The role of AI and learning

How to put our experience to work
Learning to manage

- Operations Research techniques learn the management policy from data and mathematical models

- Artificial Intelligence techniques learn the management policy from data and human experience
The OR role

Simulation Model

\[ x(t+1) = f(x(t), u(t)) \]

Control Variables

Search algorithm

Exact, Approximation or Heuristic Search

Simulation Results

Performance Criterion

\[ J = J(x, u) \]

Time series

Maps

\[ J_{max}? \]

yes

Stop
The AI role

Simulation Model

$x(t+1)=f(x(t),u(t))$
or a qualitative model

Control Variables

u(t)

Search algorithm

Expert

Performance Criterion

$J=J(x,u)$
or qualitative judgement

Simulation Results
may be qualitative

Time series

Maps

$J_{max}$?

Search algorithm

Stop

yes
What is AI?

• Also known as Machine Intelligence

• AI is what has not been done yet!

• “making a machine behave in ways that would be called intelligent if a human were so behaving” (McCarthy, 1956)
The goal of Artificial Intelligence

• We can see AI from 4 different perspectives:
  1. Systems that think like humans
  2. Systems that act like humans
  3. Systems that think rationally
  4. Systems that act rationally

• Each perspective leads to different definitions for AI
Systems that think like humans

• The automation of activities that we associate with human thinking, such as decision-making, problem solving, learning (Bellman 1978)

• The exciting new effort to make computers think (Haugeland, 1985)
Thinking humanly

• This implies understanding how humans think

• Once we understand the mind, we can try to replicate it on a machine

• Example: general problem solver (Newell, Simon 1961) → solve a problem, trace the reasoning steps and compare with human reasoning steps

• Cognitive science
Systems that act like humans

- The art of creating machines that perform functions that require intelligence when performed by people (Kurzweil, 1990)
- The study of how to make computers do things at which, at the moment, people are better (Rich and Knight, 1991)
Acting humanly

- Turing test (1950) → involves an interrogator, a human, a machine
- Requires the computer to have the following capabilities:
  1. natural language processing → to communicate in English
  2. knowledge representation → to store information provided during the test
  3. automated reasoning → to use stored information to answer questions and draw conclusions
  4. machine learning → to adapt to new circumstances
Systems that think rationally

- The study of mental faculties, through the use of computational models (Charniak and McDermott, 1985)

- The study of the computations that make it possible to perceive, reason and act (Winston, 1992)
Thinking rationally

- Aristotle (384-322 BC) attempts to codify rational thinking → syllogisms
- One of the basis of Formal Logic
- Write a problem as a logic formula and test satisfiability
- Problems: hard to formulate informal knowledge in logic, uncertainty, imprecision, but also complexity
- New results suggest it may be doable (Kautz, 2004)
Systems that act rationally

- A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes (Schalkoff, 1990)

- The branch of computer science that is concerned with the automation of intelligent behavior (Luger, Stubblefield, 1993)
Acting rationally: rational agent

- Acting rationally: acting to achieve one’s goals, given one’s belief
- Agent: something that perceives and acts
- AI is viewed as the study and construction of such agents
A brief history of AI

• 1950: Alan Turing: Turing test

• 1943-56: McCulloch/Pitts: research on the structure of the brain gives a model of neurons of the brain → artificial neural networks

• 1951: von Neumann helps Minsky and Edmonds to build the first neural network computer
A brief history (cont.)

• 1950: Claude Shannon publishes a paper on chess playing. Shows that a game of chess involved about $10^{120}$ moves → shows the need for heuristics

• 1956: McCarthy convinces Minsky and Shannon to organize the first AI workshop and Dartmouth college, sponsored by IBM → birth of AI
A brief history (cont.)

• Great expectations on what computers can do

• 1958: McCarthy presents a paper “Program with common sense”.

Advice taker: search for solutions of general problems

• 1962: Rosenblatt proves the perceptron convergence theorem (learning algorithm)
A brief history (cont.)


- 1965: Zadeh introduces Fuzzy sets
A brief history (cont.)

- Initial goal: build all-purpose intelligent machines
- 1970: realize this is just too optimistic
- 1971: Cook introduces NP-completeness
- Early 70s: shift from a general purpose, knowledge-sparse, weak methods to domain-specific, knowledge-intensive techniques
  1. Dendral: molecular structure of martian soil based on mass spectral data
  2. Mycin: rule-based expert system for diagnosis of infectious blood diseases
A brief history (cont.)

- Mid 80s: use of neural networks for machine learning. Much later after Rosenblatt due to computational power
- Generalization of single-layer network: Hopfield network, back-propagation etc…
- Genetic algorithms → survival of the fittest
- Knowledge engineering: use of Fuzzy logic improves computational power, improves cognitive modeling, allows to represent multiple experts
- Artificial life
Two types of AI

- Neat AI: symbolic manipulation of concepts (knowledge representation, rule-based and case-based systems)
- Scruffy AI: try to evolve intelligence through machine learning (Neural Networks, Reinforcement Learning, Bayesian learning)
Machine Learning

most of this material adapted from Tom Mitchell’s book
Overview

• Concept Learning
• Decision Trees
• Bayesian Learning
• Instance Based Learning
• Artificial Neural Networks
Concept Learning

Inferring a boolean-valued function from training examples of inputs and outputs
Representing hypothesis

- Conjunction of constraints
- E.g \{Sunny, Hot, High, Weak\}
- Play tennis IF Outlook=Sunny AND Temp=Hot AND Humidity=High AND Wind=Weak
The training examples

<table>
<thead>
<tr>
<th>#</th>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Warm</td>
<td>Norm</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Rainy</td>
<td>Cold</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Representing hypothesis

- The symbol ? stands for “don’t care”
- E.g. {Sunny, ?, ?, ?}
  - Play tennis if Outlook=Sunny
Representing hypothesis

- The symbol $\emptyset$ means that no value is acceptable
- E.g. $\{\emptyset, \emptyset, \emptyset, \emptyset\}$
- Never play tennis
General to Specific Ordering

{?, ?, ?, ?}

{Sunny, ?, ?, ?}  {?, Hot, ?, ?}  {?, ?, High, ?}  {?, ?,?,Weak}

{Sunny, Hot, ?, ?}  {?,∅, ?, ?}  {?, Hot, High, ?}  {?, Hot, ?,Weak}

...  

...  

{∅, ∅, ∅, ∅}

...
Find-S

• Start with the most specific hypothesis
• For each positive example:
  • If current hypothesis doesn’t cover the example generalise in order to satisfy it
• Ignores negative examples!
Find-S Example

• \{\emptyset, \emptyset, \emptyset, \emptyset\} Most specific

• \{Sunny, Warm, Normal, Strong\} the first example

• \{Sunny, Warm, ?, Strong\} the second example generalises the third attribute

• Since we ignore negative examples, this is the solution
Problems with Find-S

- Only outputs one hypothesis which is consistent with positive examples
- May be many other hypothesis which are potential candidates
- Ignores negative examples
Version spaces

- A version space is the set of all possible hypothesis in the hypothesis ‘language’ which are consistent with the examples.

- \( \text{Consistent}(h, D) \iff h(x) = c(x) \) for each \(<x, c(x)>\) in \(D\).
List-then-eliminate

- $\text{VS} \leftarrow \text{every } h \text{ in } H$
- For example $<x, c(x)>$
  - remove from VS $h$ where $h(x) \neq c(x)$
- Output all $h$ in VS
Candidate-Elimination

• Like Find-S but going in both directions.

• Uses positive examples to find set S of most specific hypothesis

• Uses negative examples to find set G of most general hypothesis
Example

\{Sunny, Warm, ?, Strong\}

\{Sunny, ?, ?, Strong\}

\{Sunny, ?, ?, ?\}

\{?, Warm, ?, ?\}
Decision Trees

• Outputs the hypothesis in the form of a tree
  • Can be thought of as rules, or disjunction of conjunction of constraints
  • Very popular in data-mining because it is easy to understand algorithm and output
Outlook

- Sunny
  - Humidity
    - High: No
    - Normal: Yes
  - Overcast: Yes
- Rain
  - Wind
    - Strong: No
    - Weak: Yes
Disjunction and conjunction of rules

• The previous tree corresponds to:
  • (Outlook=Sunny AND Humidity = Normal)
  • OR (Outlook=Overcast)
  • OR (Outlook=Rain AND Wind=Weak)
When to use a DT

- Instances are represented by attribute-value pairs
- The target function has discrete output values
- Disjunctive descriptions may be required
- Training data may contain errors and/or missing values
A basic algorithm

- ID3 (Quinlan, 1993)
- “which attribute should be tested at the root of the tree?”
- the best one is selected and a descendant node is created for each possible value of the attribute
• ID3(Examples, Target, Attributes)
• A ← the Attribute that best classifies Examples
• Root ← A
• For each possible value $v_i$ of A
  • add a new tree branch for $A = v_i$
  • find the subset Examples$_{v_i}$ of Examples which have value $v_i$ for A
  • If Examples$_{v_i}$ is empty
    • add a leaf with the label of the most common value of Target in Examples
  • else
    • add a new subtree ID3(Examples$_{v_i}$, target, Attributes - {A})
Entropy to select the best attribute

\[ Entropy(S) = -p^+ \log_2(p^+) - p^- \log_2(p^-) \]

- where \( p^+ \) is the proportion of positive examples in \( S \), and \( p^- \) the proportion of negative ones
- entropy specifies the minimum number of bits needed to encode the classification of an arbitrary member of \( S \)
Example

- S is a collection of 14 examples
- S=\([9^+,5^-]\) 9 examples return a positive classification and 5 a negative classification
- \(\text{Entropy}(S) = -(9/14)\log(9/14)-(5/14)\log(5/14) = 0.94\)
- It is too high! (zero entropy is best, 1 is worst)
Example

- We now select the attribute Wind, which can be either Weak or Strong
- \( \text{Entropy(Weak)} = [6+,2-] \)
- \( \text{Entropy(Strong)} = [3+,3-] \)
- \( \text{Gain}(S, \text{Wind}) = 0.94 - (8/14) \times 0.811 - (6/14) \times 1 \)

\[
\text{Gain}(S, \text{Wind}) = E(S) - \sum_{v \in [\text{Weak,Strong}]} \frac{S_v}{S} E(S_v)
\]
Bayesian Learning

- Probabilistic method
- The brute force approach
- The Naive Bayes Classifier
Features

• Each training example can incrementally increase or decrease the estimated probability of hypothesis correctness

• Prior knowledge can be combined with observed data

• Can produce probabilistic predictions

• Require the initial knowledge of many probabilities

• Significant computational cost
Bayes Theorem

\[ P(h|D) = \frac{P(D|h)P(h)}{P(D)} \]

- \( P(h) \) = Prior probability of hypothesis
- \( P(D) \) = Prior probability of training data D.
- \( P(h|D) \) = Probability of h given D.
- \( P(D|h) \) = Probability of D given h.
Choosing a Hypothesis

\[ h_{MAP} = \arg\max_{h \in H} P(h|D) = \arg\max_{h \in H} P(D|h)P(h) \]

• Maximum a posteriori hypothesis
• it is the most probable hypothesis given the observed data
• if every hypothesis has the same a priori probability:

\[ h_{MAP} = \arg\max_{h \in H} P(D|h) \]
An example

• HP1: the patient has a form of cancer
• HP2: the patient does not
• We can perform a lab test, which can be positive (+) or negative (-)
• Only .08 % of the total population has this form of cancer
• The test is a correct positive in 98% if the disease is present
• The test is a correct negative in 97% if the disease is not present
Example

- We have a new patient, from whom the lab returns a positive result. What should we do?
  - \( P(+|\text{cancer}) \times P(\text{cancer}) = .98 \times .008 = .0078 \)
  - \( P(+|\text{not cancer}) \times P(\text{not cancer}) = .0298 \)
  - \( h_{\text{MAP}} = \text{not cancer!} \)
Remarks

• Since the a priori probability of not having cancer is very high

• The a posteriori probability of having cancer is higher than the a priori one...

• ...but it is still lower than the a posteriori one of not having cancer

• Bayesian inference depends strongly on prior probabilities
Brute Force MAP Learner

• Calculate $h_{MAP}$ for all hypothesis
• Output the hypothesis with higher posterior probability.
• Useful conceptually: we can show that Find-S is a MAP learner.
Bayesian Classification

- MAP selects the most probable hypothesis
- the MAP hypothesis is the most probable classification of the new instance given the training data
- Can we do better than MAP?
An example

• $p(h_1|D) = 0.4$ \quad $p(h_2|D)=0.3$ \quad $p(h_3|D)=0.3$

• $h_1$ is the MAP hypothesis

• new instance $x$, positive for $h_1$, negative for $h_2$ and $h_3$

• the most probable classification for $x$ is negative ($p(-)=0.3+0.3 > p(+)= 0.4$)

• $\neq$ from the MAP classification
Use the available information

- We must use the prediction of all hypotheses, weighted by their posterior probabilities

- \( p(-|h1)=0 \) \( p(+|h1)=1 \)
- \( p(-|h2)=1 \) \( p(+|h2)=0 \)
- \( p(-|h3)=1 \) \( p(+|h2)=0 \)
- \( p(+|h1)\times0.4 + p(+|h2)\times0.3+p(+|h3)\times0.3=0.4 \)
- \( p(-|h1)\times0.4 + p(+|h2)\times0.3+p(+|h3)\times0.3=0.6 \)
- the maximum is 0.6 for a **negative** classification
Bayesian Classification

• Bayes optimal classification:

\[
\arg\max_{v_j \in V} P(v_j | D) = \arg\max_{v_j \in V} \sum_{h_j \in H} P(v_j | h_i) P(h_i | D)
\]

• where \( V \) is the set of values which can be assumed by the classification of the instance \( x \)

• Cannot be outperformed on average using same \( H \) and same prior knowledge.

• But too expensive in real life.
Naive Bayes classifier

- Works for learning tasks where each instance $x$ is described by a set of attributes $\langle a_1, a_2, ..., a_n \rangle$
- The target function $f(x)$ takes any value on the finite set $V$
Naive Bayes

\[ v_{MAP} = \arg \max_{v_j \in V} P(v_j | a_1, a_2, \ldots, a_n) \]

\[ = \arg \max_{v_j \in V} \frac{P(a_1, a_2, \ldots, a_n | v_j)P(v_j)}{P(a_1, a_2, \ldots, a_n)} \]

\[ = \arg \max_{v_j \in V} P(a_1, a_2, \ldots, a_n | v_j)P(v_j) \]

We must estimate \( P(a_1, \ldots, a_n | v_j) \) and \( P(v_j) \) using the training data
Naive Bayes

- $P(v_j)$ easy to estimate: count the frequencies with each target value occurs in the training data
- $P(a_1, ..., a_n | v_j)$ is impossible to compute on large data sets!
- Simplifying assumption: the attribute values are conditionally independent given the target value
  $$P(a_1, a_2, ..., a_n | v_j) = \prod_i P(a_i | v_j)$$
Naive Bayes classifier

\[ v_{NB} = \arg \max_{v_j \in V} \prod_i P(a_i | v_j) \]

- \( P(a_i | v_j) \) is easy to estimate from the frequency of the attribute \( a_i \) in the training data when \( v_j \) occurs
- No explicit search in the hypothesis space. The hyp. is formed by counting frequencies
Bayesian Belief Networks

- Naive Bayes classifier makes too many assumptions of independence
- On the other hand, full dependence would make inference and learning impossible
- Bayesian Belief Networks are a compromise
Bayesian Belief Networks

• Naive Bayes assumes all variables are conditionally independent

• Bayes Nets relax this and assume some variables are conditionally independent

• Learnt using gradient ascent methods or EM
The network

- Proximity
- Tuberculosis
- TBC or Cancer
- X-rays
- Smoker
- Lung Cancer
- Dyspnoea
- Bronchitis
The conditional dependence

- Each node conditionally depends only on its immediate ancestors (parents)
- Dyspnoea depends on TBC_or_Cancer and on Bronchitis, but not on Lung Cancer
- These tables are usually built with expert aid
- Learning the structure is difficult
Example

<table>
<thead>
<tr>
<th></th>
<th>TBC_or_C, TBC_or_C, TBC_or_C, TBC_or_C,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B, B, ¬B, ¬B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dyspnoea</th>
<th>0.9</th>
<th>0.8</th>
<th>0.6</th>
<th>0.2</th>
</tr>
</thead>
</table>

| ¬Dyspnoea | 0.1 | 0.2 | 0.4 | 0.8 |
E = evidence = a set of observed random variables = current knowledge
- Computation of $P[X=x|E=e]$: prediction, diagnosis
- Application Fields: medicine, quality control, vision, etc...
- Inference Algorithms [Pearl (88), Lauritzen and Spiegelhalter (88)]
Remarks

• BN = Discrete Probability Distribution \( O(2^N) \rightarrow O(2^{n_{\text{max}}}) \)

• Sparsity: \( n_{\text{max}} = \text{max n. of parents for a node in the net} \)
  • Models NOT Viable Otherwise
    • Estimation Problem / Spatial Complexity Problem / Temporal Complexity Problem

• Real Applicability

• Main Model for Uncertain Reasoning
Assessing Probabilities – Problems

[Chrisman (96), Cozman (96), Walley (91)]

• Assessing Precise Probabilities may not be possible

• Precise Values = Deep Knowledge of the Phenomenon

• group of experts, different beliefs

• Precise Probabilities Are NOT a Realistic Approach
Instance based learning
Instance Based Learning

- Training examples are simply stored in the system
- It ‘delays’ learning to the moment a new hypothesis is presented to the system
- It is therefore known as ‘lazy’ learning
k-Nearest Neighbour

- Given an unseen example, use a distance metric to find $k$ closest examples in training data.
- Classification: Take a vote amongst the training examples
- Regression: Take average of target values.
Classification algorithm

- Given $x_q$ to be classified
- Find $x_1 \ldots x_k$ the $k$ instances closest to $x_q$ according to a given metric
- Return $\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$
Pros and Cons of kNN

- Advantages:
  - Training is very fast
  - Can learn complex functions
  - Don’t lose any info in training set

- Disadvantages:
  - Slow at query time
  - Irrelevant attributes make classification poor
Shepard’s Method (Distance Weighted kNN)

- Instead of taking a straight vote, weight according to distance from unseen example.
- Makes sense to use all training examples instead of kNN.
Estimating $Q$

\[ \hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a') \]

- $r =$ Reward received when performing action $a$ in state $s$.
- $\gamma =$ Discounting factor
Estimating $Q$

$$Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$$

- Immediate reward
- Plus discounted estimated total reward from next state
Explore vs Exploit

- Refine estimate of Q
- Allows agent to find better rewards

- Exploit:
  - Use knowledge of Q to get biggest reward
Experimentation Strategy

• Epsilon-greedy strategy
• Choose best action with probability $1 - \epsilon$
• Choose a random action (explore) with probability $\epsilon$
Experimentation Strategy

- Choose actions probabilistically:

\[ P(a_i; s) = \frac{k^{Q(s, a_i)}}{\sum_j k^{Q(s, a_j)}} \]

- \(k > 1\) and big: Exploit more
- \(k > 1\) and small: Explore more
- \(k = 1\): Behave randomly
- \(k < 1\): Avoid reward! (Maso(h)ist agent?!
Experimentation Strategy

• Could also have agent act randomly while training

• Could provide experience
  • Control a robot with a joystick
  • Give an agent predefined sequences to learn from
Large State Spaces

- Method presented before works for small state space
  - Rote learning, no generalisation
- Could replace Q estimate with a function approximator
- Convergence guarantees don’t work
Artificial Neural Networks
How does the brain compute?

The biological inspiration of ANNs
The brain

- Biological systems able to learn are made of complex neuron networks.
- The human brain contains approximately 10 billion neurons, each one connected to at least 10,000 neurons.

The brain: my second favourite organ!
(Woody Allen)
# Brains vs computers

<table>
<thead>
<tr>
<th>elements</th>
<th>power</th>
<th>speed</th>
<th>computations</th>
<th>learns</th>
<th>sentient</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{14}$ synapses</td>
<td>30W</td>
<td>100 Hz</td>
<td>parallel, distrib</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>$10^{14}$ transistors (CPU)</td>
<td>30W (CPU)</td>
<td>$10^9$ Hz</td>
<td>sequential, centralised</td>
<td>No</td>
<td>A bit ...</td>
</tr>
</tbody>
</table>
Networks in the brain

- Cortex
- Mesencephalus
- Cerebellum
- Bulb

Each part is divided into areas and regions

Each area and region is structured as neural networks
Neurons and synapses

Input  Signal  Output
Synaptic learning

- The brain learns, thanks to the reinforcement of synaptic connections
- It is a long-term strengthening of the connections
In summary…

• We are interested in the following properties:
  – Distributed and parallel computations
  – Dense connections of the basic units
  – Connections can be modified by experience
  – Learning is constant and (usu.) un-supervisioned
  – Learning is based on local information
  – Graceful degradation
Why artificial neural networks?
The purpose of models

• Understand data and observations (statistical analyses and simulation)
• Foresee future outcomes (forecasting)
• Approximation of complex systems by means of simpler description (models and metamodels)
• Rules to control real world systems (control)
Model “spectral analysis”

- Process control
- Hydrological models
- Ecological models
- Socio-political models

- Electrical circuits
- Nuclear reactors
- Air pollution
- Economic models

uncertainty
Model “colour”

- Physical based models
  Partial differential equations

- Lumped parameters models
  Ordinary differential equations

- Black-box models
  Input output equations

# of state variables
Artificial neural networks?

• Easy to set-up (but not to train...)
• Process quickly large amounts of data
• Very flexible

Are they the ultimate substitute to conceptual/physical-based models?

No
Artificial Neural Networks!

• ANN do work if:
  • Input data are “dirty”
  • we have loads of training data as input/output couples
  • training data contain errors
  • training must be done quickly
  • it is not important to understand “why”
Neural networks at work

• ANN to learn and model how the real neural networks work
• ANN in machine learning tasks (classification)
• ANN in data modelling
  • forecast
  • analysis
  • meta-models
ANN applications

• Learning and classification:
  – Voice and hand-writing recognition, robot navigation, game playing (chess, backgammon)

• Forecast and estimation:
  – Financial forecasts, environmental forecasts

• Approximation:
  – Any given mathematical function can be approximated
Structure of Artificial Neural Networks
A simple model of an artificial neuron

\[ y = f(Wx + b) \]

- \( w_{ij} \): weights
- \( F \): activation function
- \( N \): net input
- \( b \): bias

inputs -> neuron -> output
Linear regression

74 data items:
- $x$: weight
- $y$: fuel

Given the weight of a vehicle, can we foresee its fuel consumption?

A very simple model: $y = w_1 x + w_2$

How did we get $w_1$ and $w_2$???
Minimise the error function

We minimised the error function

\[ E^P = \frac{1}{2} \sum (t_o - y_o)^2 \]

It was a linear regression and we used the least square method.

What happens when the model is not linear?
Gradient descent

Slope of $E$ positive
=> decrease $W$

Slope of $E$ negative
=> increase $W$
Gradient descent

• Choose a random value for parameters
• Compute the gradient $G$ of the error w.r.t each parameters
• Update parameters in the direction of the greatest decrement of the error $(-G)$
• Repeat the two steps above until $G$ converges to 0
It’s a neural network!

\[ y = w_1 x + w_2 \]
Feed-forward networks
A brief description of feed-forward nets

The activation function is usually a sigmoid

Network input

\[ n_i = \sum_{j=1}^{r} w_{ij} x_j \]

Activation

\[ y_i = f_i (n_i) = f_i \left( \sum_{j=1}^{r} w_{ij} x_j \right) \]
Structure of a multi-layer network

- $r$ input elements
- $s$ neurons in the hidden layer
- $t$ neurons in the output layer

$$\frac{N}{(r+2)s + 1} > 10$$

N training samples
Forward pass

\[ y_i^{(L+1)} = f_i^{(L)}(n_i^{(L)}) = f_i^{(L)} \left( \sum_{j=1}^{r} w_{ij}^{(L)} y_j^{(L-1)} \right) \quad n_i^{(L)} = \sum_{j=1}^{r} w_{ij}^{(L)} y_j^{(L-1)} \]
• Init weights to (small) random values
• repeat until converged
  • for each weight $w_{ij}$ let $\Delta w_{ij} := 0$
• for each training couple
  • feed $x$ as input
  • compute $y$
  • for each weight $w_{ij}$ compute $\Delta w_{ij}$
• update each weight $w_{ij}$ $w_{ij} := w_{ij} - \alpha \Delta w_{ij}$

Learning rate
Backpropagation

\[ \Delta w_{ij} \]

- the change in weight values is computed back propagating values in the network

\[ \Delta w_{ij} = - \frac{\partial E}{\partial w_{ij}} \]

weight gradient

- where

\[ E^P = \frac{1}{2} \sum_o (t_o - y_o)^2 \]
How to compute the gradient

\[ \Delta w_{ij}^{(L)} = - \frac{\partial E}{\partial w_{ij}^{(L)}} = - \frac{\partial E}{\partial n_i} \frac{\partial n_i}{\partial w_{ij}} = \delta_{ij}^{(L)} \text{ error in unit } i \text{ of layer } L = \delta_{ij}^{(L)} y_j^{(L-1)} \]

\[ \frac{\partial}{\partial w_{ij}} \left( \sum_{j=1}^{r} w_{ij} y_j \right) = y_j^{(L-1)} \]
Error in output and hidden layers

In the output layer
\[ \delta_o = t_o - y_o \]

In the hidden layer
\[ \delta_i^{(L)} = -\frac{\partial E}{\partial n_i^{(L)}} = -\sum_k \frac{\partial E}{\partial n_k^{(L+1)}} \frac{\partial n_k^{(L+1)}}{\partial y_i^{(L)}} \frac{\partial y_i^{(L)}}{\partial n_i^{(L)}} \]

\[ \delta_k^{(L+1)} = \frac{\partial}{\partial y_i^{(L)}} \left( \sum_{i=1}^{r} w_{ki}^{(L+1)} y_i^{(L)} \right) = w_{ki}^{(L+1)} \]

backpropagation from the output layer
In conclusion

• To compute the error for unit $i$ at layer $L$ we must know the error of successive nodes at layer $L+1$

$$\delta_i^{(L)} = -\sum_k \delta_k^{(L+1)} w_{ki}^{(L+1)} f_i'(n_i^{(L)})$$

• The final formula

$$\Delta w_{ij}^{(L)} = \delta_i^{(L)} y_j^{(L-1)}$$
Training methods (1)

• Gradient descent

• Gradient descent with momentum:
  • the weight update depends also from the previous update

\[ w_{t+1} = w_t - \alpha \Delta w_t \]
Training methods (2)

- Conjugate Gradient
- Quasi-Newton methods
  \[ w_{t+1} = w_t - H^{-1} \Delta w_t \]  
  \( H: \) matrix of second derivatives of the performance index
- Levenberg-Marquardt
- Stochastic Meta Descent
Training methods

- Levenberg-Marquardt
- LM is a Newton approximation. It stabilises inversion by the addition of a multiple of the eye matrix in order to obtain a definite positive matrix

\[
H = J \cdot J^{-1} \quad \Delta w_t = J^T e
\]

\[
w_{t+1} = w_t - \left[ J \cdot J^{-1} + \mu I \right]^{-1} J^T e
\]
Overfitting

Real data from \( y = h(x) + \varepsilon \)

Approximation with \( g(x) \) too few parameters

Approximation with \( m(x) \) too many parameters
Prevent overfitting

- Separate data in two sets
  - Training
  - Validation
- Do not update weights during validation

Other methods are based on weight decay (forget learning) and training with artificial noise
A bit of gardening: pruning

- How many hidden layers?
  - How many layers

- a) thumb rules

- b) pruning and growth

\[ \frac{N}{(k+2)^n + 1} > 10 \]
In conclusion...
ANNs as general function approximators

• It can be shown that ANN are general function approximators (Hornik 1989, Kreinovich 1991).

• Provided we use a sufficient number of hidden layers, an ANN can approximate any function

• We can approximate a highly non-linear function $F(x)$ with a neural network $\hat{F}(x, \theta)$
ANN or statistical models?

- Learning from “noisy” data → Statistical inference
- “feedforward” networks, no hidden layers → generalised linear models
- “feedforward” networks, one hidden layer → projection pursuit regression
- Probabilistic networks → kernel discriminant analysis.
- Kohonen maps → k-means cluster analysis.
- Hebbian learning → principal component analysis
Remarks

• Neural networks require the same hypotheses of the corresponding statistical models

• Neural network users are often too lazy to investigate the consequences of these hypotheses

“… it is often better to get an approximate solution to a real problem rather than an exact solution to an oversimplified one”
(Tibshirani, 1994)
Acknowledgements

• Part of these slides were developed by Nic Schraudolph (ETH) and Fred Cummins (Uni Dublin) in 1999
Case studies
ANN in environmental modelling

• Data modelling
  • rainfall and snowfall
  • floods
  • rainfall/runoff
  • water quality
• Meta-modelling
Two case studies

• Data modelling
  • A rainfall-runoff model
• Metamodelling
  • Water height forecasting, Ticino river, Pavia
Rainfall-runoff

- Objectives:
  - ARMA vs ANN comparison
  - Train and validate an ANN for R/R
- Application: high Wye (Galles) and Ouse (Yorkshire)
Data selection

• Build the training data set
  • runoff time series at the measurement point (to get the next runoff value)
  • time series of the differences of flow rates at time $t$ and $t-1$ (to get the next flow rate change)

• **Hint:** use both data sets to let “emerge” what’s relevant
Data setup

Available data: hourly time step from 1984 to 1986

| C | L | O | C | K | F | L | O | W | F | L | O | W | F | L | O | W | D | I | F | F | D | I | F | F | D | I | F | F | D | I | F | W |
| sin | cos | t-6 | t-5 | t-4 | t-3 | t-2 | t-1 | t-6 | t-5 | t-4 | t-3 | t-2 | t-1 | t |

5 - 6 points are sufficient to define the ascending part of the unit hydrograph (Johnstone & Cross, 1949)
Network structure

- Choice of the structure
- “trial and error” for two configurations
- 12 nets for each configuration
  # nodes hidden layer = n*6 with n=1..12

1 hidden layer

2 hidden layers
Training

- Setup: random values of weights (+1, -1)
- Termination: a given number of epochs
- Evaluation: least squares of training error
Training results

• A single hidden layer is enough

• Two alternative structures:
  • 14:12:1
  • 14:6:1

choose the simplest structure
Sample output

Training on 1985 data
One step forecast
Horizon: 500 hours
14.00 : 6 December - 10.00 : 26 December 1986
A metamodelling example

The old town in Pavia is exposed to periodical floodings of the Ticino river.

Floodings are due to:
• Ticino floods
• Po backflows

Lake Maggiore regulation must take into account downstream flooding risks.
The proposed solution

• A hydraulic model (De Saint Venant) computes the levels and flow rates along the Ticino reach between the Miorina Dam and Pavia

• Two main problems to use this model in optimisation
  • simulation times are too long
  • it requires 3-hourly flows

• A solution
  • use an ANN as a meta-model to be incorporated in the optimisation routine
Available information

Lake Maggiore

Sesto Calende, Miorina Dam

Ticino river

Pavia, Ponte Coperto

Po river

Piacenza
Data selection

- $h_t^{Pv}$ Ticino level at Ponte Coperto measured at 12:00, on day $t$ (1943-1998 Annali Idrologici)

- $r_t$ mean daily flow rate of Ticino river at Miorina Dam, from 8:00 on day $t-1$ to 8:00 of day $t$ (1943-1998 Cons. Ticino)

- $q_t$ mean daily flow rate of Po river in Piacenza, from 0:00 to 24:00 of day $t$ (1926-1997 Enel Ricerca)
Casual links

Miorina

100 Km
28 h (1 m/s)

Pavia
30 + 8 Km
14 h (0.75 m/s)
4 h (0.6 m/s)

Piacenza

$t-1$ $t$ $t+1$

$r_{t-1}$ $r_t$ $h_t$

$q_t$ $q_{t+1}$

30 + 8 Km
100 Km

$14$ h (0.75 m/s)
$4$ h (0.6 m/s)
The input data sets

- A) $r_{t-1}, r_t, q_t^{Pc}$ averaging the flow rate at Miorina
- B) $r_t, q_t^{Pc}, q_{t+1}^{Pc}$ averaging the flow rates in Piacenza
- C) $r_{t-1}, r_t, q_t^{Pc}, q_{t+1}^{Pc}$ both effects at once
Data pre-processing

- The level in Pavia depends also on Po river level
- The Ticino riverbed is eroded year after year
Average river depth
Data re-scaling

• River depth measures are available only from 1957 to 1972. Elsewhere we have minimal levels

• We re-scale data to use a much bigger data set → 30 flood events for a total of 846 n-uples

• We consider only n-uples where rt >= 750 m3/s → 518 n-uples
Network structure

\[
\frac{N}{(k+2)*n + 1} > 10
\]

- \(N\): n-uples
- \(k\): #inputs
- \(n\): # hidden neurons

3,4 inputs and 6 neurons in the hidden layer
we need at least 310 n-uples for training
1 hidden layer, activation \(\text{tanh}(x)\), linear output
Performance evaluation

- It is more important to have a better description of the flow ascent, rather than a good estimation of the peaks

<table>
<thead>
<tr>
<th>Net Name</th>
<th>Inputs</th>
<th>Neurons</th>
</tr>
</thead>
<tbody>
<tr>
<td>3in_a5</td>
<td>$r_{t-1}, r_t, q_t^{Pc}$</td>
<td>5</td>
</tr>
<tr>
<td>3in_b5</td>
<td>$r_t, q_t^{Pc}, q_{t+1}^{Pc}$</td>
<td>5</td>
</tr>
<tr>
<td>3in_b6</td>
<td>$r_t, q_t^{Pc}, q_{t+1}^{Pc}$</td>
<td>6</td>
</tr>
<tr>
<td>4in_c5</td>
<td>$r_{t-1}, r_t, q_t^{Pc}, q_{t+1}^{Pc}$</td>
<td>5</td>
</tr>
</tbody>
</table>
Evaluation: 3in_a5

<table>
<thead>
<tr>
<th></th>
<th>Addestramento</th>
<th>Validazione</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlazione tra i colmi</td>
<td>0.9816</td>
<td>0.9684</td>
</tr>
<tr>
<td>SQM tra i colmi [m]</td>
<td>0.2596</td>
<td>0.3602</td>
</tr>
<tr>
<td>Correlazione per livelli &gt; 1.39 m</td>
<td>0.8987</td>
<td>0.8259</td>
</tr>
<tr>
<td>Massimi sfasati di un giorno</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Massima differenza fra i colmi [m]</td>
<td>0.8650</td>
<td>1.0490</td>
</tr>
<tr>
<td>Minima differenza fra i colmi [m]</td>
<td>0.033</td>
<td>0.025</td>
</tr>
<tr>
<td>Piene previste correttamente</td>
<td>34</td>
<td>27</td>
</tr>
<tr>
<td>Piene previste correttamente [%]</td>
<td>73.9%</td>
<td>81.8%</td>
</tr>
<tr>
<td>False piene</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Piene non previste</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>Allarmi previsti a 0.89 m</td>
<td>68</td>
<td>42</td>
</tr>
<tr>
<td>Allarmi previsti a 0.89 m [%]</td>
<td>90.7%</td>
<td>84.0%</td>
</tr>
<tr>
<td>falsi allarmi a 0.89 m</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Allarmi non previsti a 0.89 m</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Allarmi previsti a 0.39 m</td>
<td>111</td>
<td>72</td>
</tr>
<tr>
<td>Allarmi previsti a 0.39 m [%]</td>
<td>91.0%</td>
<td>91.1%</td>
</tr>
<tr>
<td>falsi allarmi a 0.39 m</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>Allarmi non previsti a 0.39 m</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>km² allagati storici/km² allagati previsti</td>
<td>1.1483</td>
<td>0.9284</td>
</tr>
</tbody>
</table>

Good correlation between peaks, bad forecast of flooded area
Evaluation: 3in_b5

<table>
<thead>
<tr>
<th></th>
<th>Addestramento</th>
<th>Validazione</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlazione tra i colmi</td>
<td>0.9825</td>
<td>0.9669</td>
</tr>
<tr>
<td>SQM tra i colmi [m]</td>
<td>0.2283</td>
<td>0.3686</td>
</tr>
<tr>
<td>Correlazione per livelli &gt; 1.39 m</td>
<td>0.9327</td>
<td>0.8759</td>
</tr>
<tr>
<td>Massimi sfasati di un giorno</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Massima differenza fra i colmi [m]</td>
<td>0.5320</td>
<td>0.6790</td>
</tr>
<tr>
<td>Minima differenza fra i colmi [m]</td>
<td>0.003</td>
<td>0.001</td>
</tr>
<tr>
<td>Pieni previste correttamente</td>
<td>37</td>
<td>27</td>
</tr>
<tr>
<td>Pieni previste correttamente [%]</td>
<td>80.4%</td>
<td>81.8%</td>
</tr>
<tr>
<td>False pieni</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Pieni non previste</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>Allarmi previsti a 0.89 m</td>
<td>79</td>
<td>44</td>
</tr>
<tr>
<td>Allarmi previsti a 0.89 m [%]</td>
<td>94.0%</td>
<td>88.0%</td>
</tr>
<tr>
<td>Falsi allarmi a 0.89 m</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Allarmi non previsti a 0.89 m</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Allarmi previsti a 0.39 m</td>
<td>113</td>
<td>76</td>
</tr>
<tr>
<td>Allarmi previsti a 0.39 m [%]</td>
<td>92.6%</td>
<td>96.2%</td>
</tr>
<tr>
<td>Falsi allarmi a 0.39 m</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Allarmi non previsti a 0.39 m</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>km² allagati storici/km² allagati previsti</td>
<td>1.1179</td>
<td>0.9457</td>
</tr>
</tbody>
</table>

Improves the flood forecasting ability
A trajectory
In conclusion

• ANN are data hungry
• To find a good structure is often more like an art
• Garbage-in Garbage-out
• Performance evaluation is not trivial
• ANN aren’t a cure for all problems
Knowledge representation

A quick introduction to KR models and techniques
The problem

- store and manipulate knowledge in order to be processed by information systems
- we do not know how knowledge is actually stored (at the meta level) in our brain
Applications

- Expert systems
- Machine translation systems
- Information retrieval
- Content management systems
Representation

• Notations
  • RDF (Resource Description Framework)
    • subject-predicate-object
    • adopted by the Semantic Web initiative
  • It is an XML application
Representation

• Artificial languages

• CyCL (part of CyC project)

• OWL (ontology web language)
  • Markup language for publishing and sharing data using ontologies on the Internet.
  • OWL is a vocabulary extension of RDF (the Resource Description Framework) and is derived from the DAML+OIL Web Ontology Language.
  • Part of the semantic web project.
Techniques

• Semantic networks
  • nodes are concepts
  • arcs are relations

• Frames
  • object-oriented approach
  • is-a / part-of
Ontology

• Ontos = Being  Logos = Science
• a conceptual schema, defines
  • entities
  • hierarchically organised
  • their relationships
  • and the governing rules
Ontologies

Concepts are organised in taxonomies, linked by relations and conforming to axioms

Animals
  - Fish
  - Mammals
    - Carnivores
    - Rodents

Axioms can be very general, e.g. "carnivores eat fish"
Well-known ontologies

• WordNet groups English words into synsets
  • 140’000 words, 110’000 synsets, 12 Mbytes (compressed)
  • http://www.cogsci.princeton.edu/~wn/

• Various ontologies implement the DublinCore meta-data initiative
  • interoperable online meta-data standard
  • http://dublincore.org/
Rule-based reasoning
Expert Systems

Acknowledgements: Miquel Sànchez i Marrè
Expert Systems

• A definition

  • an expert system is a computer program that solves complex problems, which are usually heuristically solved by a human expert with a very high knowledge of the problem domain
Expert knowledge

Data and Algorithms = Programs

Knowledge and Inferences = ES
ES classification

- Hayes-Roth, 1983
- Interpretation: describe what is happening from data
- Prediction: describe what could happen
- Diagnosis: identify the causes of a malfunction
- Design: generate objects satisfying design constraints
ES classification

• Planning: identify a course of actions to achieve an objective

• Monitoring: examine a system behaviour over time

• Control: define the system behaviour over time
Steps in knowledge engineering

1. Identification
2. Conceptualisation
3. Formalisation
4. Implementation
5. Test
Identification

- is an ES the right answer? Can we build it?
- Find the expert knowledge sources
- What is exactly the problem we want to solve?
Conceptualisation

• Identify facts and relationships
• Identify evidence, hypothesis and actions
• Decompose the problem in subproblems
• Analyse the flow of knowledge and the reasoning mechanisms
Formalisation

• What reasoning task? (Select among classification/diagnosis/planning/control/etc.)

• What is the search space? (Define the variable domains)

• What is the solution strategy? (Classification, Proof)

• How do we deal with uncertainty and completeness?
Implementation and test

- Implement the knowledge base
  - rules
  - facts
- Test on a set of known cases
The ES architecture

- Knowledge base
  - contains facts and rules, represented using a formalism (semantic nets, frames,...)

- Inference engine
  - applies rules to facts and produces new knowledge
Inference Rules

• Given a set of premises, an inference rule carries out a conclusion

• Modus Ponens: if P then Q. P, therefore Q.

• P is the premise, Q is the conclusion
Representing a rule

- Rule ID
- Premises
  - propositions
  - first-order predicates
- Conclusions
  - new facts
  - actions
  - computations
- Rule certainty

Rule R023

Wind
Sunny

Play tennis
80%
Inference engine

- Interprets rules against facts using a strategy
- Strategy
  - detection: finds “firing” rules
  - selection: solves conflicts
  - application: inference (fw and bw chaining)
Selection

• From the rules identified during the detection stage, find the best one
• Solve conflicts, with various criteria:
  • use the most popular rule
  • most specific/general
  • the most informative (high number of new facts)
  • highest certainty
Application

- Execute the selected rule
- Add new facts into the rule base
- Propagate rules
- Check for termination (if all rules have been applied, or a conclusion has been reached)
Forward chaining

• Based on modus ponens
• Start from initial state and fire rules in succession
• From evidence to hypothesis verification
## An example

<table>
<thead>
<tr>
<th>Knowledge base</th>
<th>Facts</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: $A \land B \land C \rightarrow D$</td>
<td>A</td>
</tr>
<tr>
<td>R2: $A \land E \land F \rightarrow G$</td>
<td>E</td>
</tr>
<tr>
<td>R3: $B \land C \land D \rightarrow H$</td>
<td>B</td>
</tr>
<tr>
<td>R4: $E \rightarrow C$</td>
<td></td>
</tr>
<tr>
<td>R5: $A \land H \rightarrow F$</td>
<td></td>
</tr>
<tr>
<td>R6: $A \land C \rightarrow H$</td>
<td></td>
</tr>
</tbody>
</table>

Obj: G
I: A, E, B
2: A, E, B, C (R4)

R1

R3

R5

R6

A, E, B, C, D, H

A, E, B, C, D, H, F

A, E, B, C, D, H, F, G

A, E, B, C, H
Forward chaining: pros and cons

• **Cons:**
  - the strategy to solve conflicts is critical
  - combinatorial explosion of the search tree

• **Pros:**
  - intuitive representation of knowledge processing
Backward chaining

• Uses induction

• if B is true, and B depends from A, then A is true

• Objective drive. We start from an hypothesis and we try to validate it, given the evidence.
An example

<table>
<thead>
<tr>
<th>Knowledge base</th>
<th>Facts</th>
<th>Obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1 (A \land B \rightarrow C)</td>
<td>A</td>
<td>H</td>
</tr>
<tr>
<td>R2 (C \rightarrow D)</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>R3 (E \land F \rightarrow G)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R4 (A \rightarrow E)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R5 (D \rightarrow G)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R6 (A \land G \rightarrow H)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The fact F true is not in the base. **backtrack**
The hypothesis is satisfied.
Case-Based Reasoning

Expanding Instance-based learning

Acknowledgements: Miquel Sànchez i Marrè
Why CBR?

- Experts find it difficult to formalise their knowledge in rules
- The abstraction process, necessary to create generic rules, valid over a number of instances, is very difficult
Because...

- CBR uses “reasoning by analogy” using past experience in its original form.
- There is no need to translate experience into rules.
- (think of teaching your language to someone by providing a set of grammar rules... hard, isn’t it?)
A definition

• “.... transferring knowledge from past problem solving episodes to new problems that share significant aspects with corresponding past experience and using the transferred knowledge to construct solutions to new problems.”

(Carbonell, 1986)
The structure

Case Library

Case #34

Problem

Solution

New Problem

New Solution
CBR foundations

• Our general knowledge about situations is recorded as scripts [Schank & Abelson, 1977]
• Cognitive model is the Theory of Dynamic Memory [Schank, 1982]
  • Indexing is the key to use experience in understanding
  • Remembering, understanding, experiencing, and learning cannot be separated from each other
  • Human memory is dynamic, and change as a result of its experiences
• CBR derives from a view of understanding problem-solving as an explanation process [Riesbeck & Schank, 1989]
The CBR loop

Retrieve

new case

retrieved cases

CASE LIBRARY
DOMAIN KNOWLEDGE

Eval

adapted solution

Evaluated solution (fail/success)

Learn
case to store

Learn

Adapt

best case

Adapt

Eval
CBR components

- Cases: simple or structured
- Case library: flat or hierarchic memory
- Retrieval techniques: search in library index or similarity evaluation
- Adaptation methods
- Evaluation methods
- Learning algorithms
Examples

• CHEF [Hammond, 1986, 1989], a case-based planner for recipe creation
• CASEY [Koton 1988, 1989], a case-based diagnosis program to diagnose a causal explanation of the patient disorders
• JULIA [Hinrichs 1988-1992], a case-based designer in the domain of meal planning
• HYPO [Ashley, 1990], a case-based interpretive program that works in the domain of law
• PROTOS [Bareiss, 1989], a case-based classification program for audiological disorders
• CLAVIER [Hennessy & Hinkle, 1992], a case-based program for configuration of the layout of composite airplane parts for curing in autoclave
Representing cases

- Attribute-value representation: a case is a set of features
  - case identifier
  - derivation of the case
  - description of the problem
  - diagnostic of the problem
  - solution to the problem
  - evaluation of the solution (success/failure)
  - utility measure
  - other relevant information
- Structured representation: a case is a structure relating features and other elements
  - tree or network
Case retrieval

• The problem is that we must retrieve elements in the case library which are similar to the case at hand

• We need to define a similarity metric

• As we did in instance-based learning

\[
dist(C_i, C_j) = \sum_{k=1}^{n} w_k \times atr \_ dist(C_{ik}, C_{jk})
\]
Case retrieval

- The efficiency of case retrieval depends strongly on the memory structure

Hierarchic memories
- Difficult to manage
- Quick to look-up
- Heuristic search

Flat memories
- Easy to manage
- Slow to access
- Always find the best
Adaptation

- If the matching between the case under study and the retrieved case is perfect, no adaptation is needed.
- Else, we may need a structural adaptation based on transformation and/or substitution of some case elements.
Evaluation

• Three alternatives to evaluate the quality of the found solution
  • implement in the real world
  • ask a human expert
  • perform a simulation
When to use CBR

- Lots of data are available
- Experts tend to focus on particular situations
- Experience as valuable as textbook rules
- Partial domain knowledge
- Many exceptions to rules
Some CBR application areas

• Fault diagnosis
• Medical diagnosis
• Loan evaluation
• Legal reasoning
• Plant control and supervision
CBR advantages

• Easy to interact with the expert to extract cases
• It is possible to set up a self-learning environment
• It deals with exceptional cases
CBR disadvantages

- The structure of the Case Library is not easily readable by a human
- Adaptation functions are very domain-specific
- The system cannot infer on something that has never happened
## Rules vs Cases

<table>
<thead>
<tr>
<th>Rules</th>
<th>Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generic knowledge</strong></td>
<td><strong>Specific knowledge</strong></td>
</tr>
<tr>
<td>Difficult to learn new rules and maintain the KB - static knowledge - no learning</td>
<td>Easy to learn new cases - dynamic knowledge - learning</td>
</tr>
<tr>
<td>Difficult to extract knowledge from the expert</td>
<td>Relatively easy to acquire knowledge from experts</td>
</tr>
<tr>
<td>No change in performance</td>
<td>Performance increases over time</td>
</tr>
</tbody>
</table>
End of Part III