Course 395: Machine Learning – Lectures

- Lecture 1-2: Concept Learning (*M. Pantic*)
- Lecture 3-4: Decision Trees & CBC Intro (*M. Pantic*)
- Lecture 5-6: Artificial Neural Networks (*THs*)
- ➢ Lecture 7-8: Instance Based Learning (*M. Pantic*)
- Lecture 9-10: Genetic Algorithms (*M. Pantic*)
- Lecture 11-12: Evaluating Hypotheses (*THs*)
- Lecture 13-14: Guest Lectures on ML Applications
- Lecture 15-16: Inductive Logic Programming (*S. Muggleton*)
- Lecture 17-18: Inductive Logic Programming (*S. Muggleton*)

Instance Based Learning – Lecture Overview

- Lazy learning
- K-Nearest Neighbour learning
- Locally weighted regression
- Case-based reasoning (CBR)
- Advantages and disadvantages of lazy learning
- (Example: CBR-based system for facial expression interpretation)

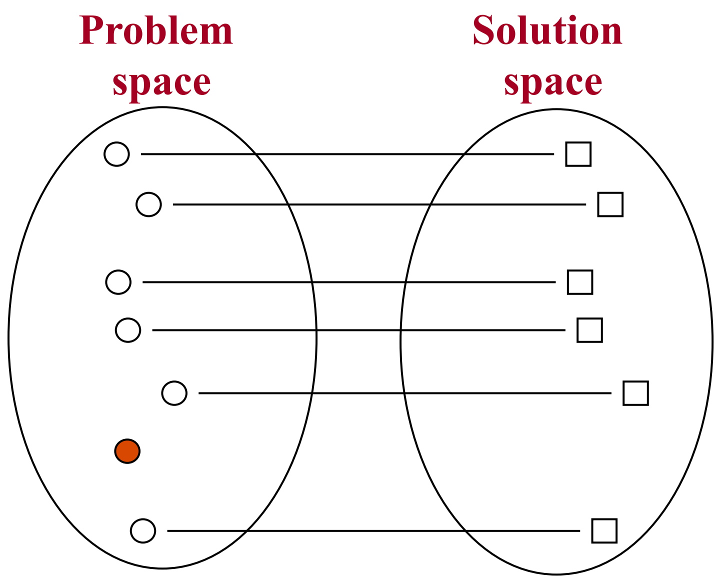
Eager vs. Lazy Learning

- Eager learning methods construct general, explicit description of the target function based on the provided training examples.

≡ one-fits-all ***≡ input independent***

Eager vs. Lazy Learning

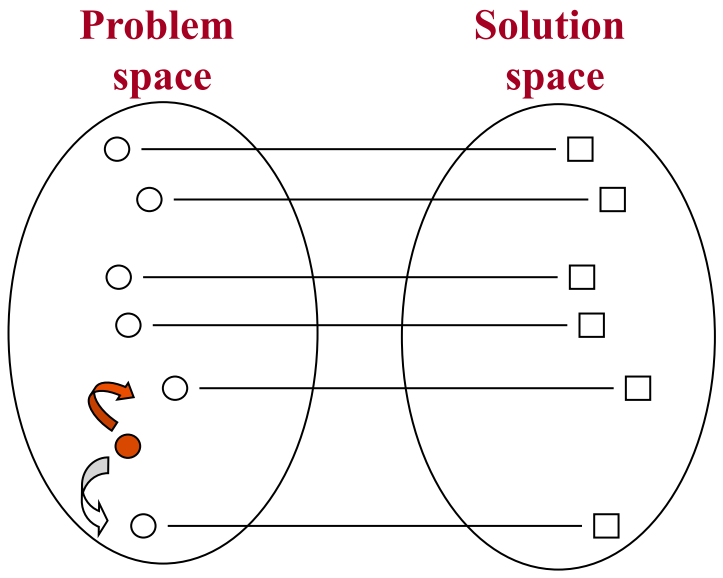
- Eager learning methods construct general, explicit description of the target function based on the provided training examples.
- Lazy learning methods simply store the data and generalizing beyond these data is postponed until an explicit request is made.



1. Search the memory for similar instances
2. Retrieve the related solutions
3. Adapt the solutions to the current instance
4. Assign the value of the target function estimated for the current instance

Eager vs. Lazy Learning

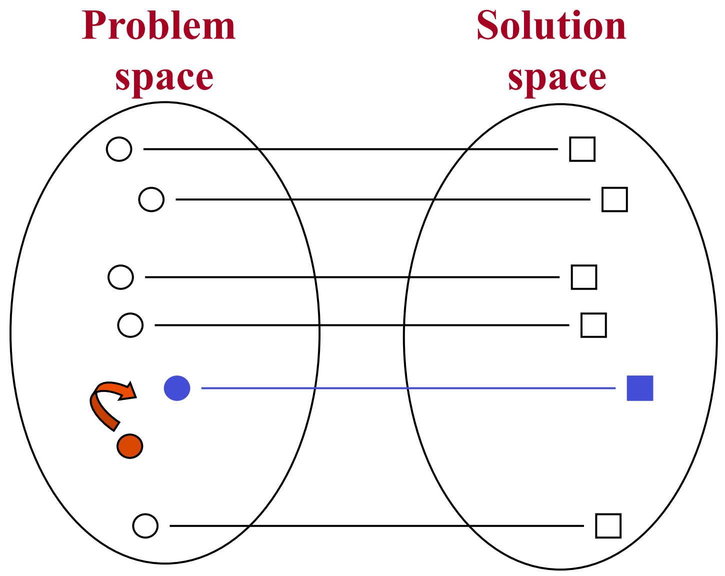
- Eager learning methods construct general, explicit description of the target function based on the provided training examples.
- Lazy learning methods simply store the data and generalizing beyond these data is postponed until an explicit request is made.



1. **Search the memory for similar instances**
2. Retrieve the related solutions
3. Adapt the solutions to the current instance
4. Assign the value of the target function estimated for the current instance

Eager vs. Lazy Learning

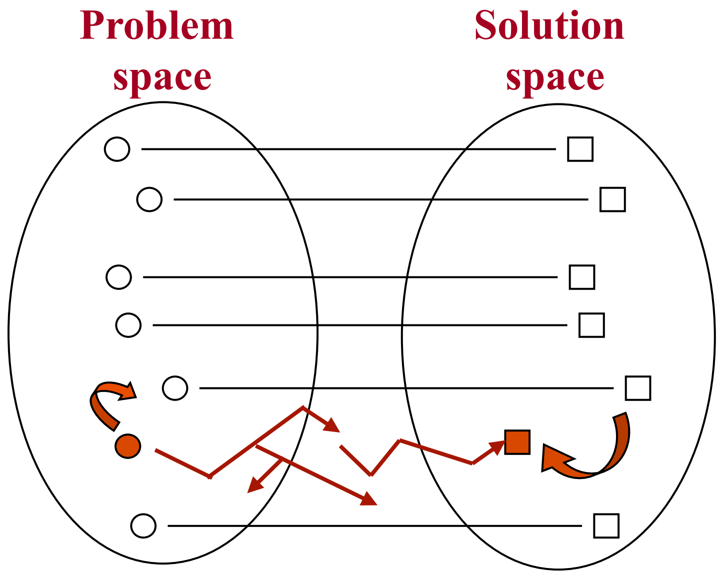
- Eager learning methods construct general, explicit description of the target function based on the provided training examples.
- Lazy learning methods simply store the data and generalizing beyond these data is postponed until an explicit request is made.



1. Search the memory for similar instances
2. **Retrieve the related solutions**
3. Adapt the solutions to the current instance
4. Assign the value of the target function estimated for the current instance

Eager vs. Lazy Learning

- Eager learning methods construct general, explicit description of the target function based on the provided training examples.
- Lazy learning methods simply store the data and generalizing beyond these data is postponed until an explicit request is made.

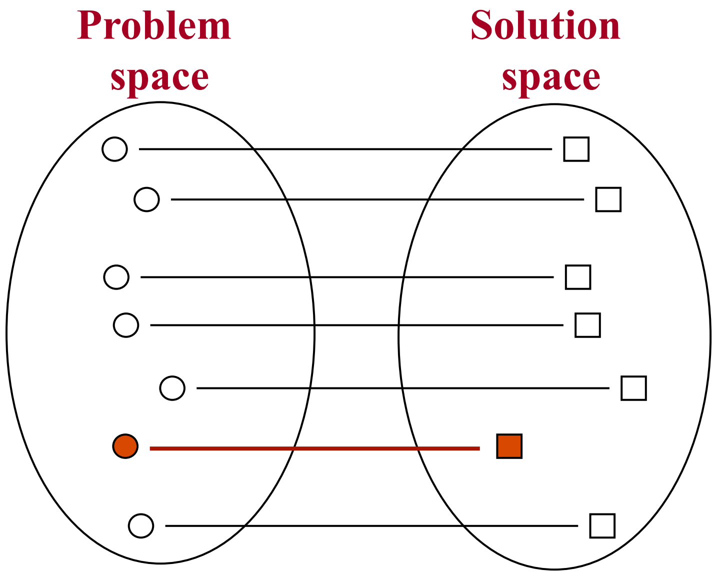


1. Search the memory for similar instances
2. Retrieve the related solutions
3. **Adapt the solutions to the current instance**
4. Assign the value of the target function estimated for the current instance

Eager vs. Lazy Learning

- Eager learning methods construct general, explicit description of the target function based on the provided training examples.

- Lazy learning methods simply store the data and generalizing beyond these data is postponed until an explicit request is made.



1. Search the memory for similar instances
2. Retrieve the related solutions
3. Adapt the solutions to the current instance
4. **Assign the value of the target function estimated for the current instance**

Eager vs. Lazy Learning

- Eager learning methods construct general, explicit description of the target function based on the provided training examples.
- Lazy learning methods simply store the data and generalizing beyond these data is postponed until an explicit request is made.
- Lazy learning methods can construct a different approximation to the target function for each encountered query instance.
- Eager learning methods use the same approximation to the target function, which must be learned based on training examples and before input queries are observed.

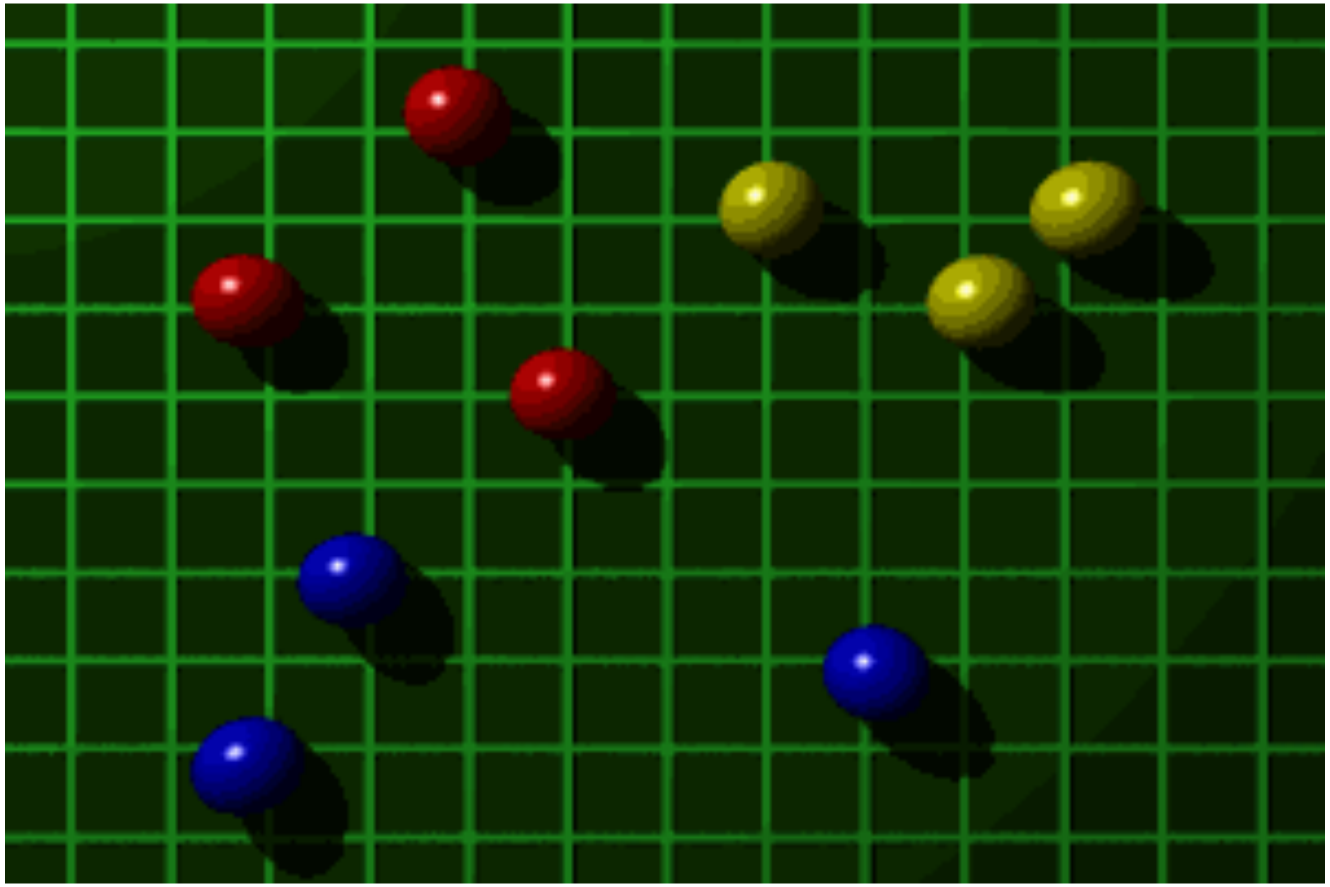
Eager vs. Lazy Learning

- Eager learning methods construct general, explicit description of the target function based on the provided training examples.
- Lazy learning methods simply store the data and generalizing beyond these data is postponed until an explicit request is made.
- Lazy learning methods can construct a different approximation to the target function for each encountered query instance.
- Eager learning methods use the same approximation to the target function, which must be learned based on training examples and before input queries are observed.

Lazy learning is very suitable for complex and incomplete problem domains, where a complex target function can be represented by a collection of less complex local approximations.

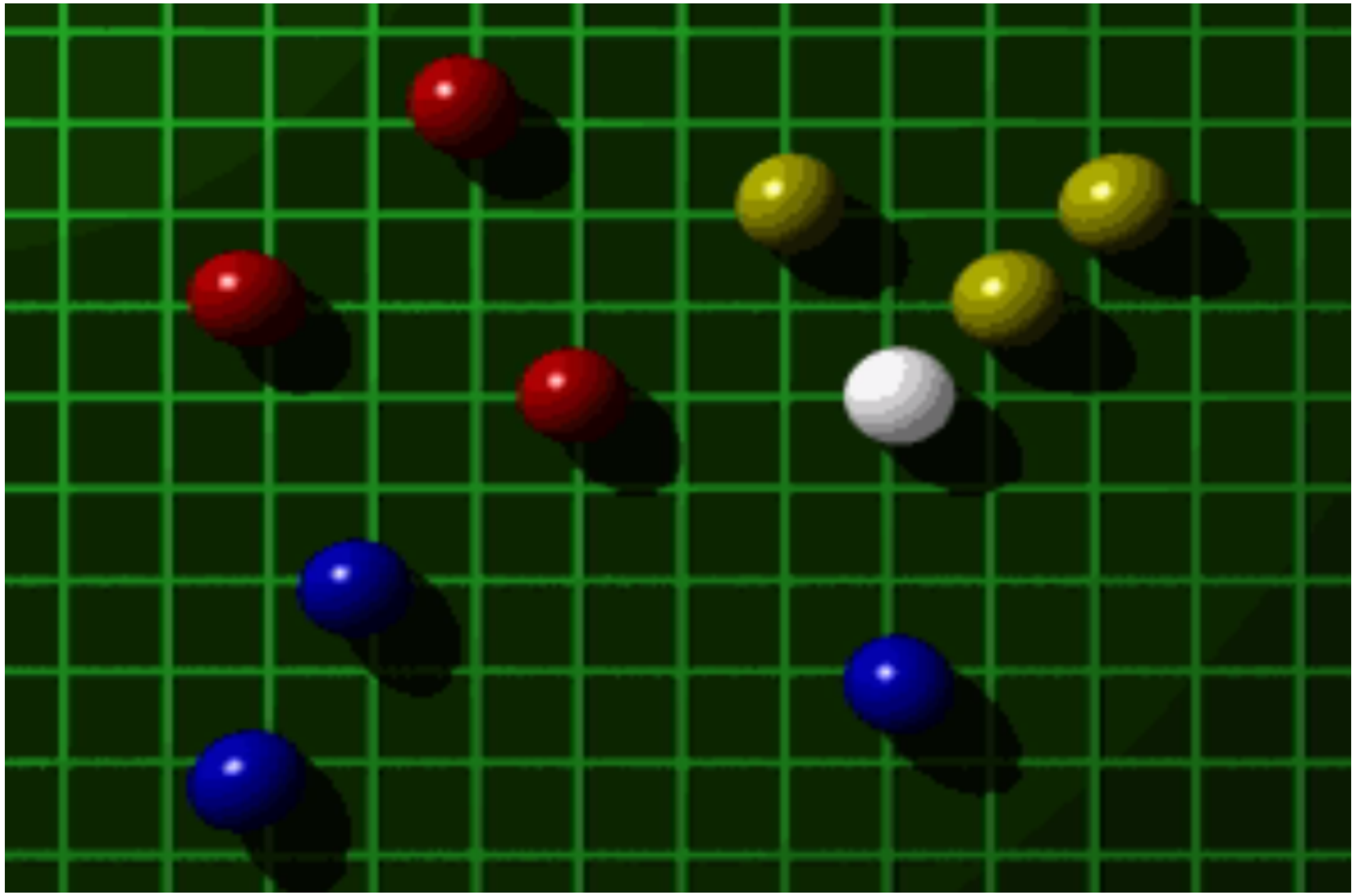
k-Nearest Neighbour Learning

The main idea behind *k*-NN learning is so-called ***majority voting***.



k-Nearest Neighbour Learning

The main idea behind *k*-NN learning is so-called ***majority voting***.



k-Nearest Neighbour Learning

- Given the target function $V: X \rightarrow C$ and a set of n already observed instances (x_i, c_j), where $x_i \in X$, $i = [1..n]$, $c_j \in C$, $j = [1..m]$, $V(x_i) = c_j$, k-NN algorithm will decide the class of the new query instance x_q based on its k nearest neighbours (previously observed instances) x_r, r = $[1..k]$, in the following way:

$$V(x_q) \leftarrow c_l \in C \quad \leftrightarrow \quad (\forall j \neq l) \ \sum_r E(c_l, V(x_r)) > \sum_r E(c_j, V(x_r)) \text{ where}$$

$$E(a, b) = 1 \text{ if } a = b \text{ and } E(a, b) = 0 \text{ if } a \neq b$$

- The nearest neighbours of a query instance x_q are usually defined in terms of standard Euclidean distance:

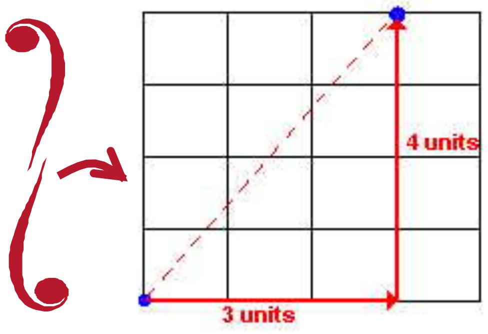
$$d_e(x_i, x_q) = \sqrt{\{\sum_g (a_g(x_i) - a_g(x_q))^2\}}$$

where the instances $x_i, x_q \in X$ are described with a set of $g = [1..p]$ arguments a_g

k-Nearest Neighbour Learning

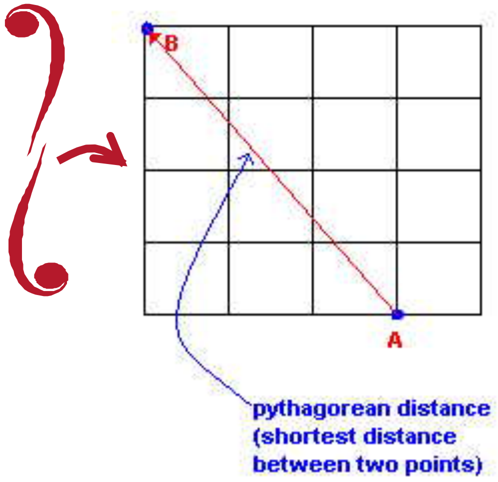
- Distance between two instances $x_i, x_q \in X$, described with a set of $g = [1..p]$ arguments a_g, can be calculated as:
  - City-block (Manhattan) distance (L1-norm):

    $$d_e(x_i, x_q) = \sum_g |a_g(x_i) - a_g(x_q)|$$



  - Euclidean distance (L2-norm):

    $$d_e(x_i, x_q) = \sqrt{\{\sum_g (a_g(x_i) - a_g(x_q))^2\}}$$

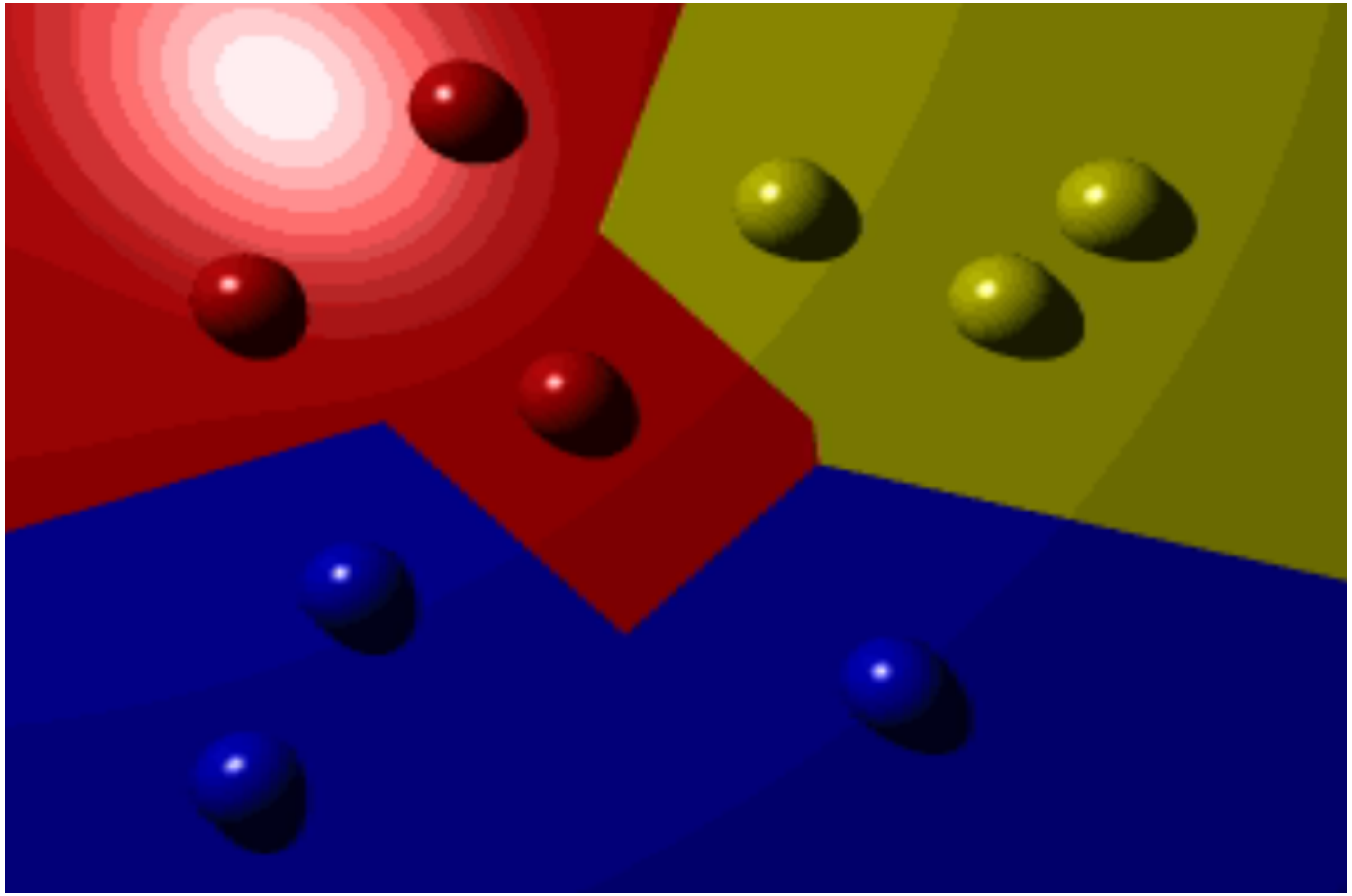


  - Chebyshev distance (L-infinity-norm):

    $$d_e(x_i, x_q) = max_g |a_g(x_i) - a_g(x_q)|$$

k-Nearest Neighbour Learning

For $k = 1$, the decision surface is a set of polygons (***Voronoi diagram***), completely defined by previously observed instances (training examples).



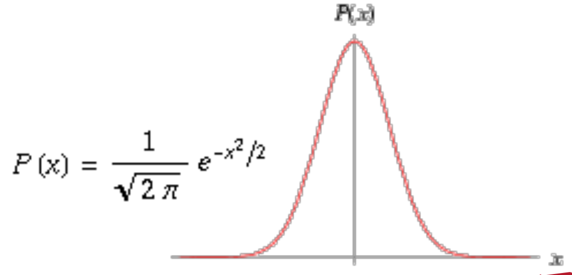
k-Nearest Neighbour Learning

- The nearest neighbours (previously observed instances) x_r, r = [1..k], of a query instance x_q are defined based on a distance $d(x_r, x_q)$ such as the Euclidian distance.

- A refinement of the k-NN algorithm: assign a weight w_r to each neighbour x_r of the query instance x_q based on the distance $d(x_r, x_q)$ such that $(d(x_r, x_q)\downarrow \leftrightarrow w_r\uparrow)$

- **Distance-weighted *k*-NN algorithm**: Given the target function $V: X \rightarrow C$ and a set of n already observed instances (x_i, c_j), where $x_i \in X$, $i = [1..n]$, $c_j \in C$, $j = [1..m]$, $V(x_i) = c_j$, distance weighted k-NN algorithm will decide the class of the query instance x_q based on its k nearest neighbours x_r, r = [1..k], in the following way:

$$V(x_q) \leftarrow c_l \in C \quad \leftrightarrow \quad (\forall\, j \neq l)\ \textstyle\sum_r w_r \cdot E(c_l, V(x_r)) > \sum_r w_r \cdot E(c_j, V(x_r)) \text{ where}$$



$E(a, b) = 1$ if $a = b$, $E(a, b) = 0$ if $a \neq b$, and

$$\boxed{w_r = 1 / (d(x_r, x_q))^2}$$

any other measure favouring the votes of nearby neighbours will do (e.g. Gaussian distribution)

k-Nearest Neighbour Learning: Remarks

- By the distance-weighted *k*-NN algorithm, the value of *k* is of minor importance as distant examples will have very small weight and will not greatly affect the value of $V(x_q)$.

- If $k = n$, where n is the total number of previously observed instances, we call the algorithm a global method. Otherwise, if $k < n$, the algorithm is called a local method.

- ***Advantage*** – Distance-weighted *k*-NN algorithm is robust to noisy training data: it calculates $V(x_q)$ based on a weighted $V(x_r)$ values of *all k* nearest neighbours x_r, effectively smoothing out the impact of isolated noisy training data.

- ***Disadvantage*** – All *k*-NN algorithms calculate the distance between instances based on *all* attributes $\rightarrow$ if there are many irrelevant attributes, instances that belong together may still be distant from one another.

- ***Remedy*** – weight each attribute differently when calculating the distance between two instances

Locally Weighted Regression

- Locally weighted regression is a most general form of *k*-NN learning. It constructs an explicit approximation to target function V that fits the training examples in the local neighbourhood of the query instance x_q.

- *Local* – V is approximated based only on the data (neighbours) near x_q.
  Weighted – contribution of a datum is weighted by its distance from x_q.
  Regression – refers to the problem of approximating a real-valued target function.

- **Locally weighted regression**:
  target function: $V: X \to C$,
  target function approximation near x_q: $V'(x_q) = w_0 + w_1\, a_1(x_q) + \dots + w_n\, a_n(x_q)$,
  where $x_q \in X$ is described with a set of $g = [1..n]$ arguments a_j
  training examples: set of k nearest neighbours x_r, $r = [1..k]$, of the query instance x_q,
  learning problem: learn the most optimal weights w given the set of training examples
  learning algorithm (distance-weighted gradient descent training rule):

$$\Delta w_j = \eta \cdot \textstyle\sum_r \{ \boxed{K(d(x_r, x_q))} \cdot (V(x_r) - V'(x_r)) \cdot a_j(x_r) \}$$

→ function of distance that determines weights of x_r

Instance Based Learning – Lecture Overview

- Lazy learning
- K-Nearest Neighbour learning
- Locally weighted regression
- ➢ Case-based reasoning (CBR)
- Advantages and disadvantages of lazy learning
- (Example: CBR-based system for facial expression interpretation)

Case Based Reasoning (CBR) – Schank's Theory

- The work of Roger Schank, inspired by findings in cognitive sciences on human reasoning and memory organization, is held to be the origin of CBR.
- Human knowledge about the world is organized in memory packets holding similar concepts and/or episodes that one experienced.
- If a memory packet contains a situation when a problem was successfully solved and the person experiences a similar situation, the previous experience is recollected and the same steps are followed to reach a solution.

Rather than following a general set of ruls, reapplying previously successful solution schemes in a new but similar context solves the newly encountered problems.

≡ general approximation to target function

≡ local approximation to target function

Case Based Reasoning (CBR) – Schank's Theory

- The work of Roger Schank, inspired by findings in cognitive sciences on human reasoning and memory organization, is held to be the origin of CBR.
- Human knowledge about the world is organized in memory packets holding similar concepts and/or episodes that one experienced.
- If a memory packet contains a situation when a problem was successfully solved and the person experiences a similar situation, the previous experience is recollected and the same steps are followed to reach a solution.

➔ Rather than following a general set of ruls, reapplying previously successful solution schemes in a new but similar context solves the newly encountered problems.

➔ Lazy learning is much closer to human reasoning model than this is the case with eager learning

Case Based Reasoning (CBR) – Schank's Theory

Schank's memory-based reasoning model:

based on similarity of cases

– The memory of experiences is derived from enumaration of the observed cases, which are stored further in memory organization packets.

– If problems occur to which no specific case can match exactly, reason from more general similarities to come up with solutions. *1*-NN, *otherwise k*-NN
Note: the retrieval is almost never full breadth (exhaustive).

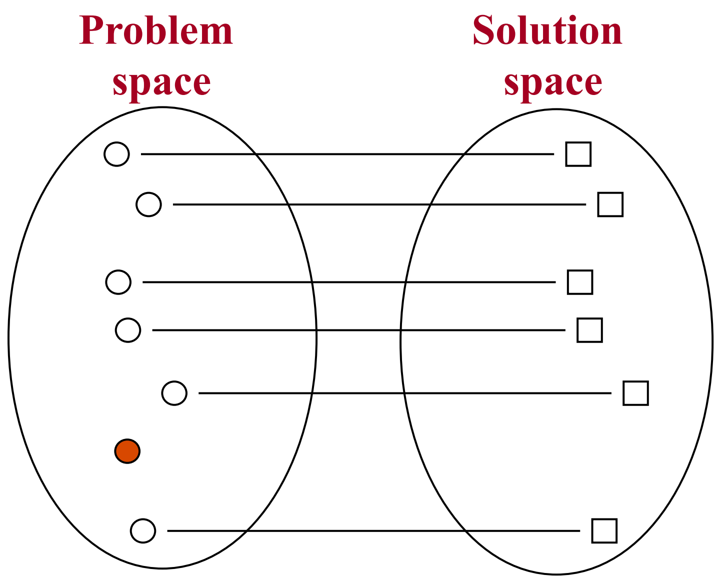
distance measure

– The basis of memory-based model is automatic (online) learning: Memory of experiences is augmented by each novel experience (case). I.e., the process of learning never ceases.

opposite of offline learning (typical for eager learning methods), where the process of learning ceases when the training is completed

Case Based Reasoning (CBR)

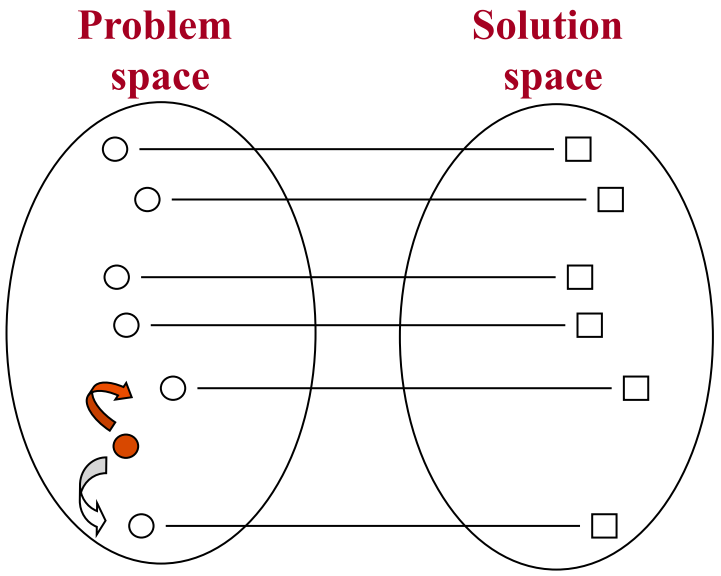
- Schank's memory-based reasoning model is the underlying reasoning model of CBR.
- CBR is *reasoning by remembering*: previously solved cases are used to suggest solutions for novel but similar problems.



1. Search the memory for similar instances
2. Retrieve the related solutions (1- / *k*-NN)
3. Adapt the solutions to the current instance
4. Store the new case in the memory of experiences

Case Based Reasoning (CBR)

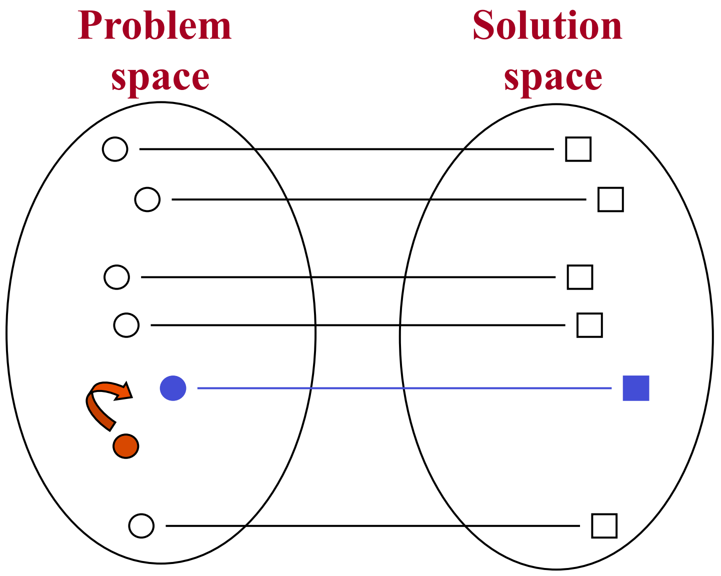
- Schank's memory-based reasoning model is the underlying reasoning model of CBR.
- CBR is *reasoning by remembering*: previously solved cases are used to suggest solutions for novel but similar problems.



1. **Search the memory for similar instances**
2. Retrieve the related solutions (1- / *k*-NN)
3. Adapt the solutions to the current instance
4. Store the new case in the memory of experiences

Case Based Reasoning (CBR)

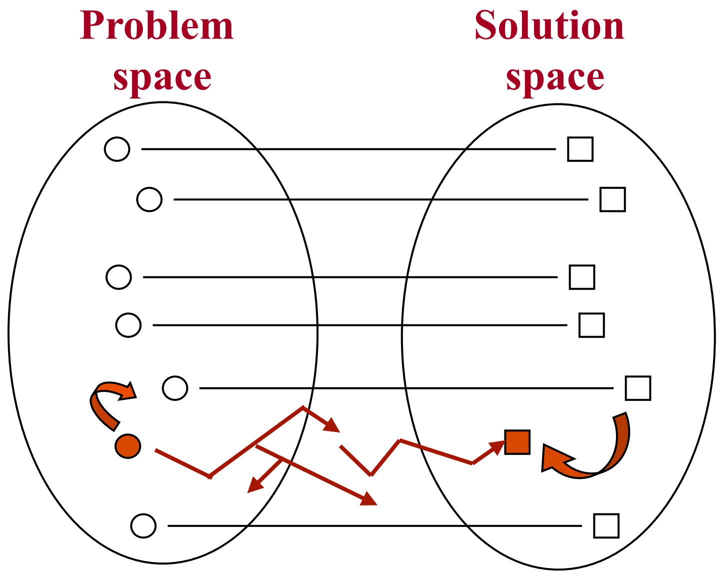
- Schank's memory-based reasoning model is the underlying reasoning model of CBR.
- CBR is *reasoning by remembering*: previously solved cases are used to suggest solutions for novel but similar problems.



1. Search the memory for similar instances
2. **Retrieve the related solutions (1- / *k*-NN)**
3. Adapt the solutions to the current instance
4. Store the new case in the memory of experiences

Case Based Reasoning (CBR)

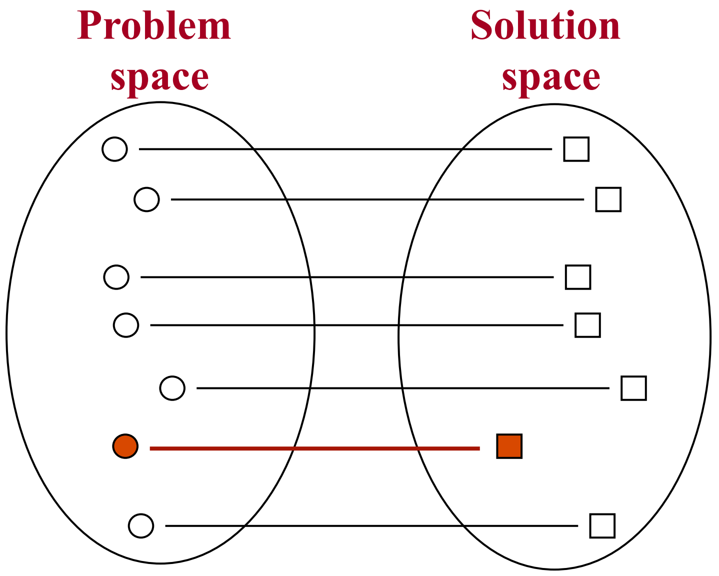
- Schank's memory-based reasoning model is the underlying reasoning model of CBR.
- CBR is *reasoning by remembering*: previously solved cases are used to suggest solutions for novel but similar problems.



1. Search the memory for similar instances
2. Retrieve the related solutions (1- / *k*-NN)
3. **Adapt the solutions to the current instance**
4. Store the new case in the memory of experiences

Case Based Reasoning (CBR)

- Schank's memory-based reasoning model is the underlying reasoning model of CBR.
- CBR is *reasoning by remembering*: previously solved cases are used to suggest solutions for novel but similar problems.



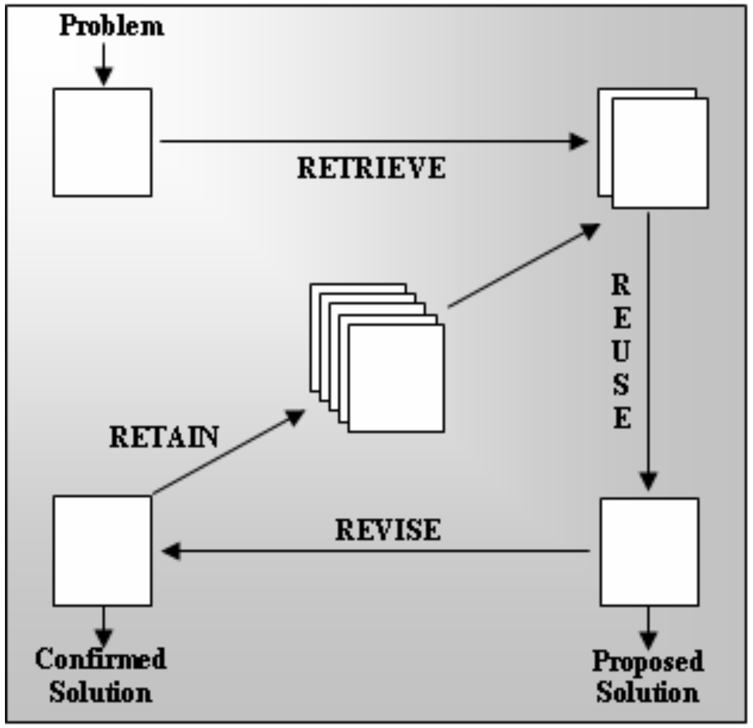
1. Search the memory for similar instances
2. Retrieve the related solutions (1- / *k*-NN)
3. Adapt the solutions to the current instance
4. **Store the new case in the memory of experiences**

Case Based Reasoning (CBR) – Working Cycle

CBR working cycle:

1. RETRIEVE the most similar case(s).
2. REUSE the case(s) to suggest the solution for the current case.
3. REVISE the suggested solution.
4. RETAIN the case by storing it in the memory of experiences.

case base



Case Based Reasoning (CBR) – System Design

- How the cases will be represented?
- How the case base should be organized?
- How the indexing (assigning indexes to cases to facilitate their retrieval) should be defined?
- Which retrieval algorithm is to be used?
- Which (case base) adaptation algorithm is to be used?

Case Based Reasoning (CBR) – Cases

- Cases contain knowledge about previous experiences (solved problems).
- A case is typically composed of the problem description and the problem solution.
- The classic guideline '*the more information it stores, the more useful the case is*', should be applied cautiously.
- Problem description should contain enough data for an accurate and efficient case retrieval. Useful info: retrieval statistics.
- Problem solution can be either atomic (e.g., an action) or compound (e.g., a sequence of actions).
- Cases can be either monolithic (e.g., observation → action) or compound (e.g., a set of observations → a sequence of actions; Note: parts can be processed separately).
- Cases can be represented in various ways: feature vectors, semantic nets, objects, frames, rules...

Cases should be such that an accurate and efficient retrieval is facilitated.

Case Based Reasoning (CBR) – Organisation

- *Flat* Case Base Organisation

  The simplest case base organisation without any specific structure.
  Case retrieval is based on case-by-case search.

- *Clustered* Case Base Organisation

  Cases are stored in clusters of similar cases (as originally proposed by Schank).
  Case retrieval includes finding the appropriate cluster(s) and searching through it for similar cases.
  Case addition / deletion algorithm is more complex than by flat organisation.

- *Hierarchical* Case Base Organisation

  Cases that share *features* are grouped together.
  A semantic network containing interlinked features and categories is used.
  Cases are associated with categories.
  Case retrieval is feature based. It is fast and accurate.
  Reorganisation of the case base may be very complex and difficult.

Case Based Reasoning (CBR) – Indexing

- Case indexing: assigning indexes to cases to facilitate efficient and accurate retrieval of cases from the case base.

- Indexes are defined in terms of features / attributes of cases.

- Indexes should be:
  - *predictive* of the case relevance
  - *recognisable* – it should be clear why they are used
  - *abstract* enough to allow for widening of the case base
  - *discriminative* enough to facilitate efficient and accurate case retrieval

most informative features

trade-off between the generality and specificity of the hypotheses (set of features) to be used for indexing

Case Based Reasoning (CBR) – Retrieval

- Retrieval algorithm should retrieve case(s) most similar to the currently presented problem / situation.

preferred as it results in faster retrieval and more accurate solutions

- *1-NN (k-NN) search*

  A case-by-case search. Search is accurate but highly time consuming.

- *1-NN (k-NN) search* through *preselected* cases

  Uses the indexing structure of the case base to preselect the cases.
  Then, applies *1*-NN or *k*-NN search. Faster than simple case-by-case search.
  It can happen that the best match is not in the preselected cases.

- *1-NN (k-NN) search* through (*preselected* and) *ranked* cases

  Uses the retrieval statistics to rank the cases.
  Then applies the *1*-NN or *k*-NN search (through preselected cases). Search is faster than in the above mentioned cases but not necessarily more accurate.

➜ Good retrieval algorithm: the best compromise between accuracy and efficiency.

Case Based Reasoning (CBR) – Adaptation

- Adaptation algorithm adapts the solutions associated with the retrieved cases to the currently presented problem / situation.

- *Structural* Adaptation

  Applies a set of adaptation rules directly to the retrieved solutions. Adaptation rules can include, e.g., modifying certain attributes through interpolating between relevant attributes of the retrieved cases.

- *Derivational* Adaptation

  Uses algorithms / rules that have been used to generate the original solution. Can be used only for problem domains that are completely transparent. ↔ Not used very often.

- *Manual (User-driven)* Adaptation

  If no exact match is found, asks the user for a feedback. Adapts the solutions accordingly. Faulty adaptations cannot be encountered. Used very often.

Lazy Learning – Advantages

- *Incremental (online) learning*: The problem-solving ability is increased with each newly presented case.
- *Suitability for complex and incomplete problem domains*: A complex target function can be described as a collection of less complex local approximations and unknown classes can be learned.
- *Suitability for simultaneous application to multiple problems*: Examples are simply stored and can be used for multiple problem-solving purposes.
- *Ease of maintenance*: A lazy learner adapts automatically to changes in the problem domain.

Lazy Learning – Disadvantages

- *Handling very large problem domains*: This implies high memory / storage requirements and time-consuming search for similar examples.
- *Handling highly dynamic problem domains*: In CBR, this involves continuous reorganisation of the case base, which may introduce errors in the case base. Overall, the set previously encountered examples may become outdated if a sudden large shift in the problem domain occurs.
- *Handling overly noisy data*: Such data may result in storing same problems numerous times because of the differences in cases due to noise. In turn, this implies high memory / storage requirements and time-consuming search for similar examples.
- *Achieving fully automatic operation*: Only for complete problem domains a fully automatic operation of a lazy learner can be expected. Otherwise, user feedback is needed for situations for which the learner has no solution.

Instance Based Learning – Exam Questions

- Tom Mitchell's book –chapter 8
- Relevant exercises from chapter 8:   8.1, 8.2, 8.3
- Case-Based Reasoning Syllabus

- To prepare assignment 3 of the CBC read:

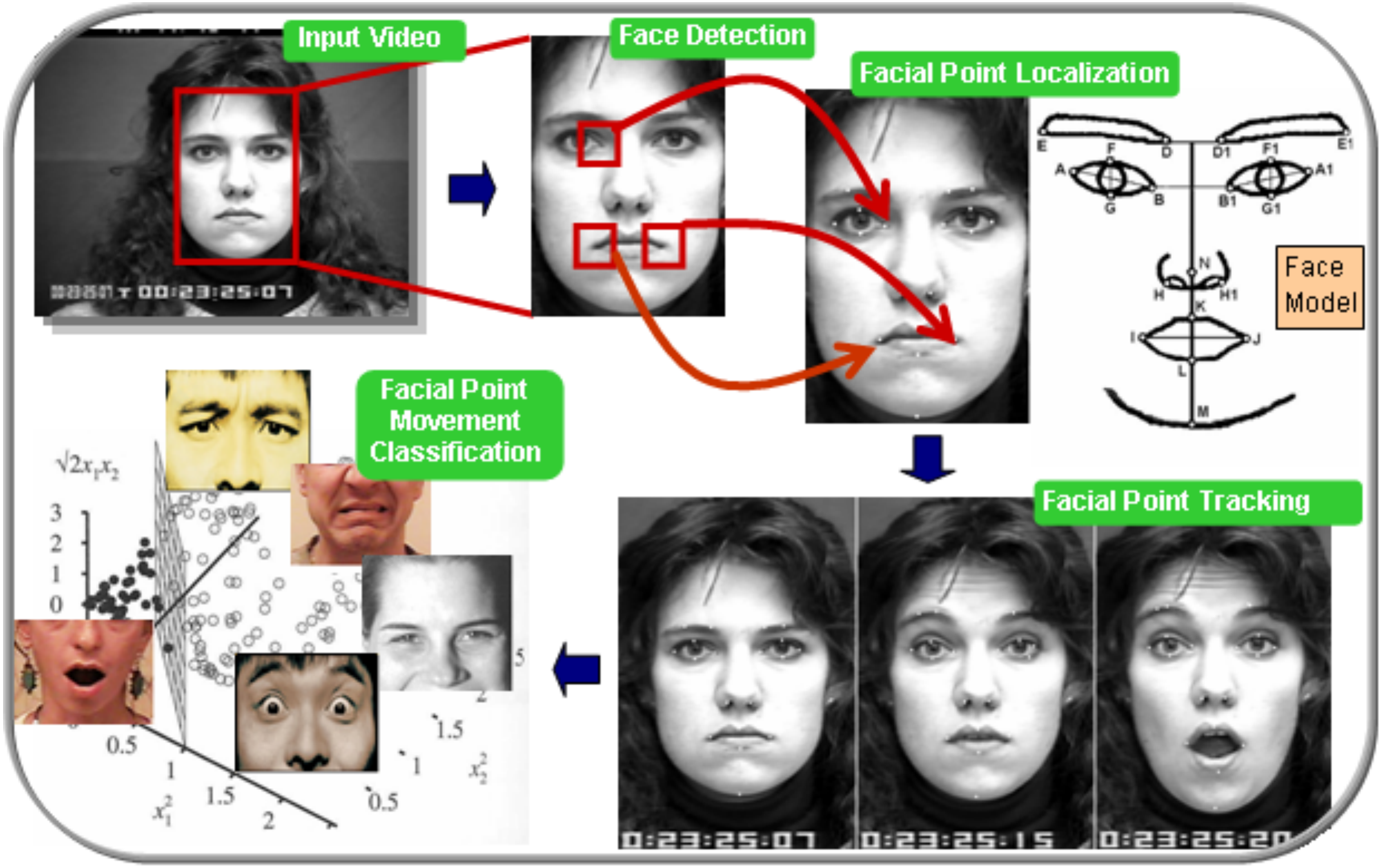
  Pantic & Rothkrantz (2004):
  "CBR for user-profiled recognition of emotions from face images"

Instance Based Learning – Lecture Overview

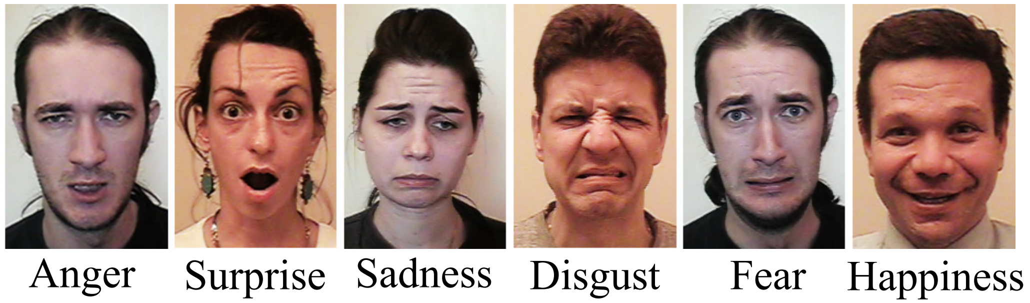
- Lazy learning
- K-Nearest Neighbour learning
- Locally weighted regression
- Case-based reasoning (CBR)
- Advantages and disadvantages of lazy learning

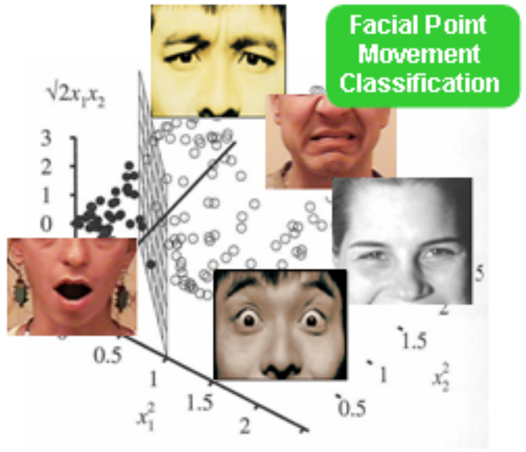
➢ (Example: CBR-based system for facial expression interpretation)

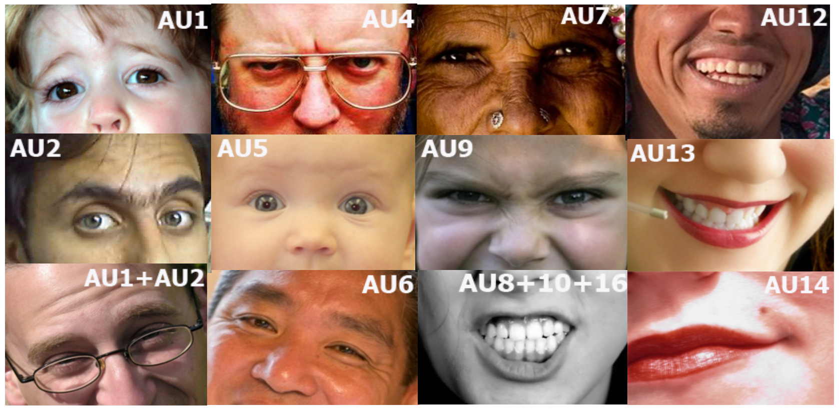
Automatic Facial Expression Analysis



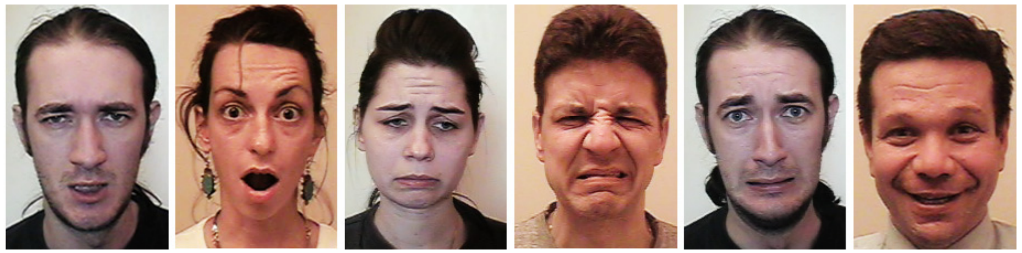
Automatic Facial Expression Analysis

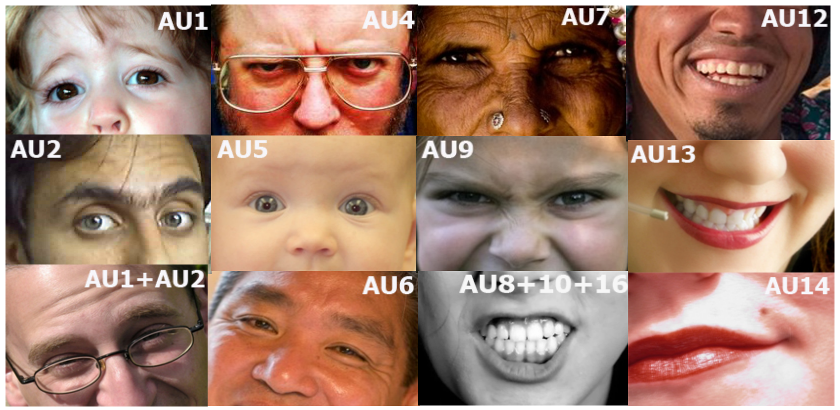






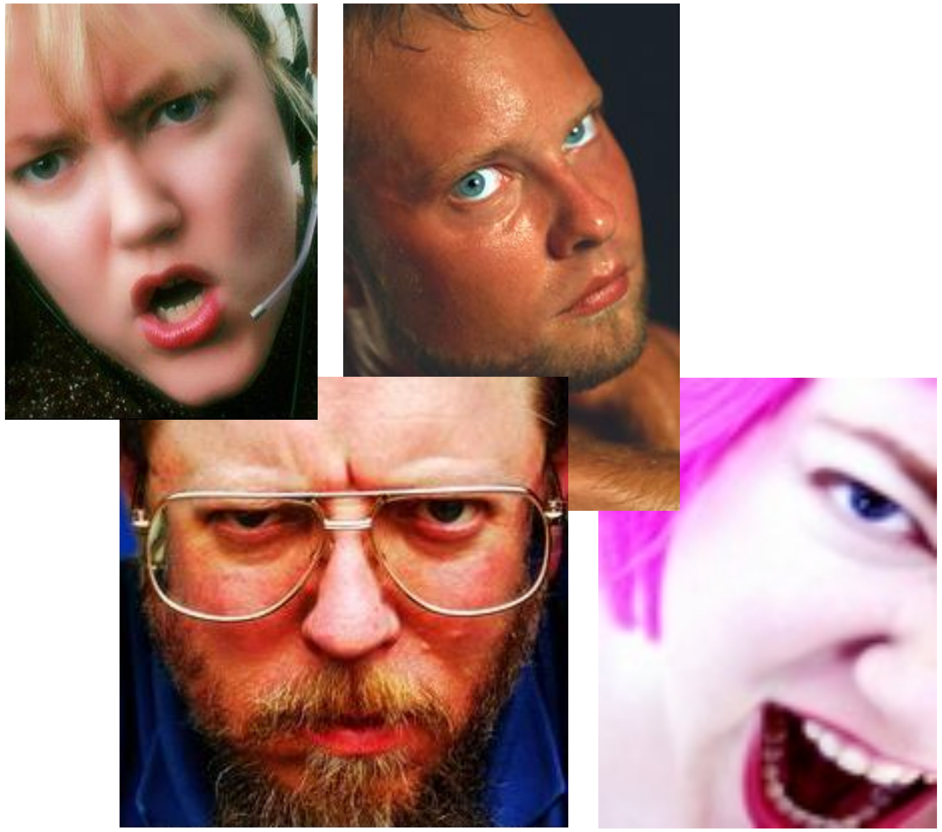
Automatic Facial Expression Analysis



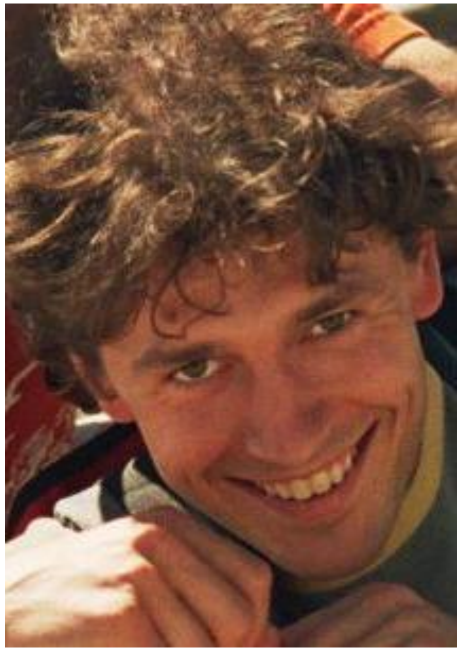


Emotion	AUs	Emotion	AUs
Happy	{12}	Fear	{1,2,4}
	{6,12}		{1,2,4,5,20, 25‖26‖27}
Sadness	{1,4}		
	{1,4,11‖15}		{1,2,4,5,25‖26‖27}
	{1,4,15,17}		{1,2,4,5}
	{6,15}		{1,2,5,25‖26‖27}
	{11,17}		{5,20,25‖26‖27}
	{1}		{5,20}
Surprise	{1,2,5,26‖27}		{20}
	{1,2,5}	Anger	{4,5,7,10,22,23,25‖26}
	{1,2,26‖27}		{4,5,7,10,23,25‖26}
	{5,26‖27}		{4,5,7,17,23‖24}
Disgust	{9‖10,17}		{4,5,7,23‖24}
	{9‖10,16,25‖26}		{4,5‖7}
	{9‖10}		{17,24}

User-profiled Facial Expression Interpretation

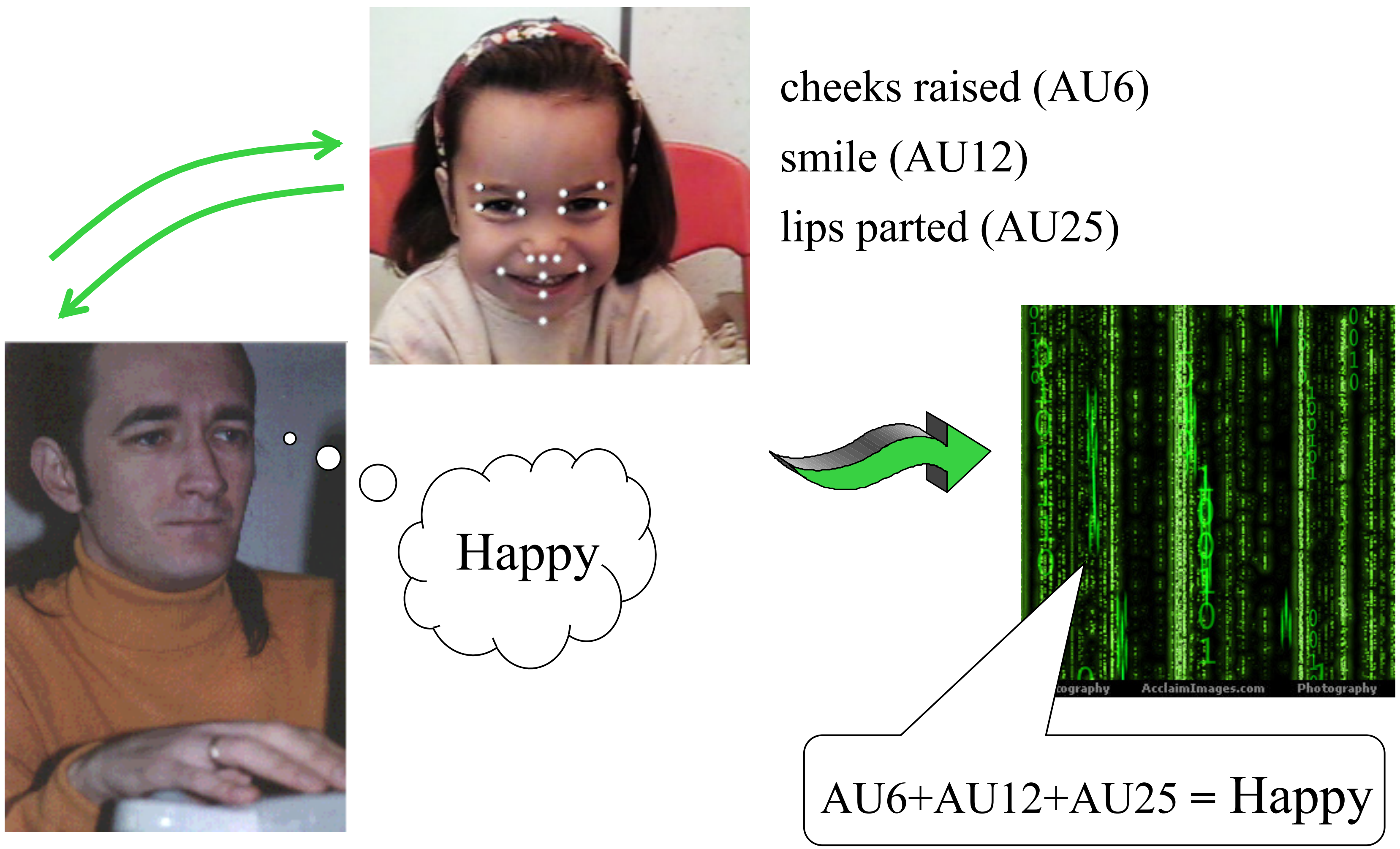


Could you please display an angry expression?

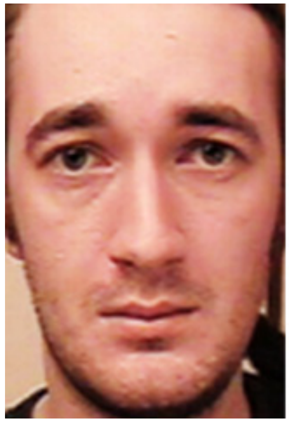


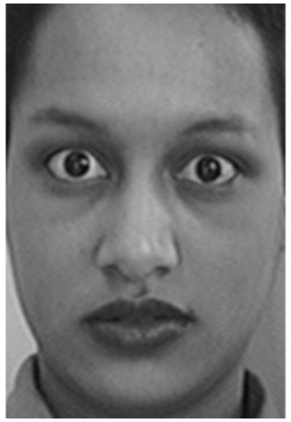
How would you interpret this? Happy? Angry? Teasing?

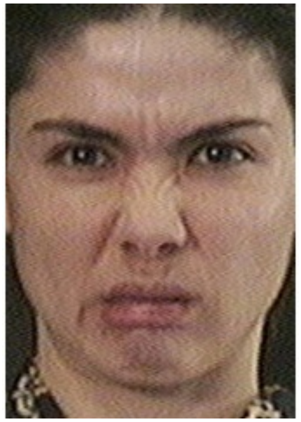
User-profiled Facial Expression Interpretation

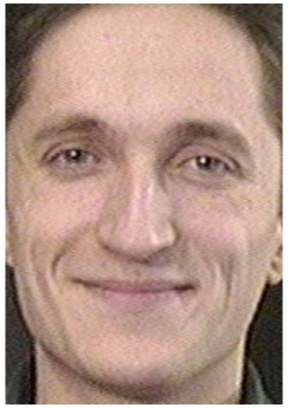


Case Base Initialisation









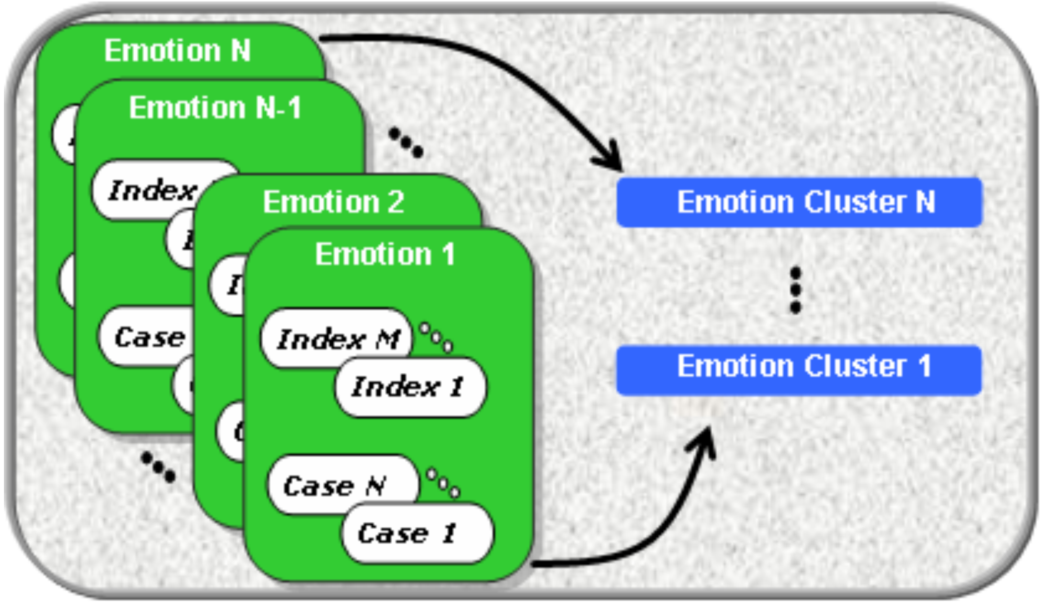
AUs	*Case explanation*
1	raised inner eyebrow
2	raised outer eyebrow
1+2	from "surprise"
4	furrowed eyebrows
5	raised upper eyelid
7	raised lower eyelid
1+4+5+7	from "fear"
…	
1+4	from "sadness"
…	
9	wrinkled nose
9+17	from "disgust"
…	
6+12	from "happiness"
…	

Case Base Initialisation

AUs	*User's interpretation*
1	disappointed
2	angry
1+2	surprised
4	angry
5	please don't
7	angry
1+4+5+7	please don't
…	
1+4	disappointed
…	
9	slimy (yak!)
9+17	slimy (yak!)
…	
6+13	happy
…	

Case Base Organisation

AUs	*User's interpretation*
1	disappointed
2	angry
1+2	surprised
4	angry
5	please don't
7	angry
1+4+5+7	please don't
…	
1+4	disappointed
…	
9	slimy (yak!)
9+17	slimy (yak!)
…	
6+13	happy
…	



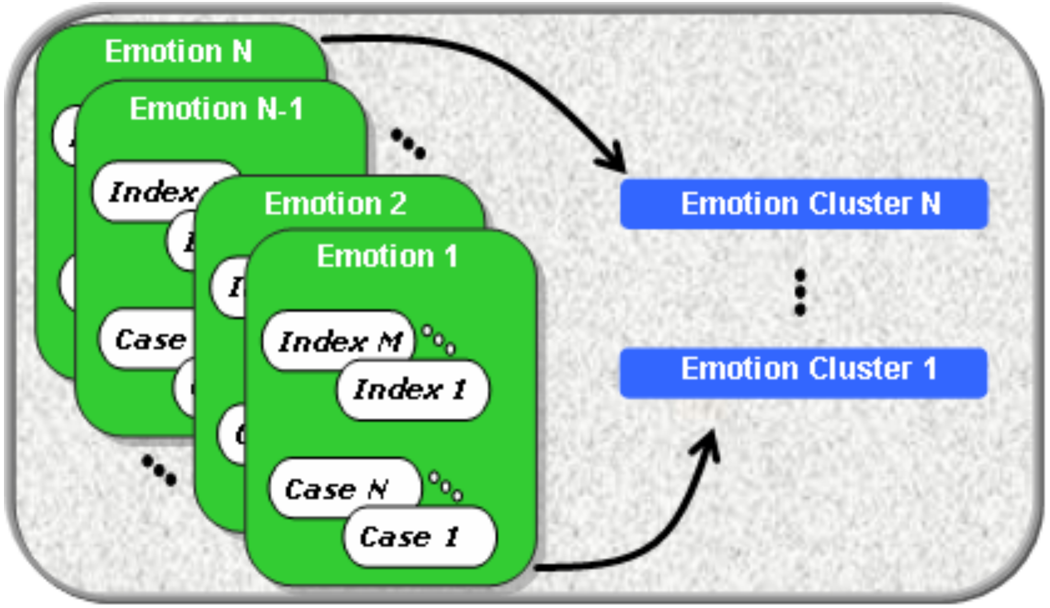
Clusters:

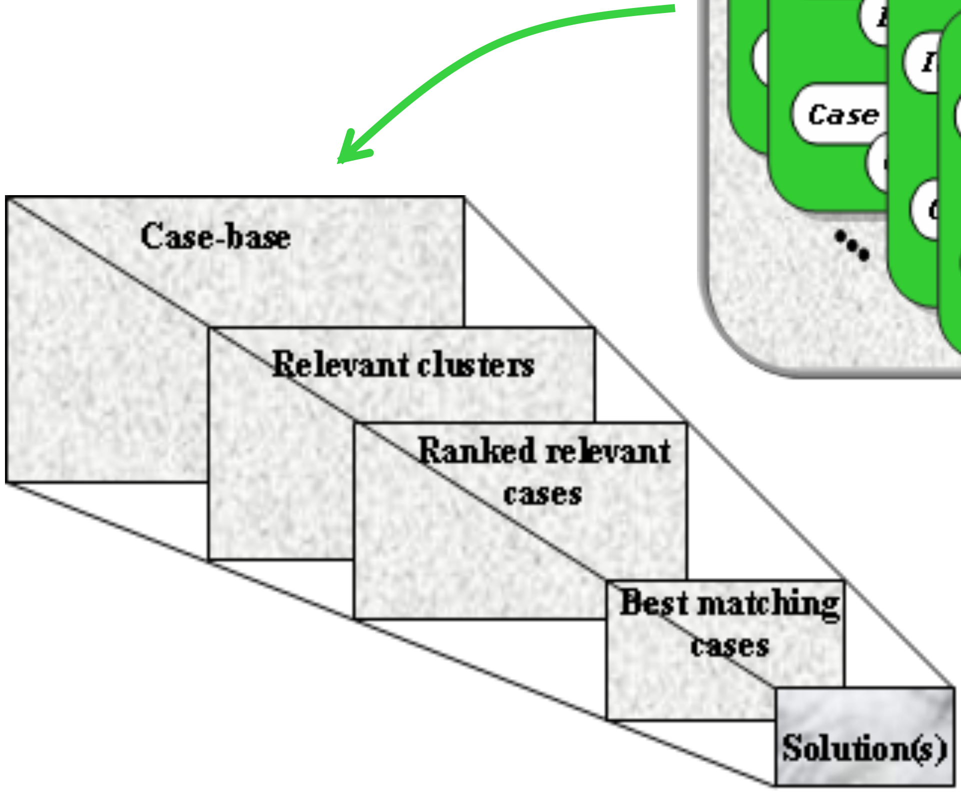
label ‹angry›

cases ‹(2,4,0); (4,0); (7,0);… ;(24,0); (24,17,0)›

index ‹4, 7, 24,…›

Retrieval





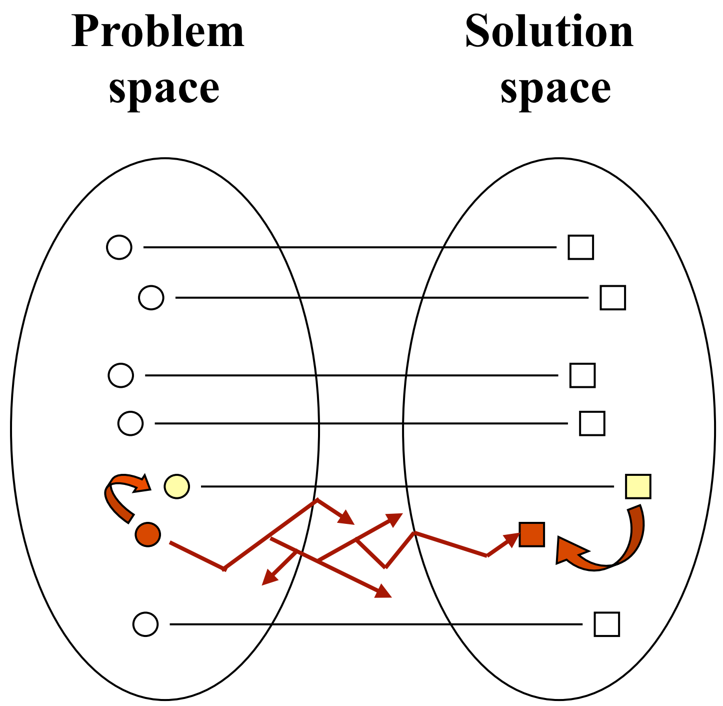
Clusters:

label ‹angry›

cases ‹(2,4,0); (4,0); (7,0);…; (24,0); (24,17,0)›

index ‹4, 7, 24,…›

Adaptation



User-profiled AU interpretation

1. Search the Case Base for similar cases, retrieve them, and interpret the input set of AUs using the interpretation labels suggested by the retrieved cases.
2. If the user is satisfied with the output, store the new case in the Case Base. Otherwise, adapt the Case Base (i.e., store the new interpretation that the user associates with the input facial expression).

Pantic and Rothkrantz, *Proc. IEEE ICME'04*

Course 395: Machine Learning – Lectures

- Lecture 1-2: Concept Learning (*M. Pantic*)
- Lecture 3-4: Decision Trees & CBC Intro (*M. Pantic*)
- Lecture 5-6: Artificial Neural Networks (*S. Zafeiriou*)
- Lecture 7-8: Instance Based Learning (*M. Pantic*)
- ➢ Lecture 9-10: Genetic Algorithms (*M. Pantic*)
- Lecture 11-12: Evaluating Hypotheses (*THs*)
- Lecture 13-14: Guest Lectures on ML Applications
- Lecture 15-16: Inductive Logic Programming (*S. Muggleton*)
- Lecture 17-18: Inductive Logic Programming (*S. Muggleton*)