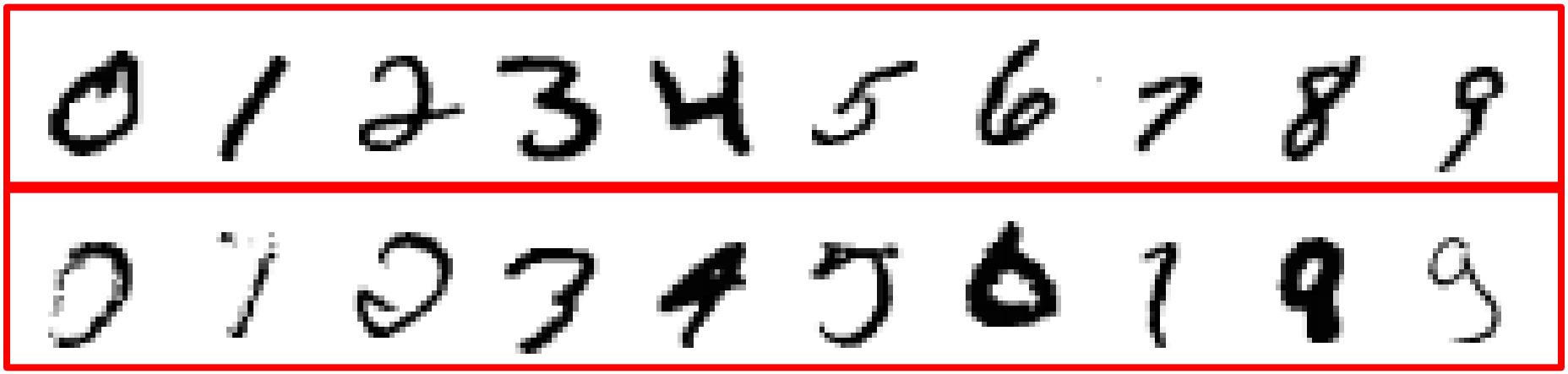


UNIVERSITÄT DES SAARLANDES

Neural Net Application Example (2)
Handwritten digit recognition

Recognizing handwritten digits is an important application!

- Rapid progress, because of algorithms & data
- Fast becoming the benchmark for new algorithms



	Error Rate
Humans	0.2% to 2.5% (estimated)
3-nearest-neighbor	2.4%
Support Vector Machine	1.1 %
LeNet (specialized NN)	0.9%
Boosted LeNet (3 copies)	0.7%

When to Consider Neural Networks

- Input is high-dimensional discrete or real-valued (e.g. raw sensor input)
- Output is discrete or real valued
- Output is a vector of values
- Possibly noisy data
- Form of target function is unknown
- Human readability of result is unimportant
- Long training, quick evaluation

Other Examples of Good Application Areas:

Speech phoneme recognition
Image classification
Financial prediction

Summary of Neural Nets

- Brains have lots of neurons
- Perceptrons (one-layer networks) are simple but insufficiently expressive
- Multi-layer networks are sufficiently expressive; can be trained by gradient descent, i.e., error back-propagation -> complex
- Support Vector Machines are a “best combination” of these two
- Many applications: speech, driving, handwriting, fraud detection, etc.
- Engineering, cognitive modelling, and neural system modelling subfields have largely diverged

Overview

Ensemble Learning

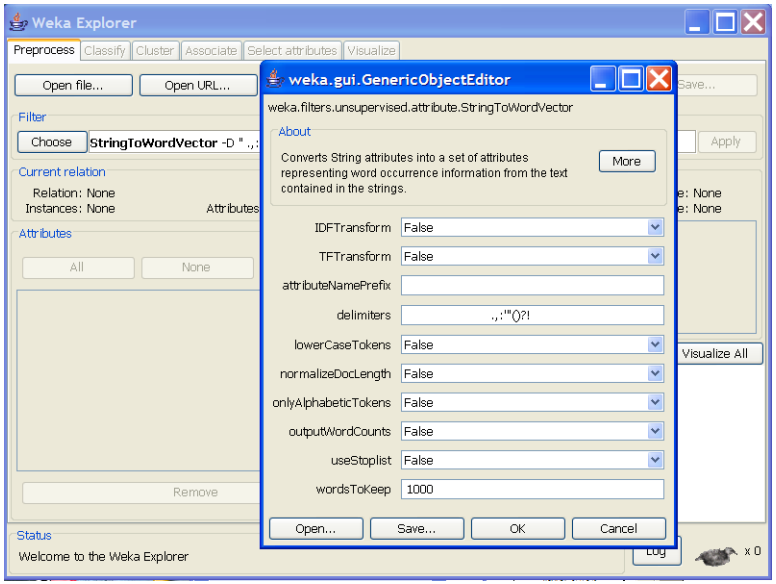
Neural Networks

- Brains
- Neural Networks
- Perceptrons
- Multi-layer Perceptrons
- Support Vector Machines
- Applications of Neural Networks

Learning from Text

Overview of a DFKI Machine Learning Project

Learning from Text: Computing Term Weights



- A common representation for text is to have one attribute per word in your vocabulary
- But what value do you assign to each attribute?
- Simplest approach:
  - Assign 1 or 0 depending upon whether the word occurs in the text at least once
- More complicated approach:
  - Assign count of words

Why is it important to think about term weights?

- How to decide?
  - If term frequency matters for your task, you might lose too much information if you just consider whether a term occurred or not
  - If term frequency doesn't matter, a simpler representation will most likely work better
- Term weights are important for *information retrieval* because in large documents, just knowing a term occurs at least once does not tell you whether that document is "about" that term
- Wisdom on the street: term weights are *less* crucial for supervised text classification tasks

Basics of Computing Term Weights

- Assume occurrence of each term is independent
  - Obviously not true, but a useful simplifying assumption
- Term weight functions have two basic components
  - *Term frequency*: How many times that term occurs in the curr. doc.?
  - *Document frequency*: How many documents did that term occur in?
- Inverse document frequency: a measure of the rarity of a term
  - $\text{Idf}_t = \log(N/n_t)$ where t is the term, N is the number of documents, and n_t is the number of documents where that term occurred at least once
  - Inverse document frequency
    - 0 in the case where a term occurs in all documents
    - approaches 1 for very rare terms
- A scheme that combines term frequency with inverse document frequency
  - $W_{t,d} = tf_{t,d} X\ idf_{t,d}$

Trying Different Term Weights

- Predicting Class1, 72 instances
- What is different is the relative weight of the different features.
- Whether this matters depends on the learning method

	VotedPerceptron	*NaiveBayes*
Binary	**Pre(.5) Rec(.44) F(.47)**	Pre(.27) Rec(.78) F(.4)
Counts	Pre(.5) Rec(.33) F(.4)	Pre(.28) Rec(.78) F(.41)
TF	Pre(.33) Rec(.11) F(.17)	**Pre(.3) Rec(.78), F(.44)**
TF.IDF	Pre(.44) Rec(.44) F(.44)	Pre(.2) Rec(1) F(.33)

Learning from Text: Is it all about Words?

- The word learning approach is useful but is that enough?
- What about the context of a conversation? e.g.,
  - "John is simply lazy!" could have possible uses
    - Initiating statement
    - Conflict-Oriented Consensus Building: In response to "John thinks that he is not talented."
    - Quick Consensus Building: In response to "The reason in the case of John is laziness."
- Current work by Rosé et al within TagHelper to account for context
  - Defining sequence-oriented features (e.g., contribution number)
  - Use of sequential learning techniques

Overview

Ensemble Learning

Neural Networks

- Brains
- Neural Networks
- Perceptrons
- Multi-layer Perceptrons
- Support Vector Machines
- Applications of Neural Networks

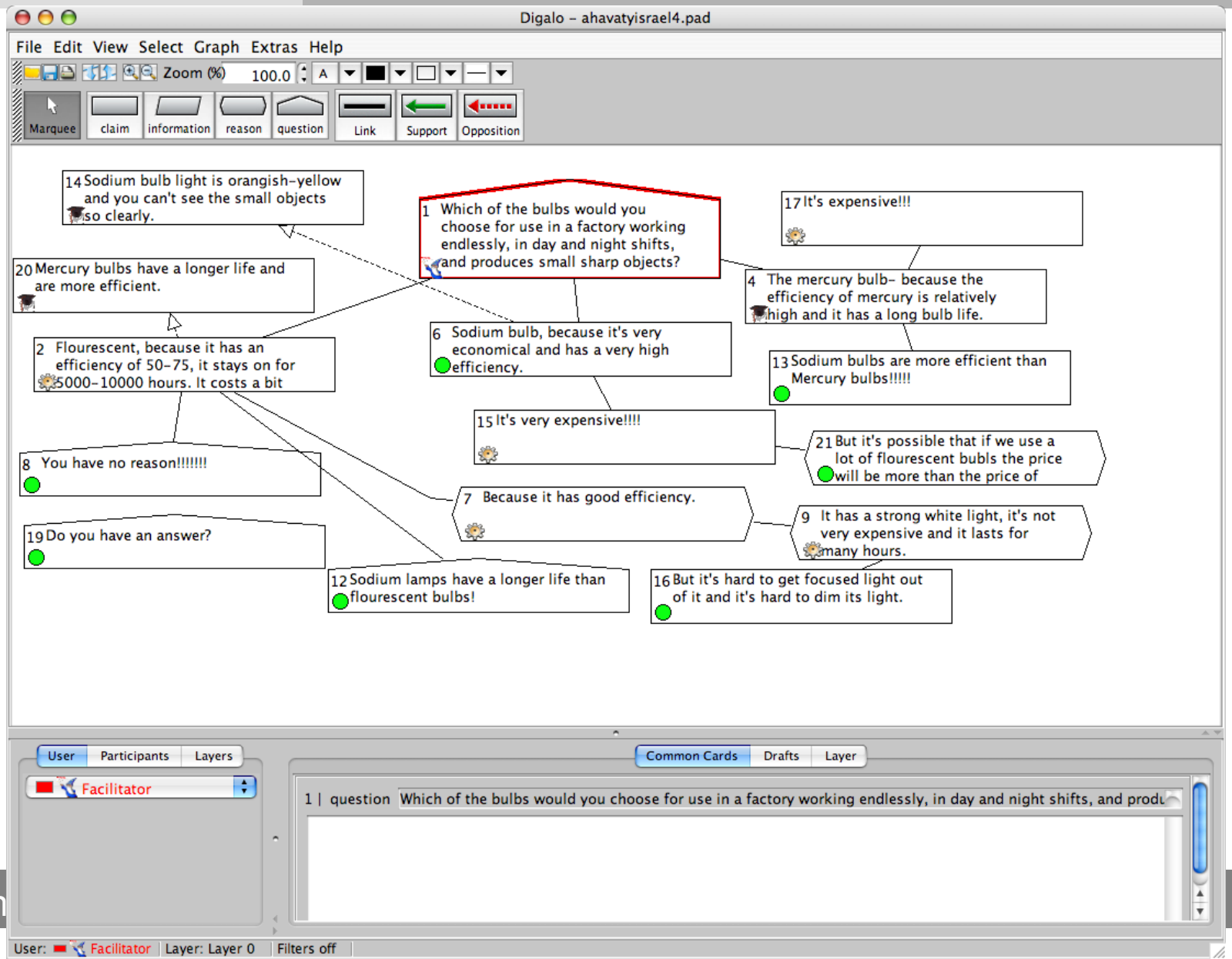
Learning from Text

Overview of a DFKI Machine Learning Project

Machine Learning in Action: The ARGUNAUT Project

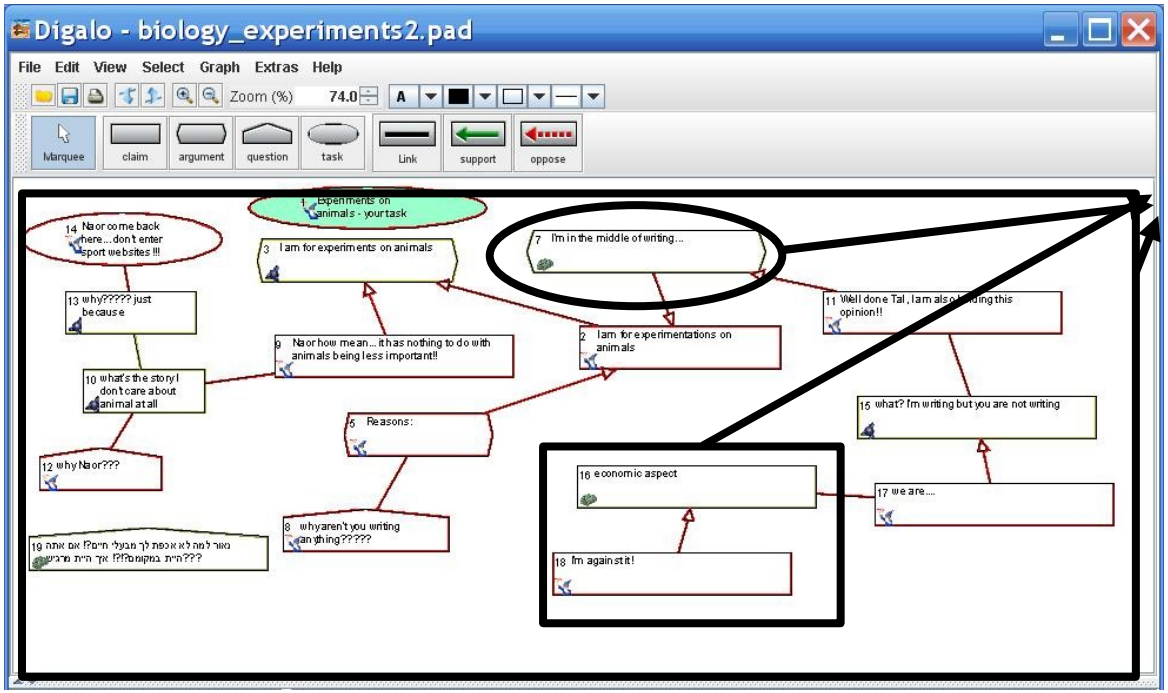
- *Educational Data Mining* - the ARGUNAUT Project
  - Use of Machine Learning Techniques to learn collaborative / argumentative behaviors of students
  - Use the resulting classifiers to support the moderator of an online collaborative discussion
- Looking for an enthusiastic student / HIWI to work on this project
  - Learn to use modern Machine Learning tools
  - Integrating data and text mining
  - Work with a team of learning scientists and PhD students
  - Academic publications!
- Talk to me after the lecture or send email to **bmclaren@dfki.de**

Students Collaborate in an e-Discussion



Mach

Discourse Analysis - Units of Analysis



Sequence-level

Map-level

Sequences of actions

Full discussion

related shapes

map as unit of analysis

(Paired shapes, clusters of shapes linked together, sequences of actions)

Coding of the Discourse

Coding Categories	*Shape 1*	*Shape 2*
Shape Level - Chain of Reasoning	Today there are resistant bacteria because of unnecessary use of antibiotics	---
Shape Level - Request for Clarification	Was the cause of death only because of bacteria resistance?	---
Paired-Shapes Level - Contribution-CounterArgument	As for the biological aspect, in the wild investing in the weak is considered a waste of resources and energy	Against. According to the human aspect you can't ignore and neglect the weak
Paired-Shapes Level - Question-Answer	What is the reason for that ??? In the wild she neglects the weak?? Out of laziness?? Or does she simply doesn't have the means??	The strongest survives. In nature it's the strong who survives, and in the wild there will most likely be no one to take care of a wounded or helpless animal, unless it's a cub.

Process of Learning from the Discourse

Digalo or FreeStyler

Common Format XML Files

Transform Maps for Annotation

Graphical View 1 Excel File

...

Graphical View N Excel File

Students

Students collaborate in an e-discussion …
For experimenting, we are use existing maps

Various Software used to:
1. Transform from process to graphical view
2. Transform to a particular graphical view
3. Transform to Excel format (i.e., tab-delimited)

Excel

Annotated View 1 Excel File

...

Annotated View N Excel File

Transform Annotated Maps for Machine Learning

Annotated Maps Database

YALE

Classifiers

Evaluation Of Classifiers

Expert Pedagogists

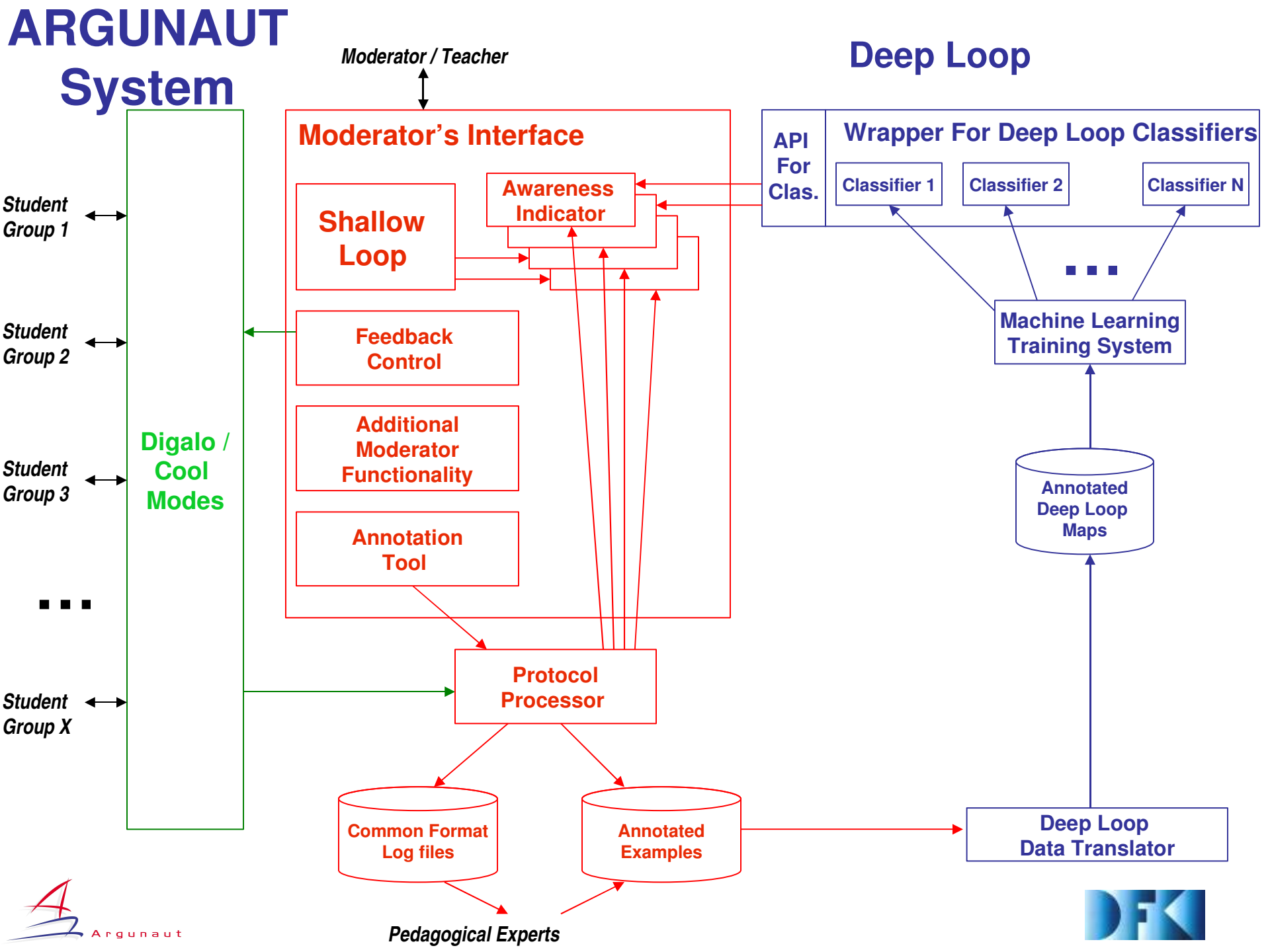
Perl script to transform from Excel format to database

Expert Pedagogists annotate graphical views …

AI Specialists

AI Specialists run machine learning experiments …

ARGUNAUT System



Tools Used on ARGUNAUT

- Many machine learning algorithms packaged within shareware tools
- **WEKA** (**W**aikato **E**nvironment for **K**nowledge **A**nalysis)
  - ✓ Contains core learning algorithms (e.g., Decision Trees, OneR)
- **YALE** (**Y**et **A**nother **L**earning **E**nvironment)
  - ✓ Object-oriented experimental environment
  - ✓ Integrated with MySQL, a database system
  - ✓ Learned classifiers can be integrated with existing applications
- **TagHelper (Rose et al)**
  - ✓ Built upon WEKA, uses learning algorithms that can process language elements (e.g., Naïve Bayes)

Machine Learning in Action: Summary of the ARGUNAUT Project

- Learning of Collaborative/Argumentation Data & Text
  - Structure of Discussion Maps
  - Actions taken
  - Content (i.e., text) of contributions
- Use what is learned to help a moderator support later e-Discussions
- Early results for some categories have been very good!
  - Shape Level: “Request for Clarification,” Kappa = 0.6
  - Paired-Shapes: “Question-Answer,” Kappa = 0.78

Argunaut

End of Machine Learning Lecture 2

- *Educational Data Mining* - the ARGUNAUT Project
  - Use of Machine Learning Techniques to learn collaborative / argumentative behaviors of students
  - Use the resulting classifiers to support the moderator of an online collaborative discussion
- Looking for an enthusiastic student / HIWI to work on this project
  - Learn to use modern Machine Learning tools
  - Integrating data and text mining
  - Work with a team of learning scientists and PhD students
  - Academic publications!
- Talk to me after the lecture or send email to **bmclaren@dfki.de**


End of Machine Learning Lecture 2