Data Mining

Lecture 9
Genetic and fuzzy mining

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

Motivation

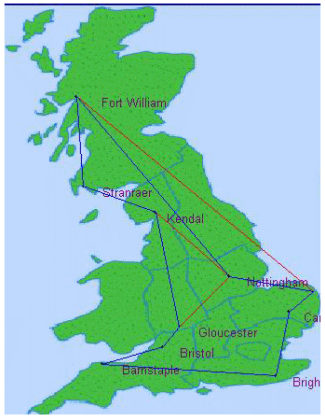
- Data mining in the real world:
  - Often related to solving ***complex problems***
  - Time for analysis and algorithms development decreases
  - More ***universal*** algorithms with automatic adaptation needed
  - “good” solutions within acceptable time are often satisfying
- Most powerful problem solvers in nature:
  - The (human) brain
    ... that created “the wheel, New York, wars and so on” [Adams 1978]
  - The evolutionary mechanism
    ... that created the human brain [Darwin 1895]
- Nature inspired approach to Data Mining:
  Approximate and ***fuzzy*** methods

Outline for today

- Approximate solutions with genetic algorithms
  - Ideas and mechanisms of evolutionary computing
  - The genetic algorithm and its basic operators
  - Classification and clustering with genetic algorithms
- Fuzzy representations with fuzzy logic
  - The fuzzy logic concept for the complex real world
  - Fuzzy sets, operations, rules, and inference
  - Fuzzy rule extraction and fuzzy clustering

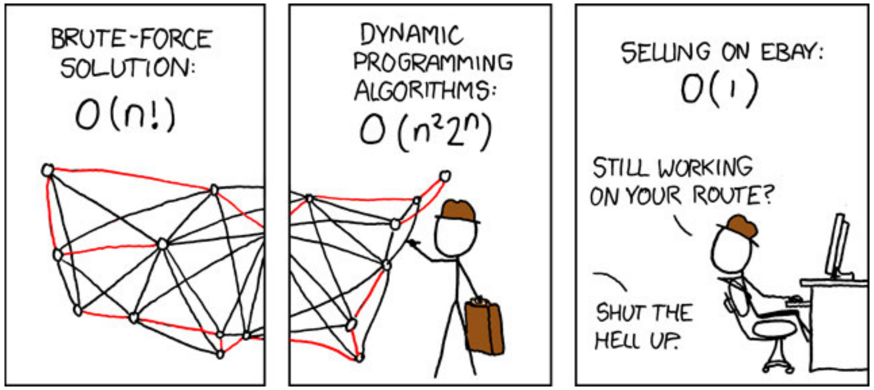
Complex Problems: The Travelling Salesman

- Find a tour that starts and ends at the same city, visits every city precisely once, and has the minimum possible distance



- Example for an NP-complete problem
  - There are n! different possible solutions (where N is the number of cities)
  - Virtually ***impossible to solve*** for N>10

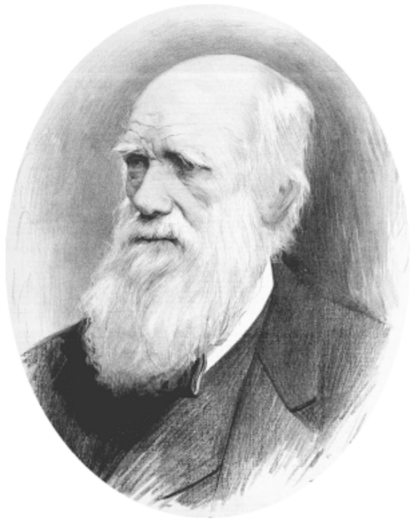
- A "good" ***approximate*** solution is ***acceptable***



Genetic Algorithms

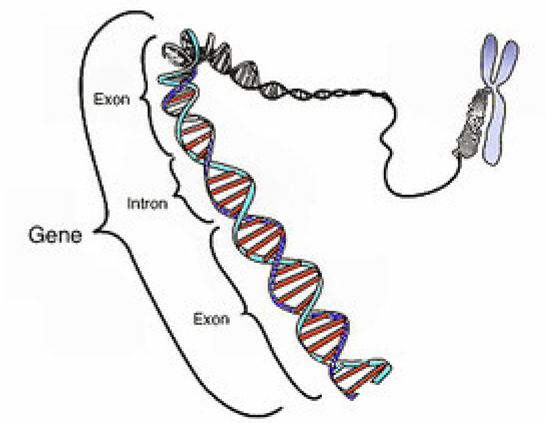
- Trial-and-error problem solving method
- Standard approach in Evolutionary Computing
- Population of individuals with reproduction and mutation
- “Survival of the fittest” (Darwin)
- “Diversity drives change” (Darwin)
- Inspired by natural evolution:

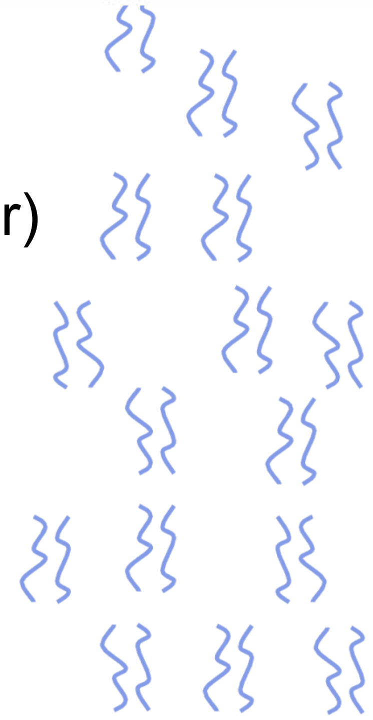
Evolution		**Problem Solving**
Environment	↔	Problem
Individual	↔	Candidate solution
Fitness	↔	Quality



Biological Background

- Genotype determines Phenotype
- Genes: ***complex mapping***
  - One gene may affect many traits (features)
  - Many genes may affect one trait
  - Small changes in the genotype lead to large changes in the organism (e.g. Height, hair colour)
  - Encoded in strands of DNA
- Human DNA is organised into chromosomes
  - Characterised by Alleles (allele is one of two or more forms of a gene or a gene locus)
  - Together define the physical attributes of the individual





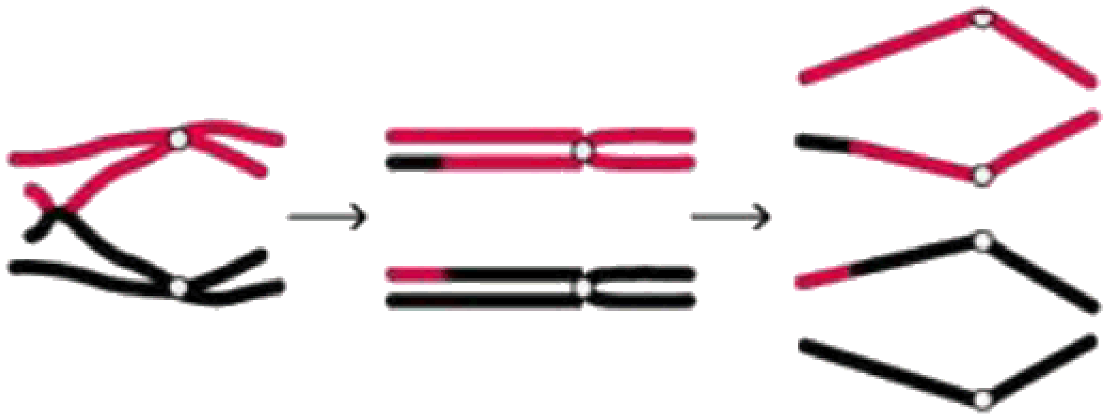
Phenotype & Genotype

- ***Phenotype***: Manifestation of the organism (appearance, behavior, etc.).
  - Selection operates on the phenotype;
  - It is affected by environment, development, and learning
- ***Genotype***: The genetic material of that organism.
  - It is transmitted during reproduction;
  - It is affected by mutations;
  - Selection does not operate directly on it
- ***Genetics***: Structure and operation of genes
- ***Functional genomics***: Role of genes in the organism
- To what extent are we determined by genotype and phenotype?

Jean-Felix & Auguste Piccard

Cell Replication

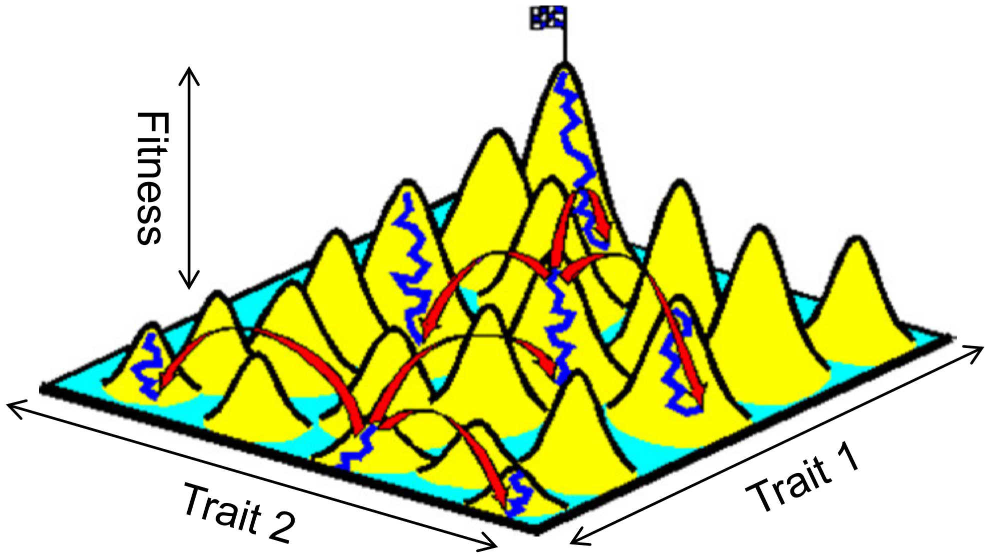
- Gametes: Sperm or egg cell
  - Contain only one single chromosome complement of chromosomes
  - Formed by a special form of cell splitting: ***Meiosis***
  - During meiosis, pairs of chromosomes undergo ***crossing-over***



- Occasionally some of the genetic material changes very slightly (replication error): ***Mutations***

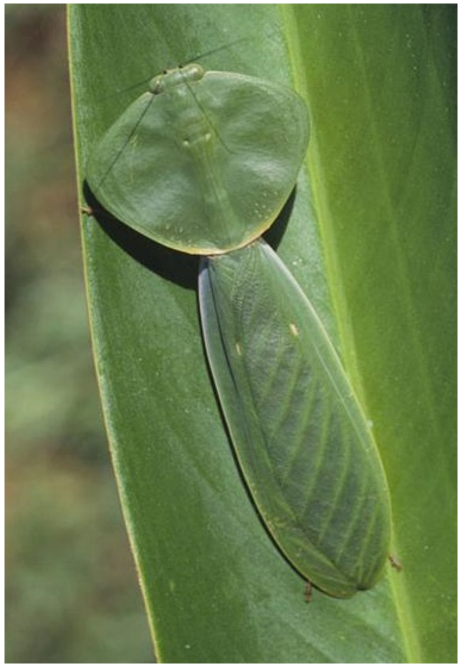
Fitness Landscapes

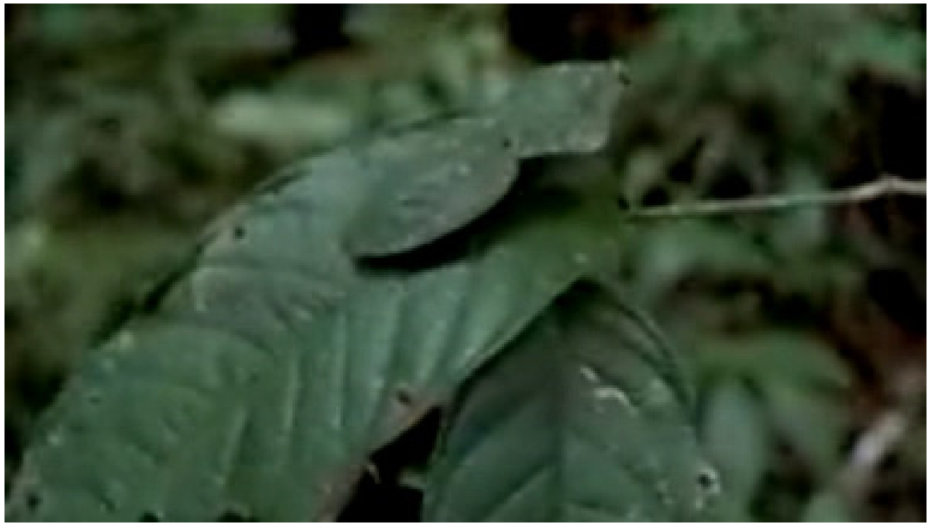
- Useful imagination aid
- Animals ***fitness*** is partly due to ***competition*** with other animals and environment
  - Changes over time
  - Evolution favors animals that evolve to peaks of the fitness landscape



- ***Exploitation*** and ***Exploration***
  - Selection forces adaptation (exploitation)
  - Crossover and mutation create novel solutions (exploration)

Example: Evolution of Camouflage



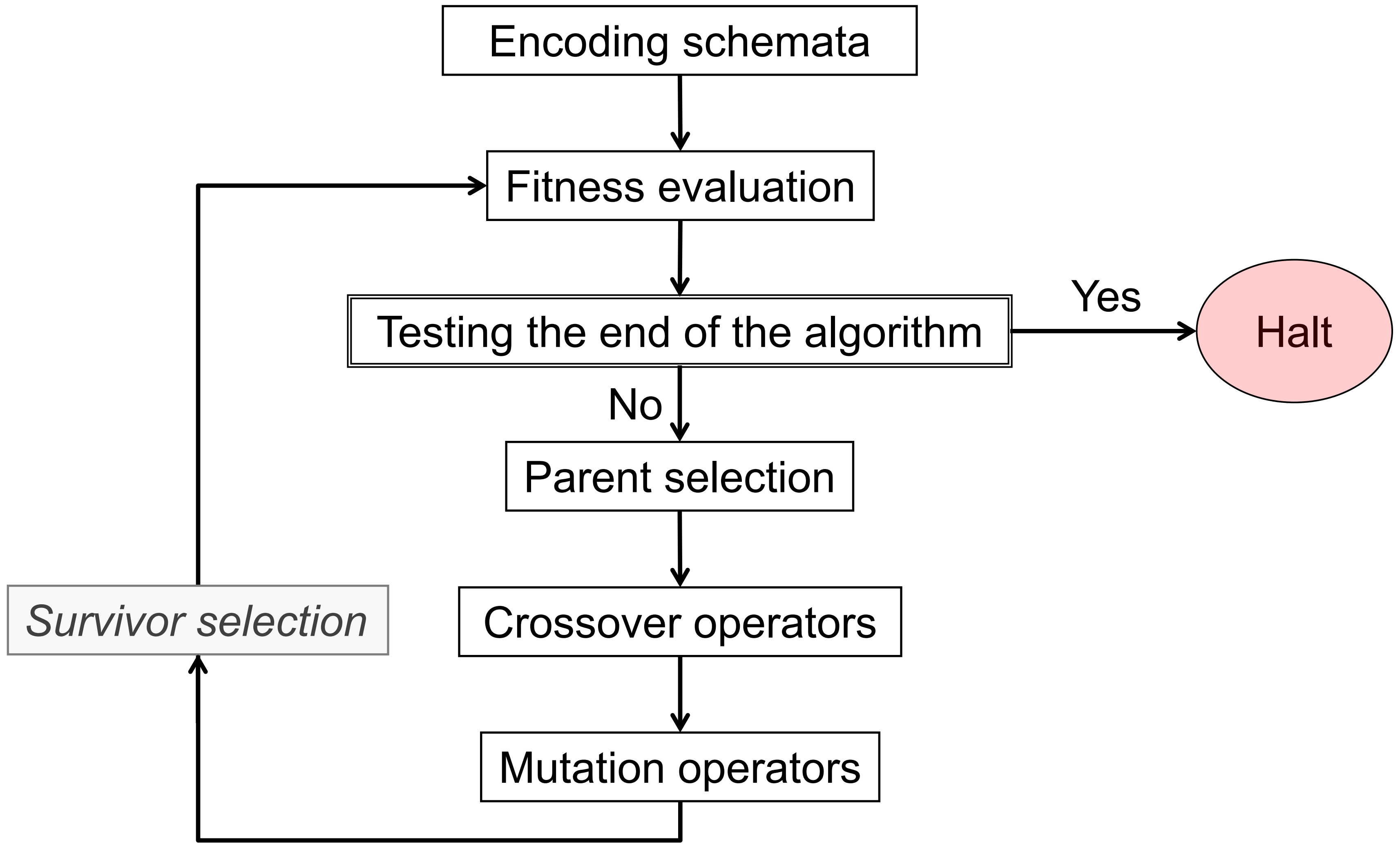


Chris Schneider, Boston University

Basic Concepts in Genetic Algorithms

Concept in Natural Evolution	Concept in Genetic Algorithms
Chromosome	String
Gene	Elements or features in the string
Locus	Position in the string
Allele	Position value from an alphabet
Genotype	String structure
Phenotype	Set of characteristics (features)

Major Phases of a Genetic Algorithm



Encoding Schemata: Representation

- Cover all possible solutions
- Encoding should only allow valid solutions (not always possible)
- Choose ***appropriate representation***:
    - Bit-string can decode integers or real numbers
    - Problems: e.g. mutation changes values significantly (work-around: *Gray coding:* two successive values differ by 1 bit
    - Better direct representation of numbers
    - Bit-string often fine
- Two meanings of representation:
    - Mapping between phenotype & genotype space: de-/encoding
    - Data structure used in genotype space
- Initialization: Mostly at random

Representation: Strings and Things

- Representation is always a bias into what can be learnt!
- Choose an alphabet
  - Possible values of each element of string
  - Often binary
- Split up the problem into discrete parts

- **Example**: bill paying
  - List of 100 bills to pay
  - Use string of 100 elements
  - Each element is whether to pay one bill
  - 10110 means pay bills 1, 3, and 4

Fitness Function

- Represents requirements to adapt to
- Basis for selection: decide how good the string is
- Assigns quality measure to genotypes
- Synonyms: ***evaluation*** or ***objective function***
- Always problem-specific

Example:

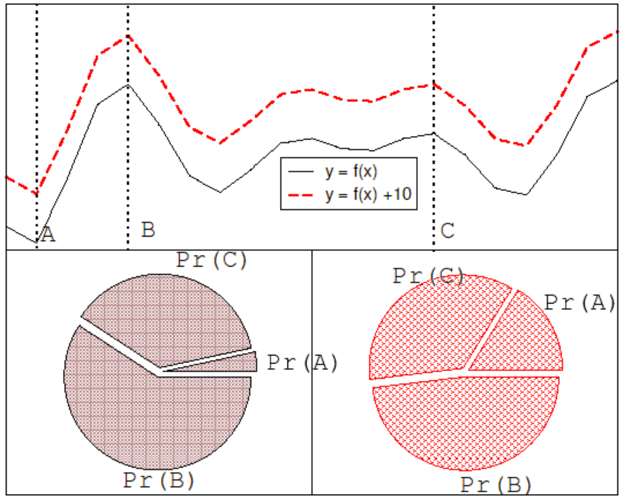
- ***Context***: Minimize x^2
- ***Phenotype***: $x \in \mathbb{N}$
- ***Genotype***: z: binary representation of x
- ***Fitness Function***: fitness $f(z)$ of genotype z is defined as 1 divided by square of its corresponding phenotype, e.g.: z = 0010 $\rightarrow$ phenotype: $x = 2$ $\rightarrow$ $f(z) = 1/x^2 = 0.25$

Parent Selection

- Select individuals to create new population (offspring)
- Mainly based on fitness
- Often ***probabilistic***:
  - Individuals with high fitness more likely to become parents
  - "Weak" individuals might also become parents (with low probability) to avoid local optima
  - Sum over all probabilities is 1.0
- Parent selection supports process of evolving better solutions over time
  - Serves the ***exploitation*** of solutions

Parent Selection – Mechanisms

- ***Fitness Proportional Selection*** (FPS):
  - Fitness of individual proportional to fitness of population
  - Probability to select i from population of size n: $p_{fps,i} = f_i \Big/ \sum_{k=1}^{n} f_k$



- ***Ranking Selection***
  - Rank individuals by fitness and select i based on rank
  - **E.g.**: Linear Rank (LRS) $p_{lrs,i} = (2-s)\big/ p_{fps} + 2i(s-1)\big/ p_{fps}(p_{fps}-1)$

	Fitness	Rank	p_{fps}	p_{lrs} (s=2)	p_{lrs} (s=1.5)
A	1	1	0.1	0	0.167
B	5	3	0.5	0.67	0.5
C	4	2	0.4	0.33	0.33
Sum	10		1.0	1.0	1.0

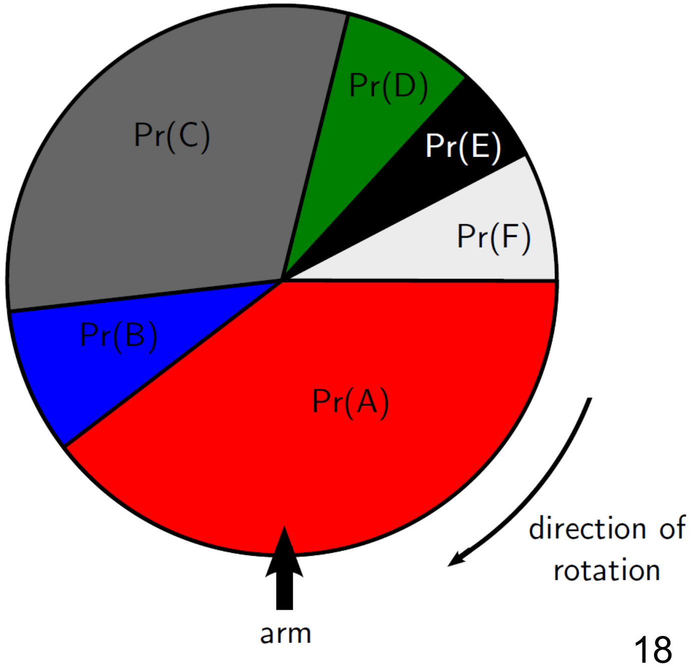
Parent Selection – Sampling: Roulette Wheel

- FPS and Ranking Selection define probability distributions *q* for selecting individuals

$$q_{fps,i} = \sum_{j=1}^{i} p_{fps,j}$$

- How to sample these distributions?
  - *Roulette Wheel (RW)!*
  - Spin the wheel **n**-times and select the winning chromosome each

- Other sampling strategies:
  - Tournament selection
  - Truncation selection
  - Elitism



Variation: Crossover

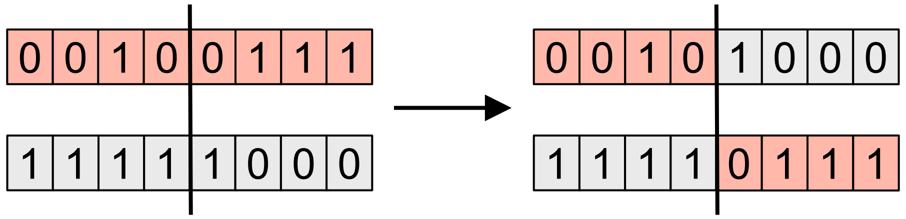
- Variation operator: create new individual(s) from parents
  - Synonym for crossover: ***Recombination*** operator
  - Distinguishes GAs from other optimization techniques
  - Merge information from parents into offspring
  - Aims for diversity,
    but sometimes a destructive jump in fitness landscape
- Crossover applied with probability p_c, e.g. $p_c \in [0.5, 1.0]$; otherwise parents are copied
- Implementation depends on representation form

n-Point Crossover

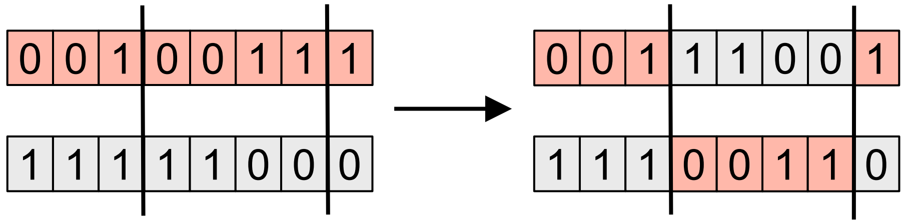
- Split parents at *n* points and recombine segments
- ***Positional*** bias:
  - *n*-Point Crossover tends to keep together genes located close to each other
  - One-Point can never keep together genes from opposite ends
  - Knowledge on problem structure often not available

Example:

One-Point Crossover



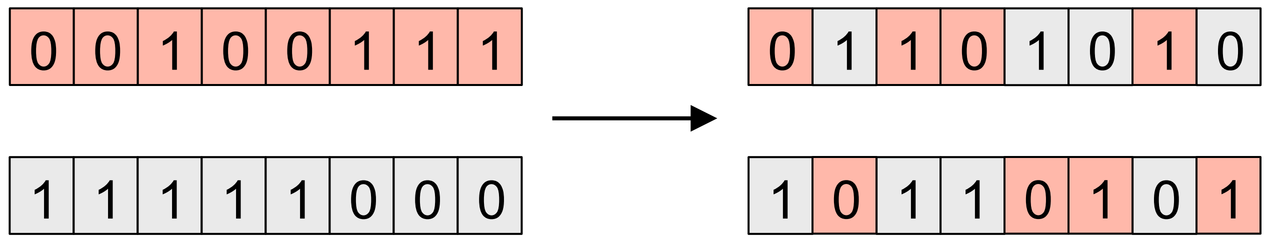
Two-Point Crossover



Uniform Crossover

- Swap genes based on vector of random values
  - Process generates first child
  - Second child: inverse process
- ***Distributional*** bias:
  - Genes are distributed among children instead of transferring larger sets of co-adapted genes to one child

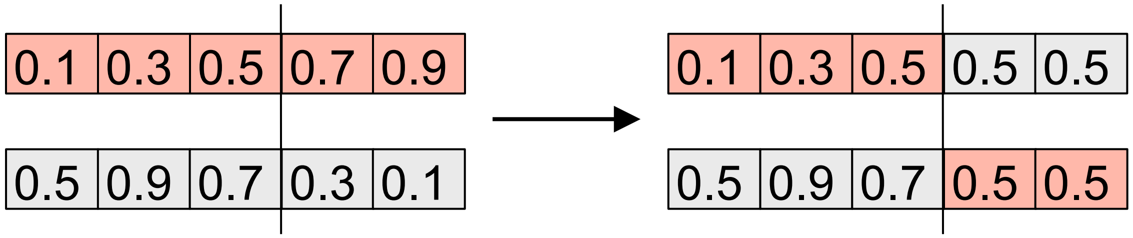
Example: p=0.5, x = (0.3, 0.7, 0.4, 0.2, 0.6, 0.9, 0.1, 0.8)



Beyond Crossover – Further Operators

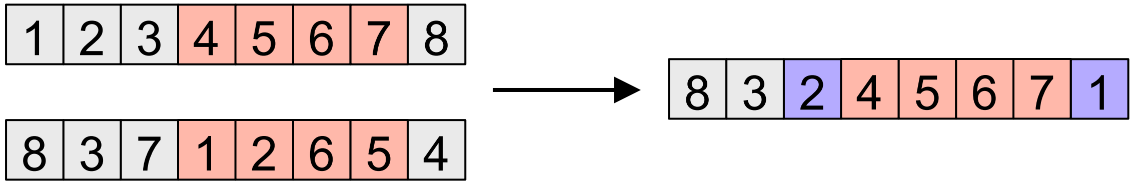
- Arithmetic Recombination:
  - Powerful for floating-point representations

    Example: Simple arithmetic recombination



- Permutation:
  - Powerful if exchange of substrings is constrainted

    Example: Partially Mapped Crossover (PMX)

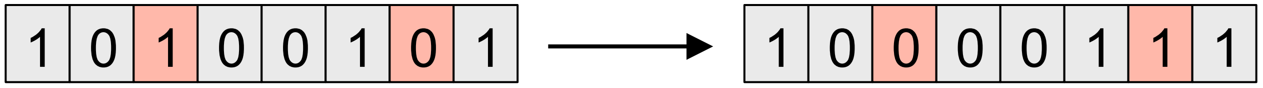


Variation: Mutation

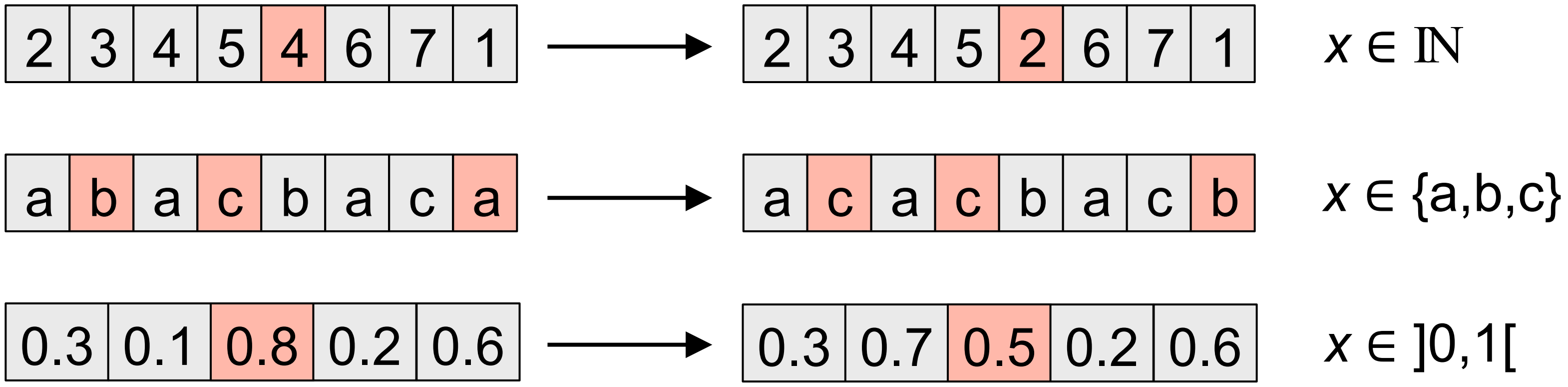
- Variation operator: spontaneously create new individual(s) from old ones
- Slightly mutates one individual
- Always ***stochastic***: random and unbiased changes
  - Mutation can prevent a single bit from converging
  - Serves the ***exploration*** of solutions
- Implementation depends on representation form

Variation: Mutation – Change of Allele Values

- Bitwise mutation
  - For every position: flip bit with probability p_m



- Random resetting / Uniform mutation
  - For every position: change value to random value from the corresponding domain with probability p_m

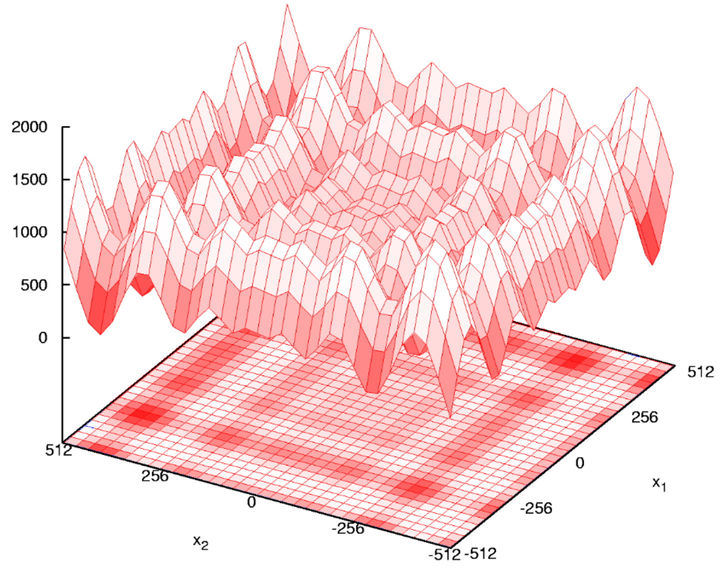


Survivor Selection

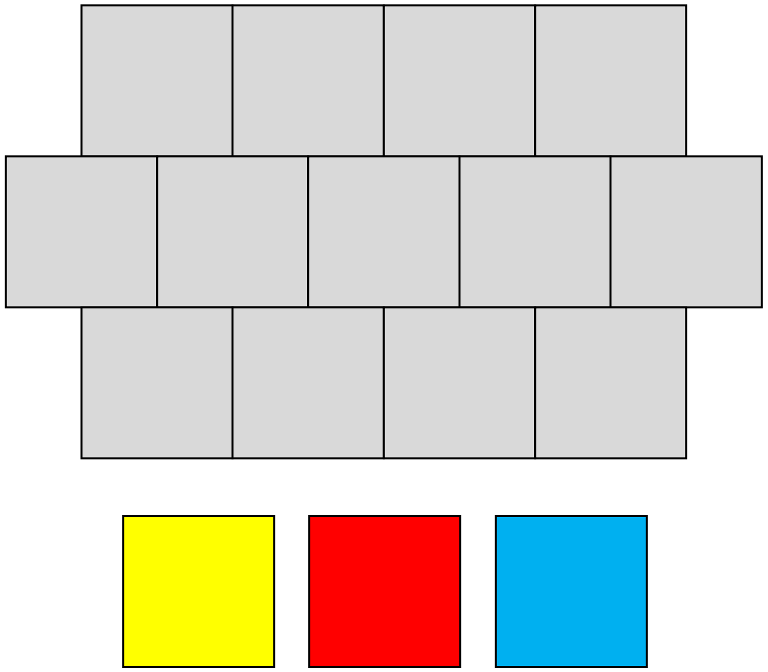
- Similarly to parent selection, survivor selection selects individuals based on quality
- Role: reduce n parents and o offspring to n individuals that constitute the next generation
  - Fitter individuals more likely to survive
  - Weak individuals may also survive
- Selection often ***deterministic*** based on age and/or fitness
  - Age-based: Choose n best from *offspring* only
  - Fitness-based: Choose n best from *parents* and *offspring*
  - Elitism: Keep the $e < n$ best, replace the rest by offspring
- Synonyms: environmental selection, replacement

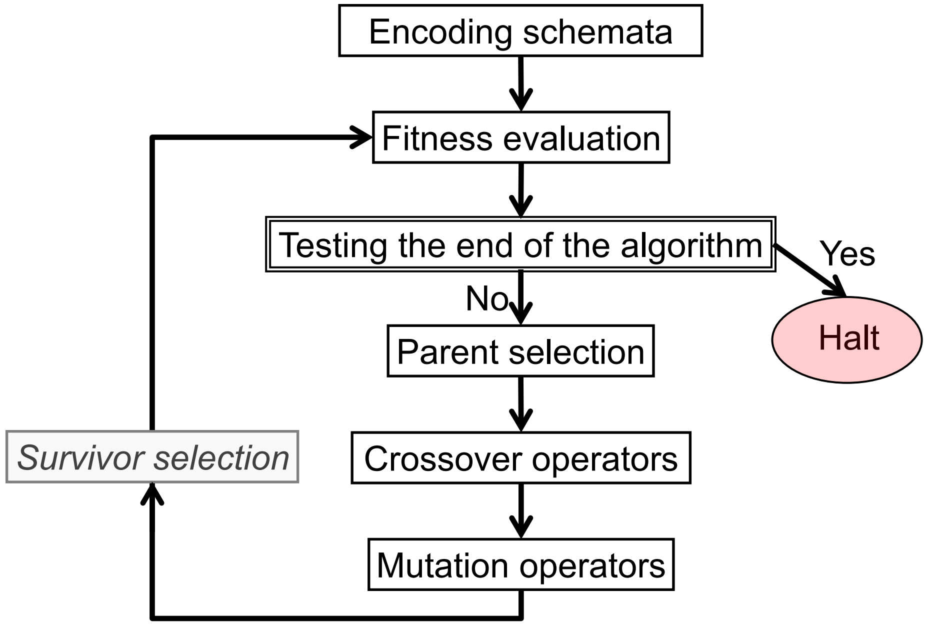
Testing the End of the Algorithm: Termination Condition

- Optimum value known: stop if reached, or if only a small $\varepsilon > 0$ away
- Problem: GAs are ***stochastic*** and may never fulfill that condition
- Other possible criteria include:
  - CPU time elapsed
  - Maximum number of iterations or fitness calculations reached
  - No fitness improvement within last *t* generations
  - Diversity of population too small



Example: Coloring a Map (3-Coloring Problem)

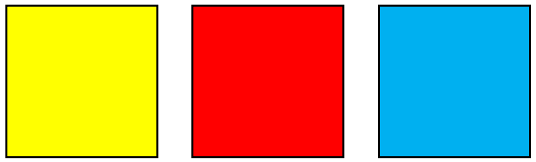


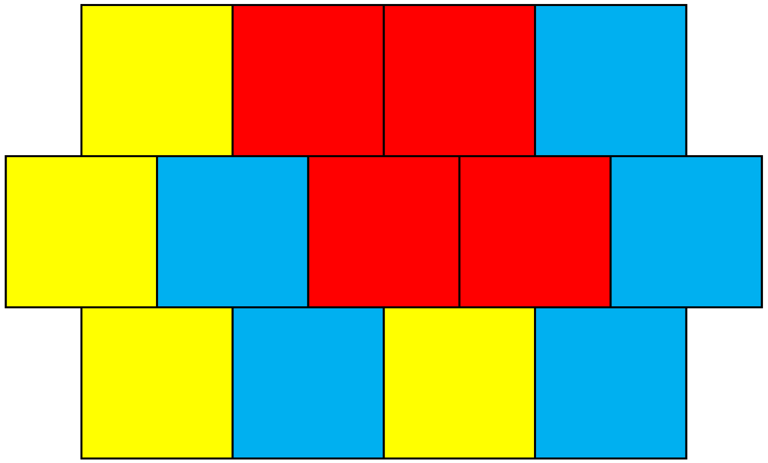


- Color the map with any of three colors
- No two bordering countries can have the same color

Coloring a Map with a Genetic Algorithm

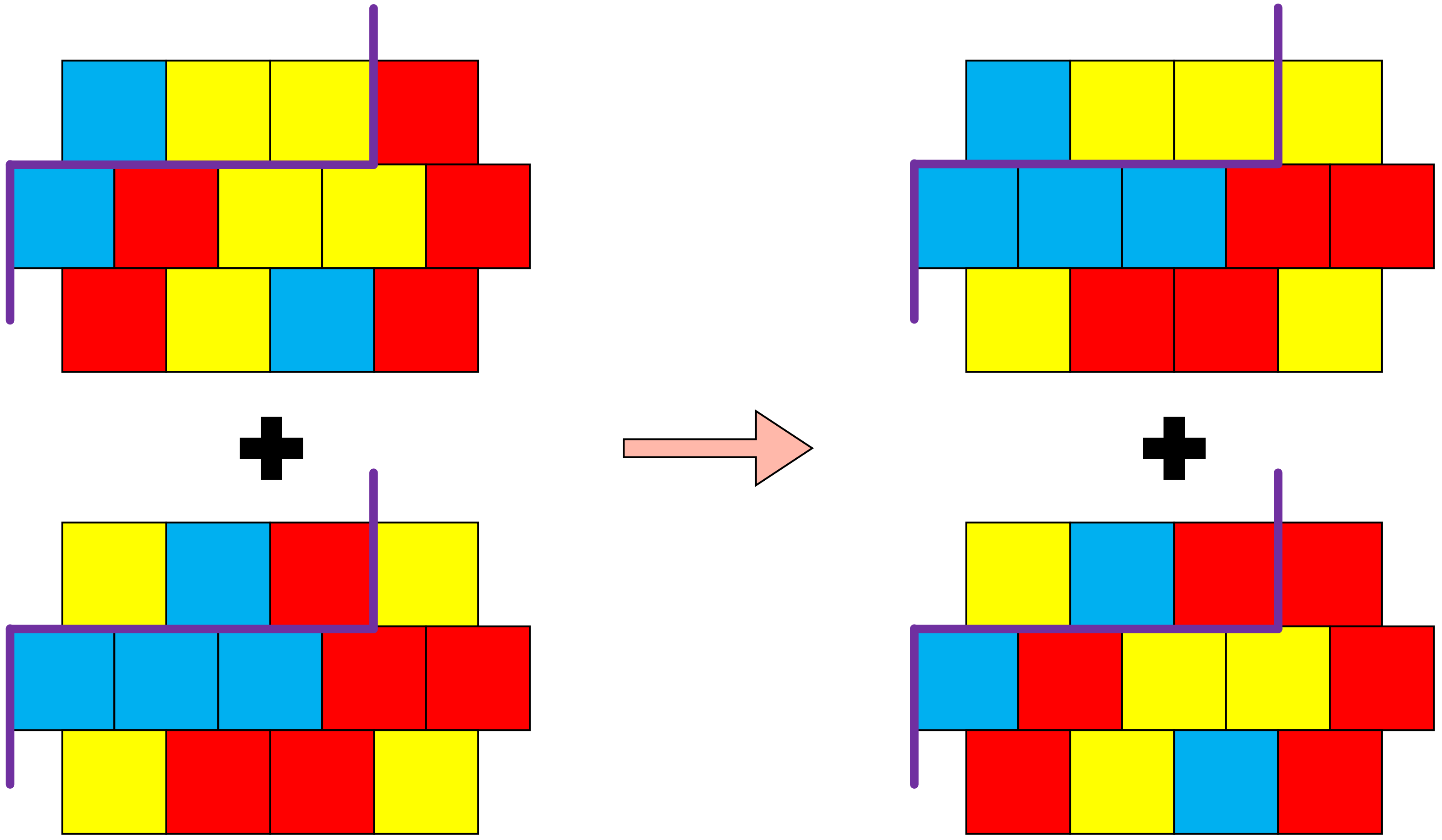
- String representation:
  - `[Y,R,R,B,Y,B,R,R,B,Y,B,Y,B]`
- Fitness function
  - Number of boundaries that are OK
  - Fitness increases for better solutions





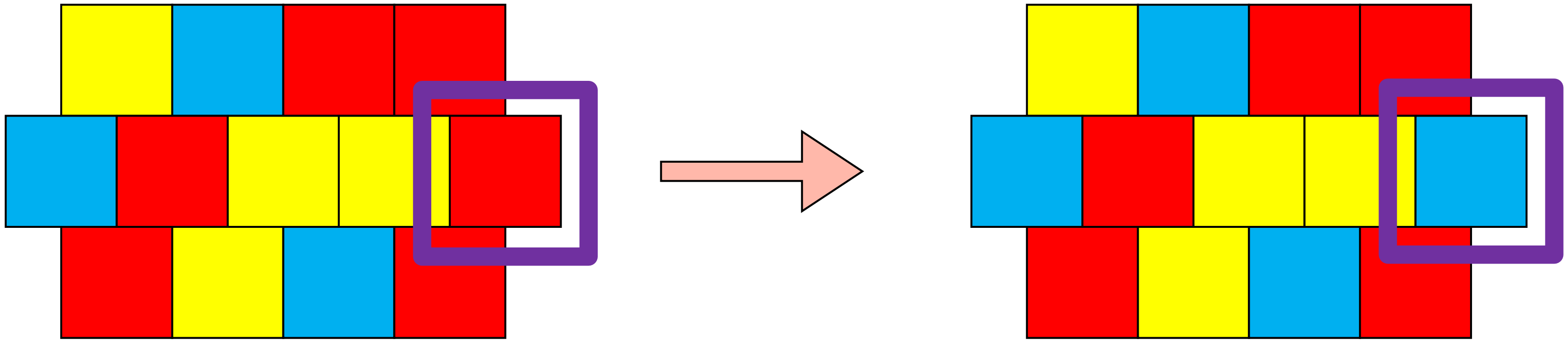
$f = 16$ – best fitness would be 26

Coloring a Map with a GA: Crossover



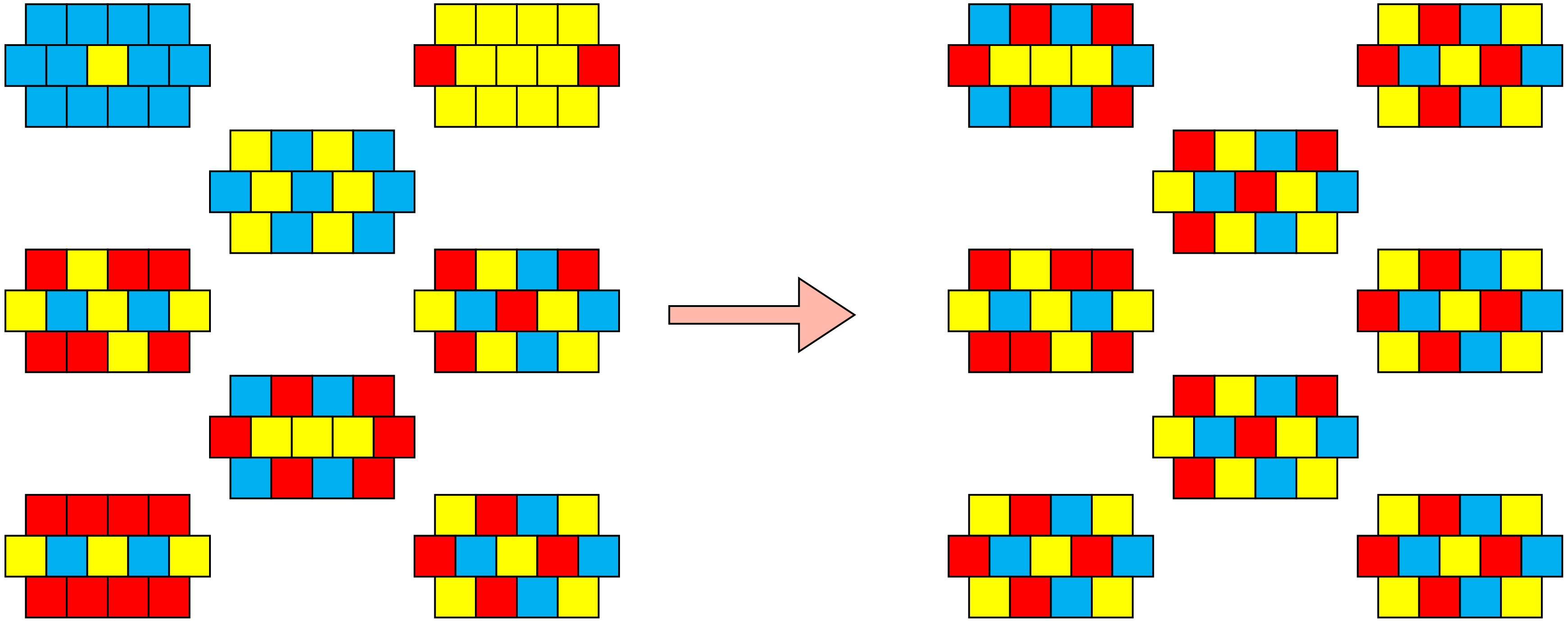
- Combine pairs of strings to form new strings

Coloring a Map with a GA: Mutation



- With small probability, change randomly selected position in the string (in the map) to another color value

Coloring a Map: Example Populations



- Apply these genetic operators, to create a new population
- Use bias-selection to pick the fitter strings more often

Schema Theory

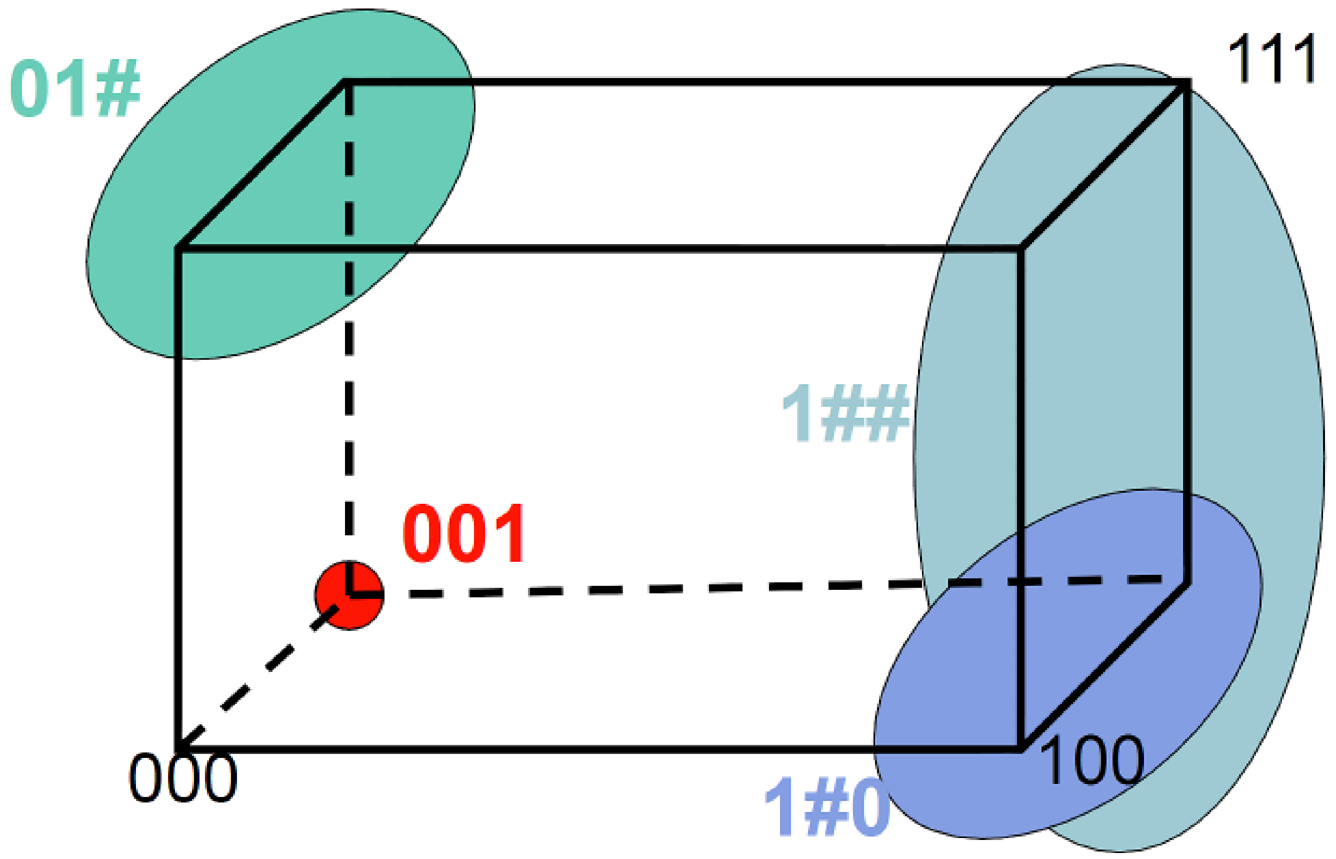
- Motivation:
  - Some ***search spaces*** (size of chromosome) are ***very big***
    - **Example**: Binary string with length 8: $2^8 = 256$ possible solutions
    - Knowledge about the search space could constrain the search
  - Sometimes ***knowledge*** is ***incomplete***
    - Mechanism is needed to fill undefined positions in the string
- ***Schema*** (pl. schemata)
  - A string in a ternary alphabet (`0`,`1`,`*` = "don't care") representing a hyper-plane within the solution space
    - **Examples**: $S1$: `*111100100`; $S2$: `001**1**0*`
  - All strings meeting this criterion are ***instances***
    - **Example**: 0111100100 and 1111100100 are instances of $S1$

Characteristics of Schemata

$S1$: **`*111100100`**; $S2$: **`01**1**0*`**

- ***Order*** (o): number of defined position
  - **Example**: $o(S1) = 10–1 = 9$; $o(S2) = 9–5 = 4$
- ***Length*** (l) of sub-string between outmost defined positions
  - **Example**: $l(S1) = 10–2 = 8$; $l(S2) = 8–1 = 7$
- ***Fitness*** of a schema S:
  - Average over all possible values in don’t care position
  - Is effectively sampled by the population, giving an estimated f

Example Schemata



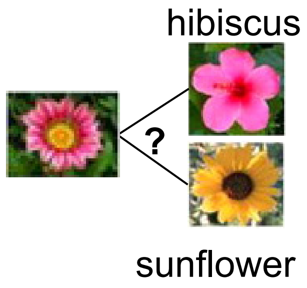
H	o(H)	d(H)
001	3	2
01#	2	1
1#0	2	2
1##	1	0

Classification and Clustering

- Recall: ***Classification***: Find the description of several predefined classes and classify a data item into one
- Recall: ***Clustering***: Identify a finite set of categories or clusters to describe the data
- Common algorithms for clustering and classification:
  - ***Specialized*** mechanisms for very good results
  - Sometimes ***inefficient*** computations due to
    - … necessity to iterate over whole dataset often (classification)
    - … complex metrics in high dimensionality (clustering)
- Genetic algorithms:
  - ***General*** method for broad number of problems
  - ***Easy to apply*** to get decent or “sufficient” results

Classification with Genetic Algorithms

- Idea:
  - Approximate a decent (not the best) function to ***map input*** (data features) ***to output*** (class)



- Approach:
  - GA can be used to find ***if-then rules***
  - **Examples**:
    - r_1: $([A_1=x] \wedge [A_5=s]) \vee ([A_1=y] \wedge [A_4=n]) \Rightarrow C_1$
    - r_2: $([A_3=y] \wedge [A_4=n]) \vee ([A_1=x]) \Rightarrow C_2$
  - More general form: $(p_1, p_2, p_3, p_4, p_5, p_6)$: c
    - for a data set with six inputs and one output
    - $i \in \{1..6\}$, $c \in \{C_1, C_2\}$

Attributes	Values
A_1	x,y,z
A_2	x,y,z
A_3	y,n
A_4	m,n,p
A_5	r,s,t,u
A_6	y,n

Classification with GAs: Schemata

r_1: $([A_1=x] \wedge [A_5=s]) \vee ([A_1=y] \wedge [A_4=n]) \Rightarrow C_1$

r_2:$([A_3=y] \wedge [A_4=n]) \vee ([A_1=x]) \Rightarrow C_2$

- Encode rules with schemata:
  - (**x***s***): C_1
  - (**y**n****): C_1
  - (****yn****): C_2
  - (**x*******): C_2

Attributes	New Values
A_1	`x,y,z,*`
A_2	`x,y,z,*`
A_3	`y,n,*`
A_4	`m,n,p,*`
A_5	`r,s,t,u,*`
A_6	`y,n,*`

- **Example**: Randomly generated population of classifiers Q

Q_1(*****ms***): 1, $s_1 = 12.3$

Q_2(****y**n**): 0, $s_2 = 10.1$

Q_3(**xy******): 1, $s_3 = 8.7$

Q_4(***z******): 0, $s_4 = 2.3$

- ***Strength s*** shows fitness of a rule based on some training data
- Class C_1: d=1
- Class C_2=not(C_1): d=0

Classification with GAs: Operators

- Fitness-Function:
  - ***Accuracy rate***
  - Complexity of the rule, ***completeness***, ***correctness***
- Mutation:
  - Change random character within the given domain,
  - Goal is to keep parents strength: $Q_{3M}\mathtt{(x{*}{*}{*}{*}{*})}{:}1, \quad s_{3M} = 8.7$
- Crossover:
  - Generate a random crossover-position point,
  - Strength of the offspring is an average (possibly weighted) of the parents‘ strength

$Q_1\mathtt{({*}{*}{*}ms{*})}{:}1$

$Q_2\mathtt{({*}{*}y{*}{*}n)}{:}0$

$\Downarrow$

$Q_{1c}\mathtt{({*}{*}{*}{*}{*}n)}{:}0$

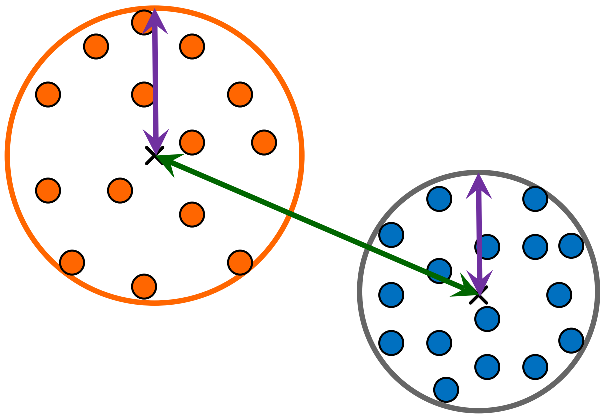
$Q_{2c}\mathtt{({*}{*}yms{*})}{:}1$

Classification with GAs: The GIL System

- Genetic-Based Inductive Learning (GIL):
  - For each attribute: translate symbolic classifiers into binary strings
  - Length is equal to number of possible values for given attribute
    - Symbolic: (**`x***r*`**) ∨ (**`y**n**`**):1
    - Binary: (100|111|11|111|1000|11 ∨ 010|111|11|010|1111|11)
- Operators:
  - Rule ***Exchange***: Crossover operator for rule sets in parents
  - Rule ***Generalization***: Unary OR operator
  - Rule ***Specialization***: Unary AND operator
  - Rule ***Split***: One parent can produce two specialized children

Clustering with Genetic Algorithms

- Idea:
  - Individuals explore the search space of "good" medoids or centroids
  - ***Generate and test*** approach



- Fitness Function:
  - Reflect the ***quality*** of the ***clustering***
  - **E.g**. (intra-cluster-distance – inter-clusters-distance)
- Clustering with GAs can be achieved with
  - … a suitable encoding scheme, and
  - … appropriate operators for crossover and mutation

Clustering with GAs: Encoding Choice

Assume three clusters are defined for a given dataset s:

- Binary encoded dataset s (Medoid-based):
  - [0100001010]
  - or 3×10 matrix:

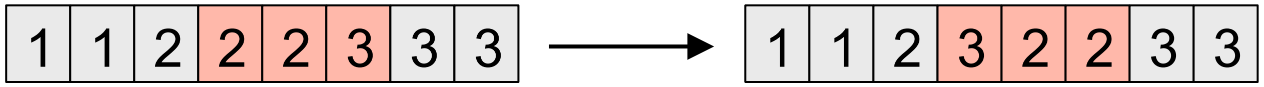
1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1

- Integer encoded dataset s:
  - [1111222233]
- Real-value encoded dataset s:
  - [1.5, 1.5, 10.5, 1.5, 5.0, 5.5]

	Sample	Feature 1	Feature 2	Cluster
	1	1	1	C1
→	2	1	2	C1
	3	2	1	C1
	4	2	2	C1
	5	10	1	C2
	6	10	2	C2
→	7	11	1	C2
	8	11	2	C2
→	9	5	5	C3
	10	5	6	C3

Clustering with GAs: Genetic Operators Choice

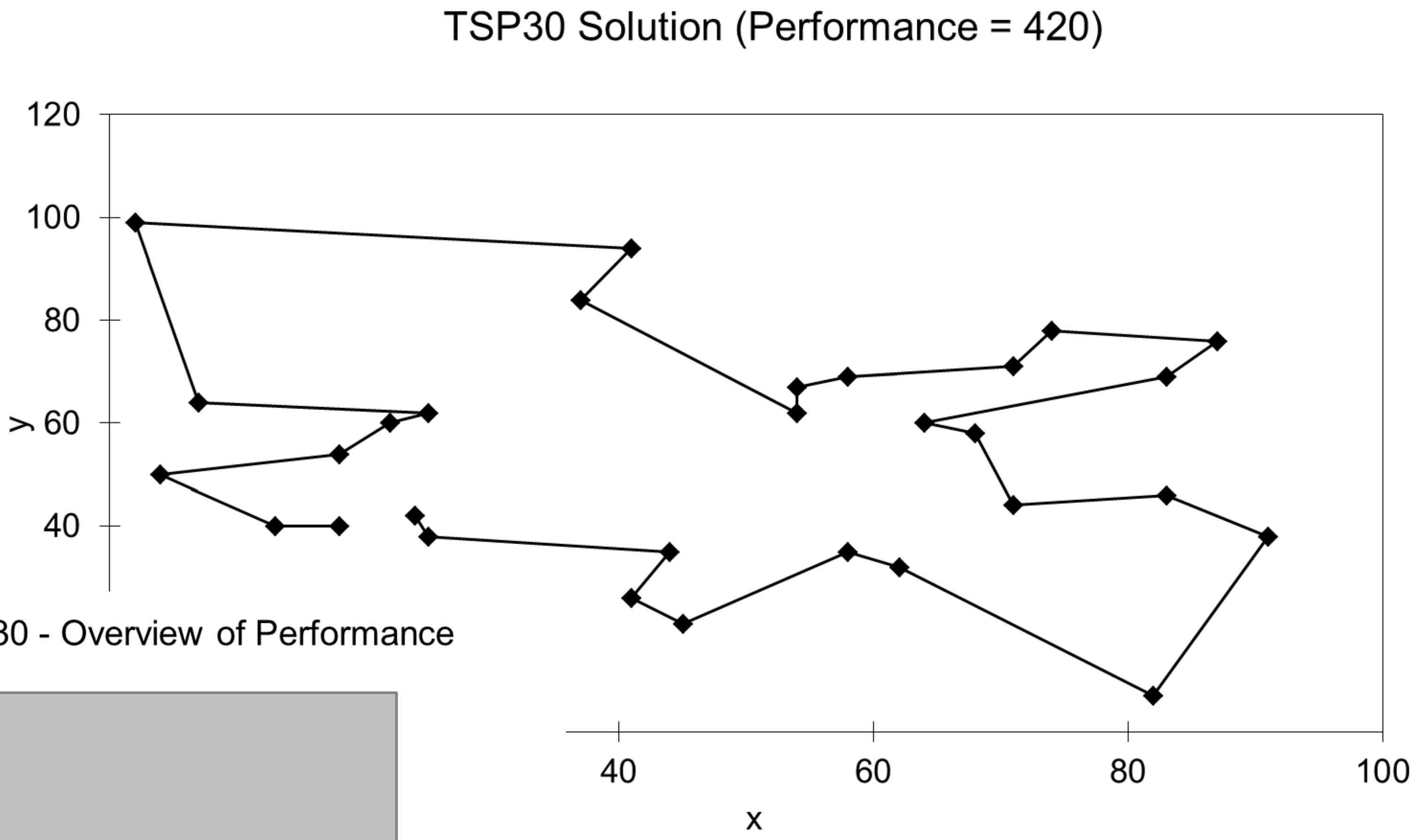
- Standard GAs: Operators work on basic arithmetic rules
- Need for many Clustering problems:
  Operators must ***maintain*** the ***representation***
  - Dataset s: Recombination must maintain the number of clusters
  - Example Crossover: ***Partially Mapped Crossover***
  - Example Mutation: ***Scramble Mutation***
    - Scramble the values between two random positions in the string

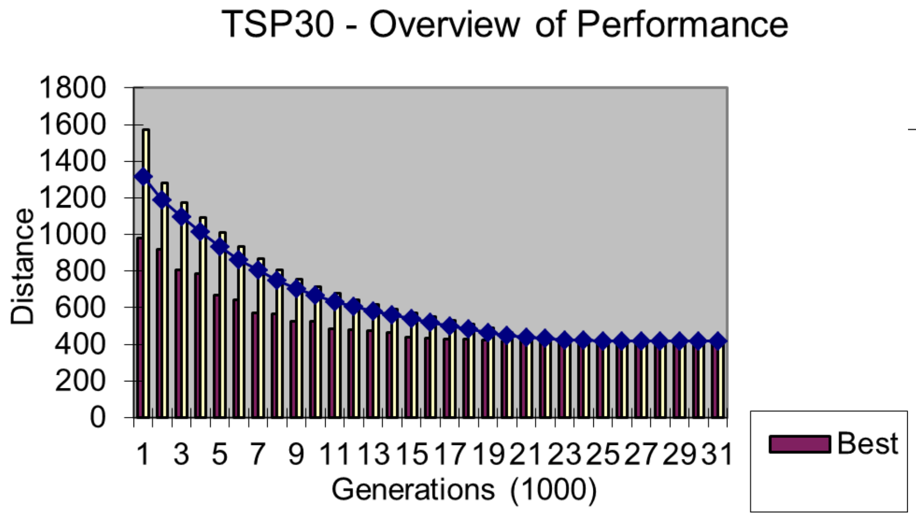


  - Classic recombination can also be used, if the fitness function ***tests for validity*** of the created offspring

$$f(C_1, C_2, \ldots, C_k) = \sum_{j=1}^{k} \sum_{x_i \in C_j} \left| x_i - z_j \right|^2$$

Example: Solving the Travelling Salesman Problem

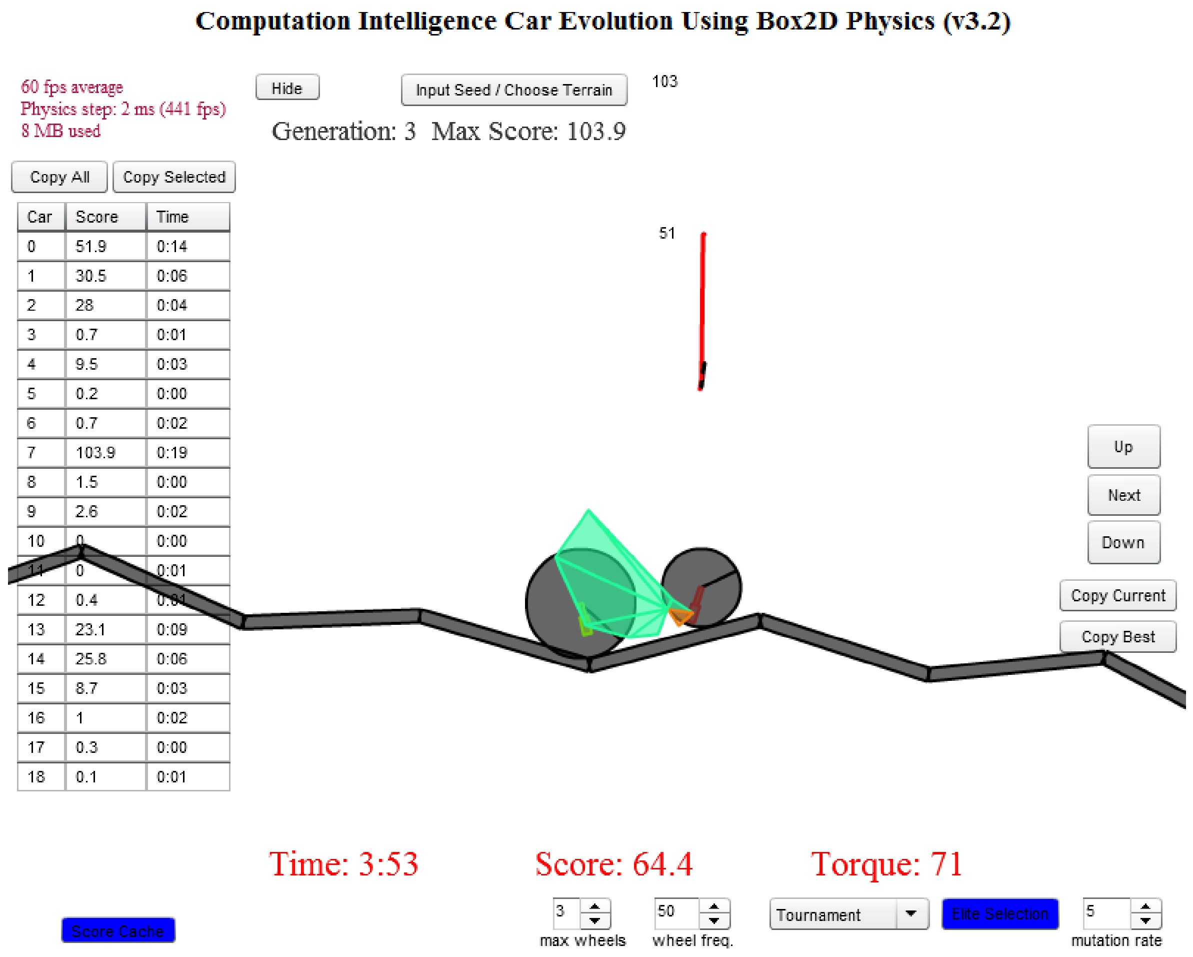




Example program:
http://www.lalena.com/AI/Tsp/

Example: Evolving Vehicle Structures

http://boxcar2d.com/



Computation Intelligence Car Evolution Using Box2D Physics (v3.2)

60 fps average
Physics step: 2 ms (441 fps)
8 MB used

Generation: 3 Max Score: 103.9

Car	Score	Time
0	51.9	0:14
1	30.5	0:06
2	28	0:04
3	0.7	0:01
4	9.5	0:03
5	0.2	0:00
6	0.7	0:02
7	103.9	0:19
8	1.5	0:00
9	2.6	0:02
10	0	0:00
11	0	0:01
12	0.4	0:01
13	23.1	0:09
14	25.8	0:06
15	8.7	0:03
16	1	0:02
17	0.3	0:00
18	0.1	0:01

Time: 3:53 Score: 64.4 Torque: 71

max wheels: 3
wheel freq.: 50
Tournament
mutation rate: 5

Genetic Algorithms – Discussion

- Applications:
  - Combinatorial optimization, e.g. classification, clustering
  - Design, e.g. vehicles, control behaviour, neural networks
  - … and many more
- Strengths:
  - General heuristic
  - Able to find good solutions in feasible computing time
  - Distributed execution possible
- Weaknesses:
  - Cannot guarantee optimal solutions
  - Long runtimes
  - Success depends also on used parameters

Fuzzy Concepts and Data Mining

- Images from one of our lab….
- Most phenomena everyday are imprecise…

Fuzzy Systems

- Most phenomena we encounter everyday are ***imprecise***
- The imprecision may be associated with their shapes, position, color, texture, semantics
- Fuzziness primarily describes ***uncertainty*** (partial truth) and ***imprecision***
  - The key idea of fuzziness comes from the multi-valued logic: Everything is a ***matter of degree***
  - Imprecision occurs in several forms, e.g. as a semantic ambiguity

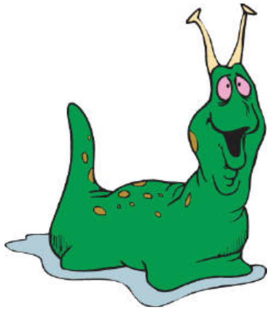
Lotfi Zadeh

- "Fuzzy Sets" paper published in 1965
- Key concept is that of ***membership values***: extent to which an object meets vague or imprecise properties
- Membership function: membership values over domain of interest
- Fuzzy set operations
- Awarded the IEEE Medal of Honor in 1995

Traditional Representation of Logic

Slow

Speed = 0



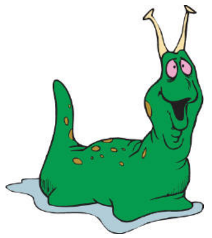
Fast

Speed = 1

```
bool speed;
get the speed
if ( speed == 0) {
//  speed is slow
}
else {
//  speed is fast
}
```

Fuzzy Logic Concept Representation

- Problem is represented in terms of fuzzy sets
- What are fuzzy sets?



Slowest

[0.0 – 0.25]

Slow

[0.25 – 0.50]

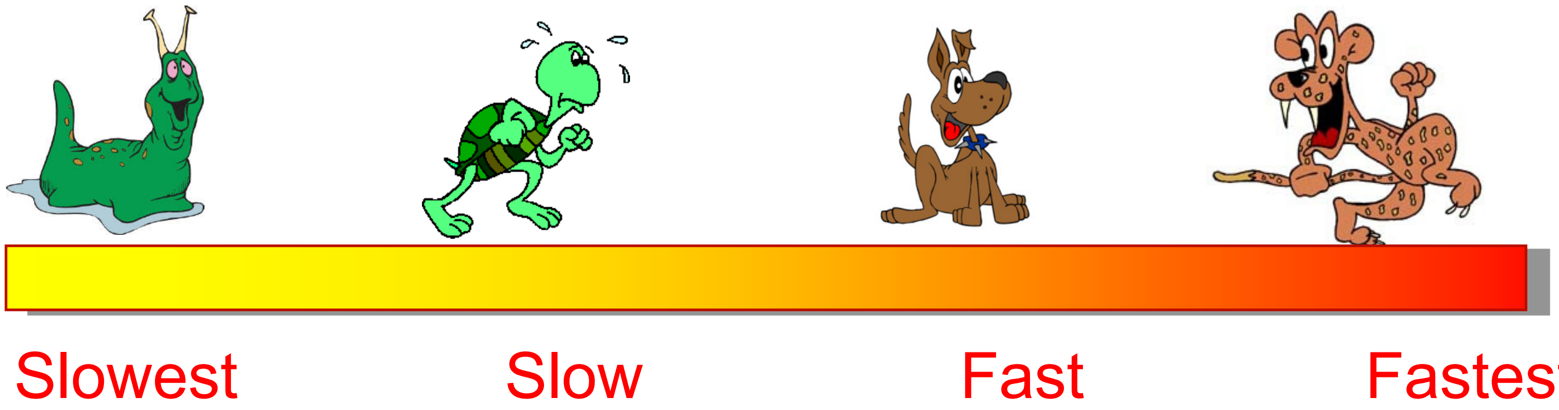
Fast

[0.50 – 0.75]

Fastest

[0.75 – 1.00]

Fuzzy Logic Concept Representation Cont.

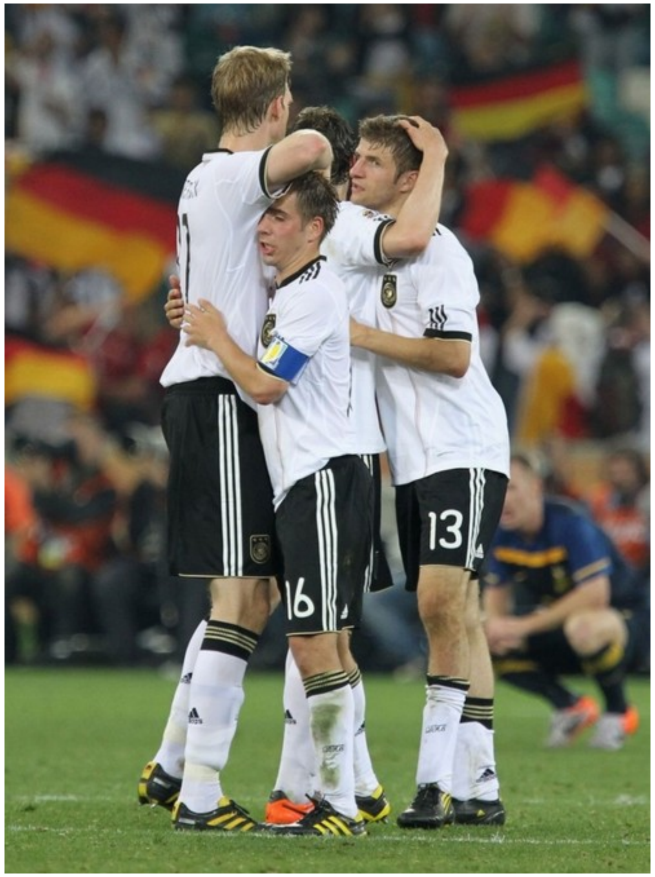


```
float speed;
get the speed
if ((speed >= 0.0)&&(speed < 0.25)) {
//  speed is slowest
}
else if ((speed >= 0.25)&&(speed < 0.5))
{
//  speed is slow
}
else if ((speed >= 0.5)&&(speed < 0.75))
{
//  speed is fast
}
else // speed >= 0.75 && speed < 1.0
{
//  speed is fastest
}
```

One could attempt to implement this with more classes and sharp boundaries, … but would not catch the point!

Fuzzy Logic Representation: Matter of Degree

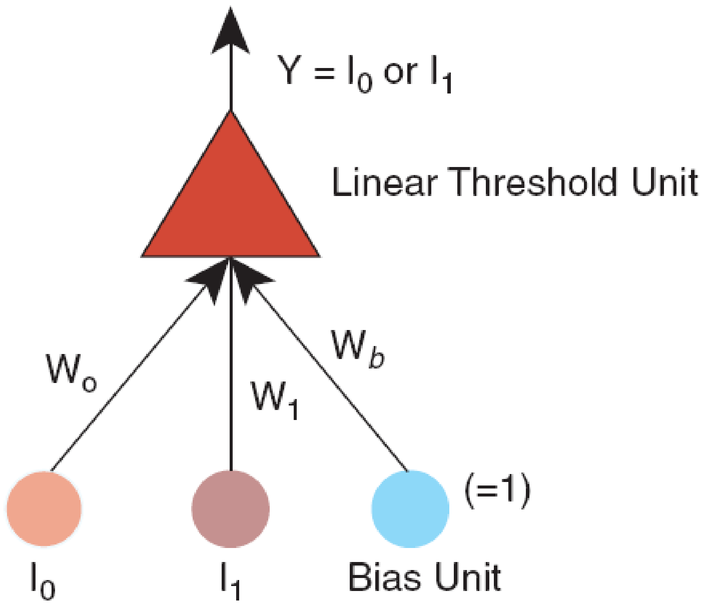
- Is Philipp Lahm a short person?



… depends!

Fuzzy Logic and Neural Networks: Link to previous Lecture – Part 1

- From logic and (neural) threshold units to fuzzy logic and neural units
- Both model aspects of the brain.
  - Fuzzy systems (mind modeling)
  - Neural Networks (brain modeling)
- Both used to create behavioral systems based on ***graded representation***

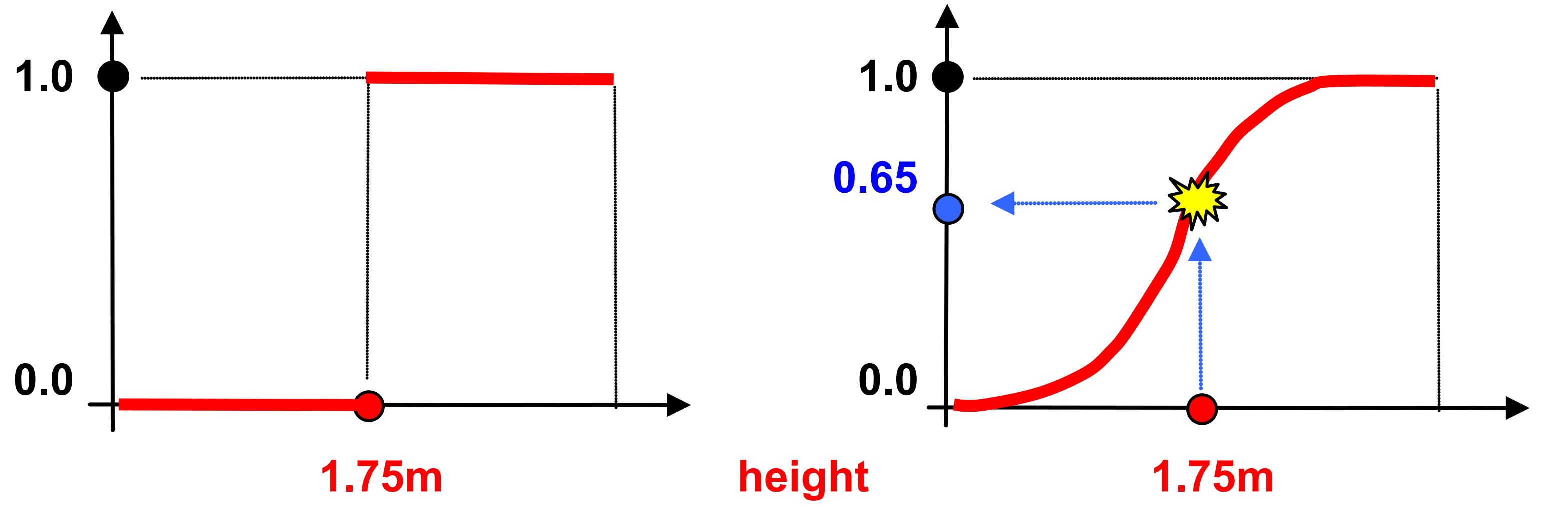


Simple, single-unit adaptive network

Fuzzy Sets (1)

- The notion of membership in fuzzy sets becomes a matter of degree (real number in the closed interval [0,1])
- Membership of an element in fuzzy set is measured by a *function that attempts to describe* ***vagueness***

crisp (classical) set A **A = set of TALL people** **fuzzy set A**



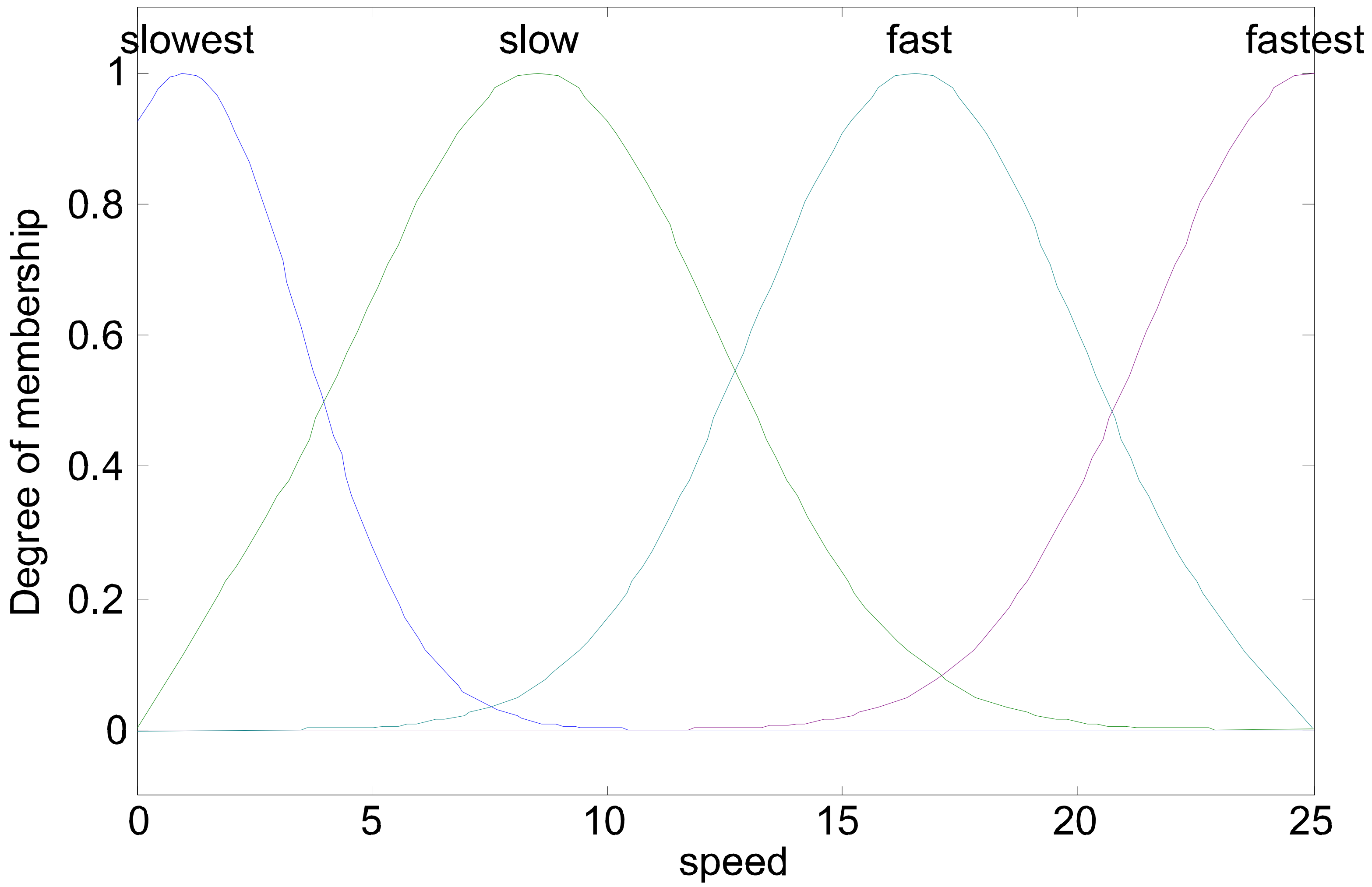
Fuzzy Sets (2)

- A Fuzzy Set is a set with smooth boundaries
- Fuzzy Set Theory generalizes classical set theory to allow ***partial membership***
- **Fuzzy Set** A is a universal set U determined by a membership function $\mu_A(x)$ that assigns to each element $x \in U$ a number A(x) in the unit interval [0,1]:

$$\underset{\text{Fuzzy set}}{A} = \{(x, \underset{\text{Membership function}}{\mu_A}[x]) \mid x \in \underset{\text{Collection of objects/Universe}}{U}\}$$

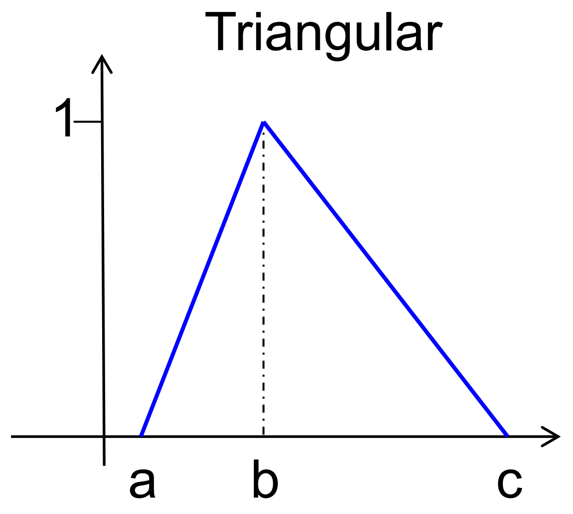
  - Universal set U (Universe of Discourse) contains all possible elements of concern for a particular application

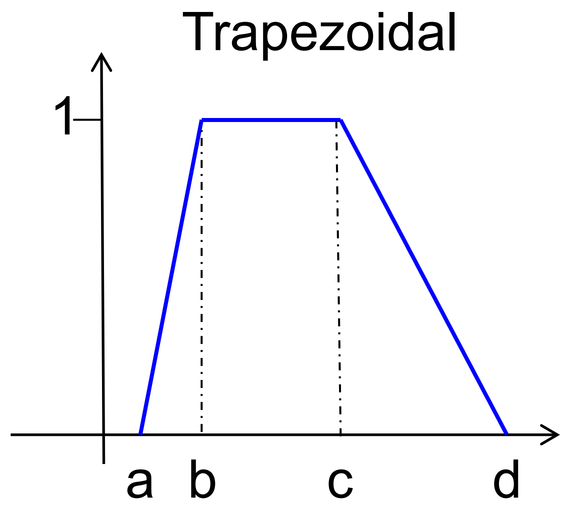
Membership Functions

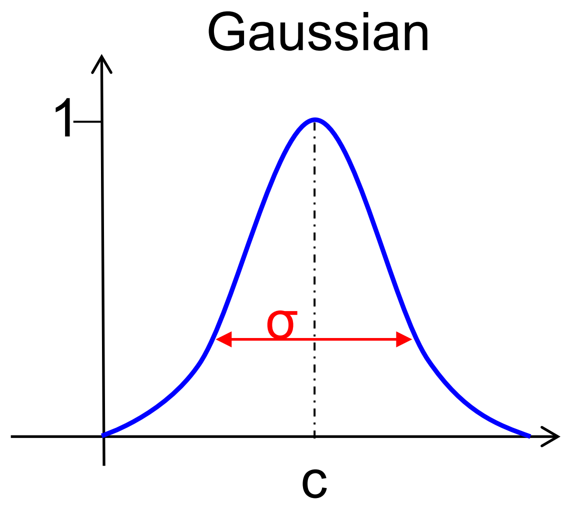


Now we catch the point!

Membership Functions: Shape







$$\mu(x) = f(x,a,b,c) = \begin{cases} 0 & \text{for} \quad x \le a \\ (x-a)/(b-a) & \text{for} \quad a \le x \le b \\ (c-x)/(c-b) & \text{for} \quad b \le x \le c \\ 0 & \text{for} \quad c \le x \end{cases}$$

$$\mu(x) = f(x,a,b,c,d) = \begin{cases} 0 & \text{for} \quad x \le a \\ (x-a)/(b-a) & \text{for} \quad a \le x \le b \\ 1 & \text{for} \quad b \le x \le c \\ (d-x)/(d-c) & \text{for} \quad c \le x \le d \\ 0 & \text{for} \quad d \le x \end{cases}$$

$$\mu(x) = f(x,c,\sigma) = e^{-1/2\cdot((x-c)/\sigma)^2}$$

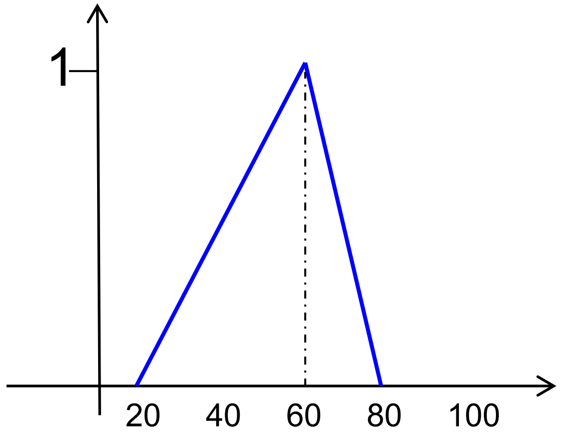
Membership Functions: Parametrization

$$\mu(x) = f(x,a,b,c) = \begin{cases} 0 & for \quad x \le a \\ (x-a)/(b-a) & for \quad a \le x \le b \\ (c-x)/(c-b) & for \quad b \le x \le c \\ 0 & for \quad c \le x \end{cases}$$

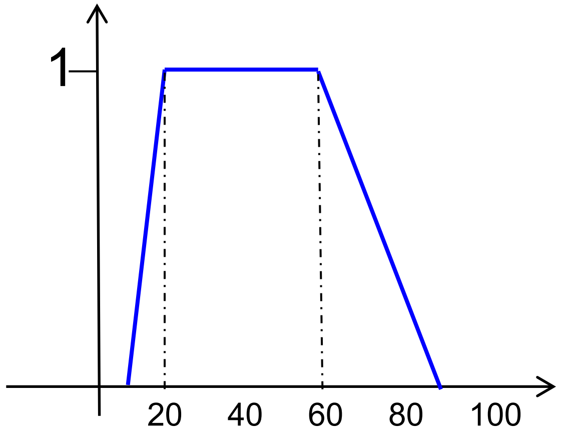
$$\mu(x) = f(x,a,b,c,d) = \begin{cases} 0 & for \quad x \le a \\ (x-a)/(b-a) & for \quad a \le x \le b \\ 1 & for \quad b \le x \le c \\ (d-x)/(d-c) & for \quad c \le x \le d \\ 0 & for \quad d \le x \end{cases}$$

$$\mu(x) = f(x,c,\sigma) = e^{-1/2\cdot((x-c)/\sigma)^2}$$

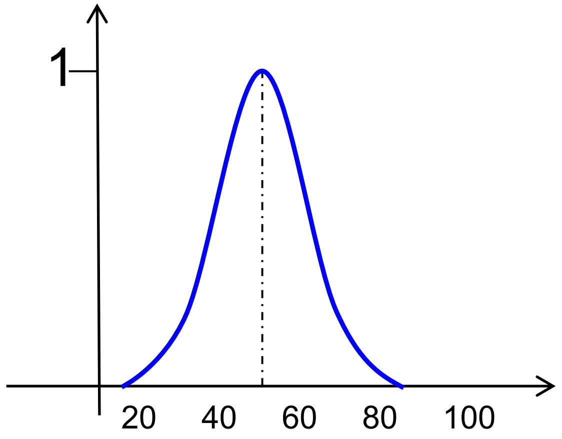
Triangular(x,20,60,80)



Trapezoidal(x,10,20,60,90)

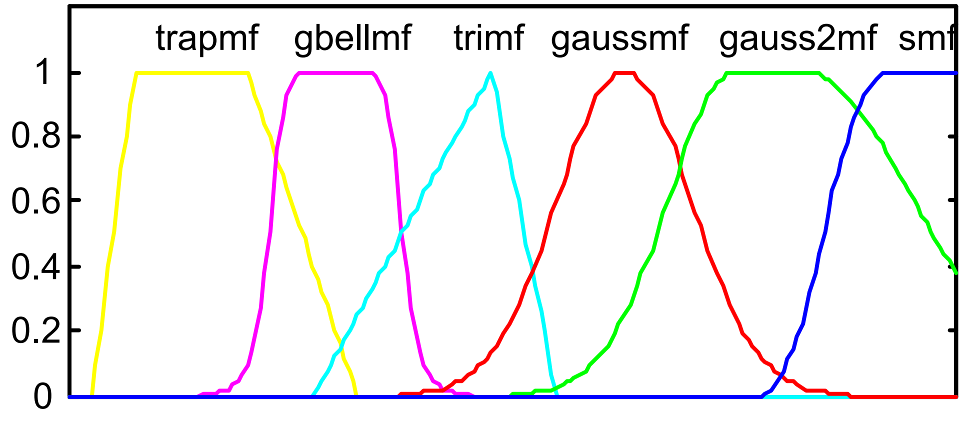


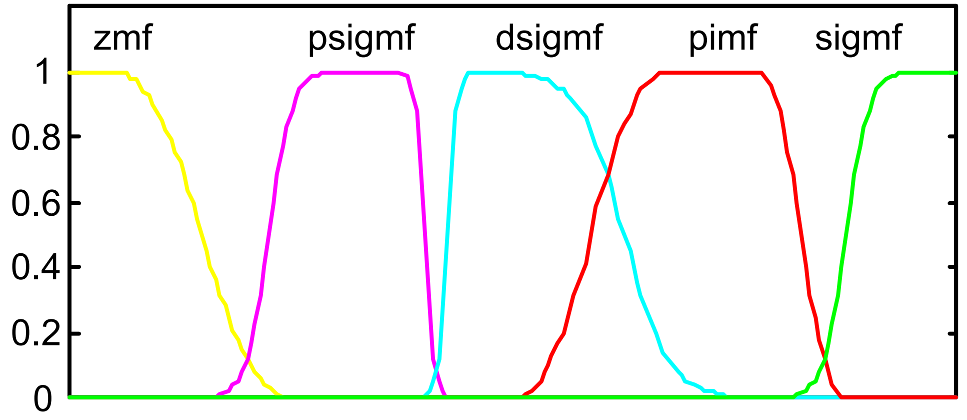
Gaussian(x,50,20)



Membership Functions

- Defines the strength of membership of each value in the set
- Many different membership functions exist
  - Trapezoid, bell, triangular, Gaussian, sigmoid
  - R-/L-functions, psi, … user defined





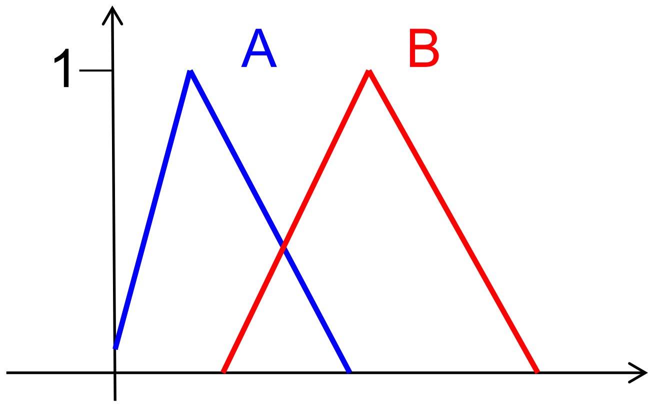
Basic Relations of Fuzzy Sets

- Fuzzy Equality: A = B
  - In traditional logic, sets containing the same members are equal: {a,b,c} = {a,b,c}
  - In fuzzy logic, however, two sets are equal if and only if all elements have identical membership values:

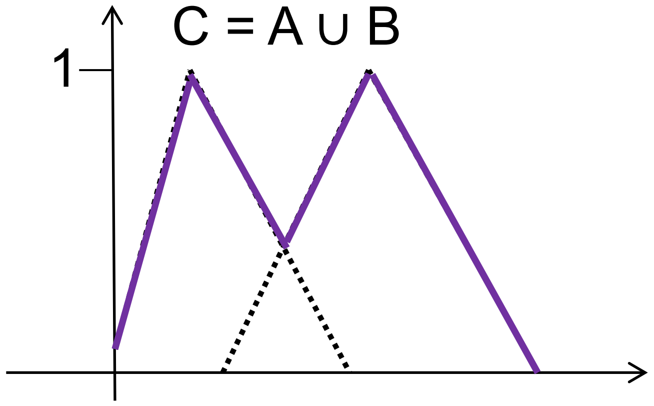
    {0.2/a,0.6/b,0.8/c} = {0.2/a,0.6/b,0.8/c}

- Fuzzy Containment: A $\subseteq$ B
  - In traditional logic: "**if and only if all elements in A are also in B**"
  - In fuzzy logic, containment means that the membership values for each element in a subset is less than or equal to the membership value of the corresponding element in the superset
  - Adding a ***hedge*** can create a subset or superset

Fuzzy Union



- In traditional logic, all elements in either (or both) set(s) are included ($\rightarrow$ OR)



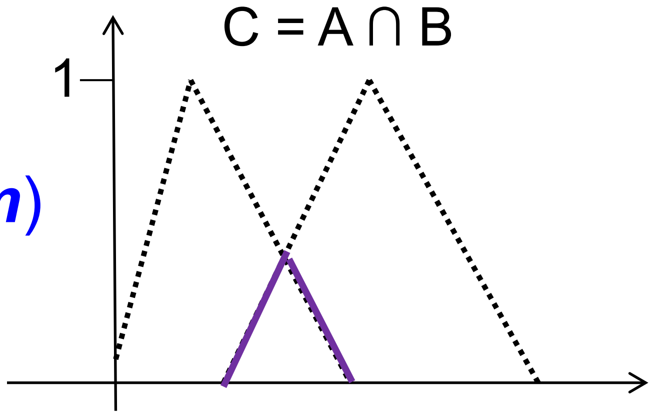
- In fuzzy logic, union is the ***maximum*** set membership value

- **Example**: If $m_A(x) = 0.7$ and $m_B(x) = 0.9$ then $m_{A \cup B}(x) = 0.9$

Fuzzy Intersection

- In standard logic, the intersection of two sets contains those elements in ***both*** sets ($\rightarrow$ AND)

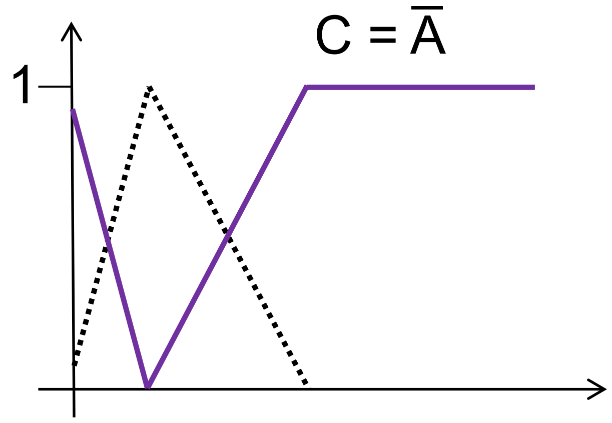
- In fuzzy logic, the weakest element (e.g. ***minimum***) determines the degree of membership in the intersection



- **Example**: If $m_A(x) = 0.5$ and $m_B(x) = 0.3$ then $m_{A\cap B}(x) = 0.3$

Fuzzy Complement

- In traditional logic, the complement of a set is all of the elements ***not*** in the set
- In fuzzy logic, the value of the complement of a membership is (1 - membership_value)
- **Example**: If $m_A(x) = 0.8$ then $m_{\bar{A}}(x) = 0.2$



- The law of the excluded middle doesn't hold!

  "Reality flourishes on ambiguity."

  – L. Zadeh

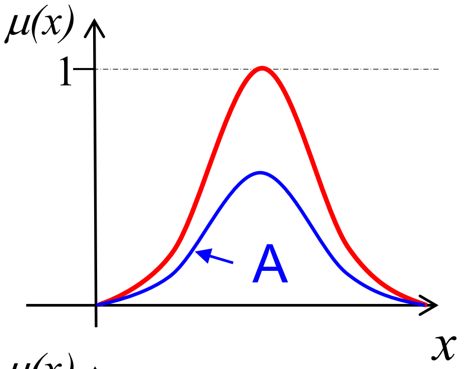
Summary of Fuzzy Relations and Operators

- If μ_A and μ_B represent the degrees to which x is a member of fuzzy sets A and B, respectively,
  and the sets have common domains, then the following are the basic relations of fuzzy sets:
  - Equality $A = B$ if $\forall x : \mu_A[x] = \mu_B[x]$
  - Containment $A \subseteq B$ if $\forall x : \mu_A[x] \leq \mu_B[x]$
- The following are the basis operations on fuzzy sets:
  - Intersection $\mu_{A \cap B}(x) = \min(\mu_A[x], \mu_B[x])$
  - Union $\mu_{A \cup B}(x) = \max(\mu_A[x], \mu_B[x])$
  - Complement $\mu_{\overline{A}}(x) = 1 - \mu_A[x]$

Unary Fuzzy-Set Operators

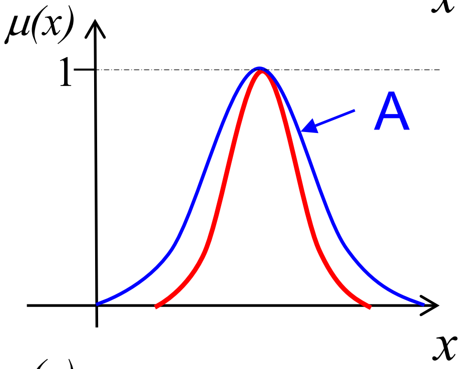
- Normalization:
  - Converts subnormal to normalized set

$$NormA(x) = \left\{ \left(\text{x}, \frac{\mu_A[x]}{\text{hgt}[x]} = \frac{\mu_A[x]}{\max(\mu_A[x])} \right), \text{where } x \in X \right\}$$



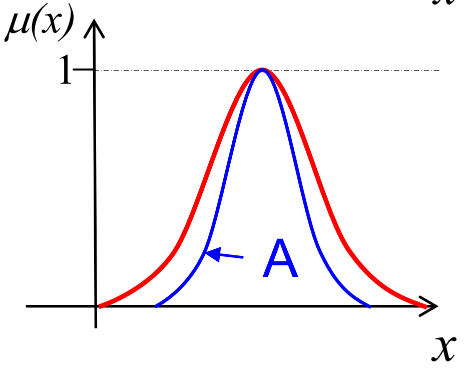
- Concentration:
  - Concentrate the MF to smaller values

$$ConA(x) = \{(x, \mu_A^{\varphi}[x]), \text{where } x \in X, \varphi \geq 1\}, \text{e.g. } \varphi = 2$$



- Dilation:
  - Expand the MF to larger values

$$DilA(x) = \{(x, \mu_A^{\varphi}[x]), \text{where } x \in X, \varphi \in [0,1]\}, \text{e.g. } \varphi = ½$$



Linguistic Variables and Hedges

- Linguistic variables are used to describe terms or concepts with vague or fuzzy values. These values are represented within the fuzzy sets.

  *IF Temperature is **Low** …*

  *THEN Heater Fan Speed is **Fast***

- Hedges are fuzzy set qualifiers used to modify the shape of fuzzy sets. They can include adverbs such as **very**, **somewhat**, **more or less** and **slightly**. Hedges manipulate the fuzzy sets by mathematical operations that can ***concentrate*** or ***dilate*** set membership.
  - All purpose modifiers, such as *very, quite or extremely*
  - Truth-values, such as *quite true* or *mostly false*
  - Probabilities, such as *likely* or *not very likely*
  - Quantifiers, such as *most*, *several or few*
  - Possibilities, such as *almost impossible* or *quite impossible*

Fuzzy Rules

- From a knowledge representation viewpoint,
  a fuzzy IF-THEN rule is a scheme for capturing knowledge that involves imprecision
- If we know a fact (***premise***), then we can infer another fact (***conclusion***)
- The building blocks for ***fuzzy IF-THEN rules*** are fuzzy sets
- A ***fuzzy system*** (FS) is constructed from a collection of fuzzy IF-THEN rules

Fuzzy Rule Format

- The rule

  "**IF** Temperature **is** Low **THEN** Heater Fan Speed **is** Fast"

  has a form:

  IF x **is** A **THEN** y **is** B,

  where fuzzy sets "Low" and "Fast" are labeled by A and B

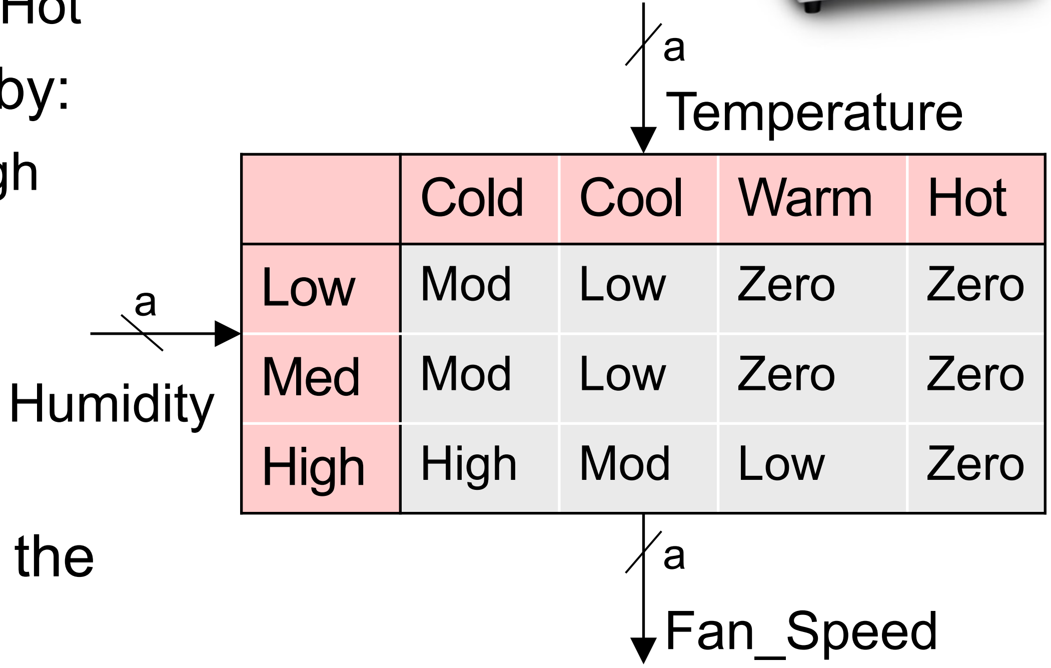
- A and B ***characterize*** fuzzy propositions about variables x and y

- Most of the information involved in human communication uses ***natural language terms*** that are often ***vague***, ***imprecise***, ***ambiguous*** by their nature

- Fuzzy sets can serve as the mathematical foundation of such vagueness of natural language

Reasoning with Uncertainty (1)

- Example task: Temperature Controller for Heating
  - Change the speed of a heater fan, based on the room temperature and humidity.
- A temperature control system (***A*$_1$**) has four settings:
  - Cold, Cool, Warm, and Hot
- Humidity is (***A*$_2$**) defined by:
  - Low, Moderate, and High
- The **subject** (***B***) is the fan speed control:
  - Zero, Low, Moderate, and High
- Using this we can define the fuzzy sets and rules



	Cold	Cool	Warm	Hot
Low	Mod	Low	Zero	Zero
Med	Mod	Low	Zero	Zero
High	High	Mod	Low	Zero

Reasoning with Uncertainty (2)

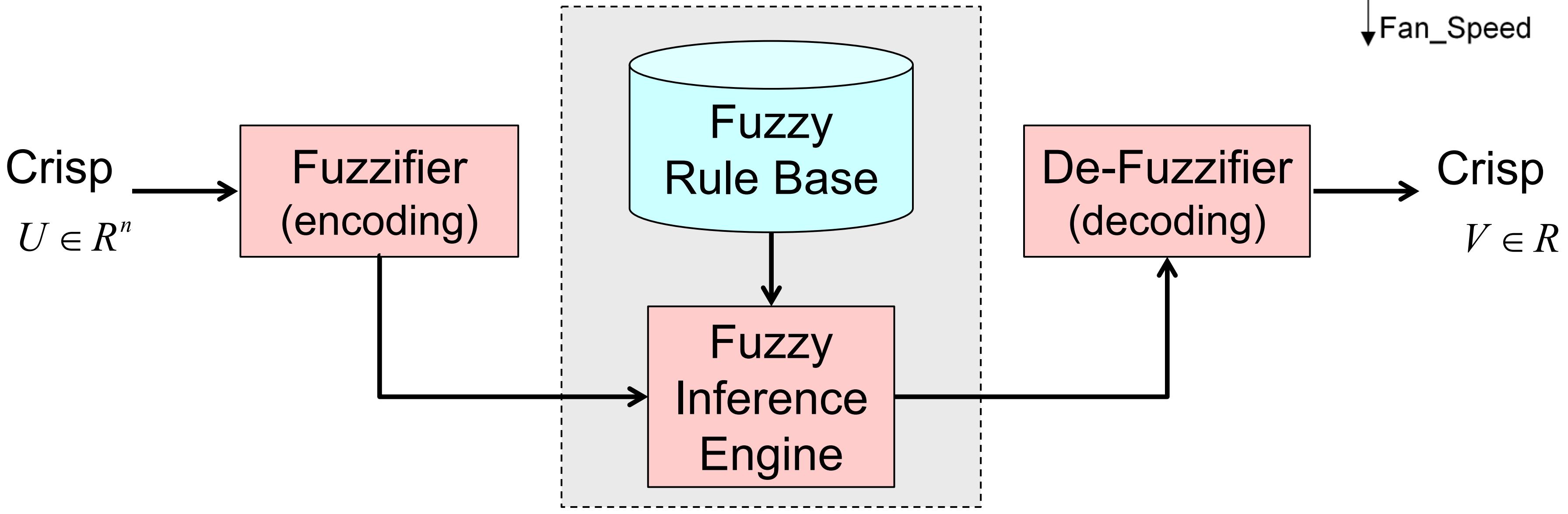
- Allows use of ***intuitive terms*** such as
  - The temperature is "cool", the humidity is "high"
- Outputs can be in terms of
  - The fan speed is "moderate", …

a Temperature

a Humidity

	Cold	Cool	Warm	Hot
Low	Mod	Low	Zero	Zero
Med	Mod	Low	Zero	Zero
High	High	Mod	Low	Zero

a Fan_Speed



Fuzzification: Determine the Rule Set

- Possible rules in the Example:
  - IF Temperature is **cold** AND Humidity is **high** THEN Speed is **high**
  - IF Temperature is **cold** AND (Humidity is **medium** OR Humidity is **low**) THEN Speed is **moderate**
  - IF Temperature is **cool** AND Humidity is **high** THEN Speed is **moderate**
  - IF Temperature is **cool** AND (Humidity is **medium** OR Humidity is **low**) THEN Speed is **low**
  - IF Temperature is **warm** AND Humidity is **high** THEN Speed is **low**
  - IF Temperature is **warm** AND (Humidity is **medium** OR Humidity is **low**) THEN Speed is **zero**
  - IF Temperature is **hot** THEN Speed is **zero**
- Rules get defined by human experience or from the data

Inference

Also called:

Max-Min inference

- Compositional Rule of Inference:

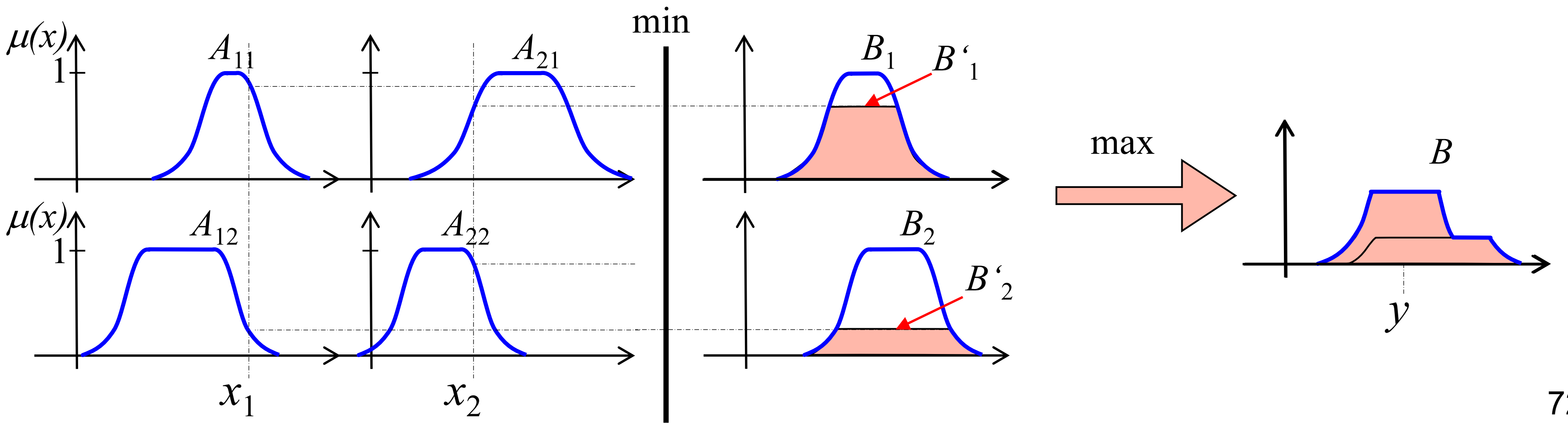
$$\mu_b[y] = \max_x \{\min(\mu_{A11}[x_1], \mu_{A21}[x_2], \ldots, \mu_{An1}[x_n]), \min(\mu_{A12}[x_1], \mu_{A22}[x_2], \ldots, \mu_{An2}[x_n])\}$$

Operators of choice (depending on norm)

Rule 1

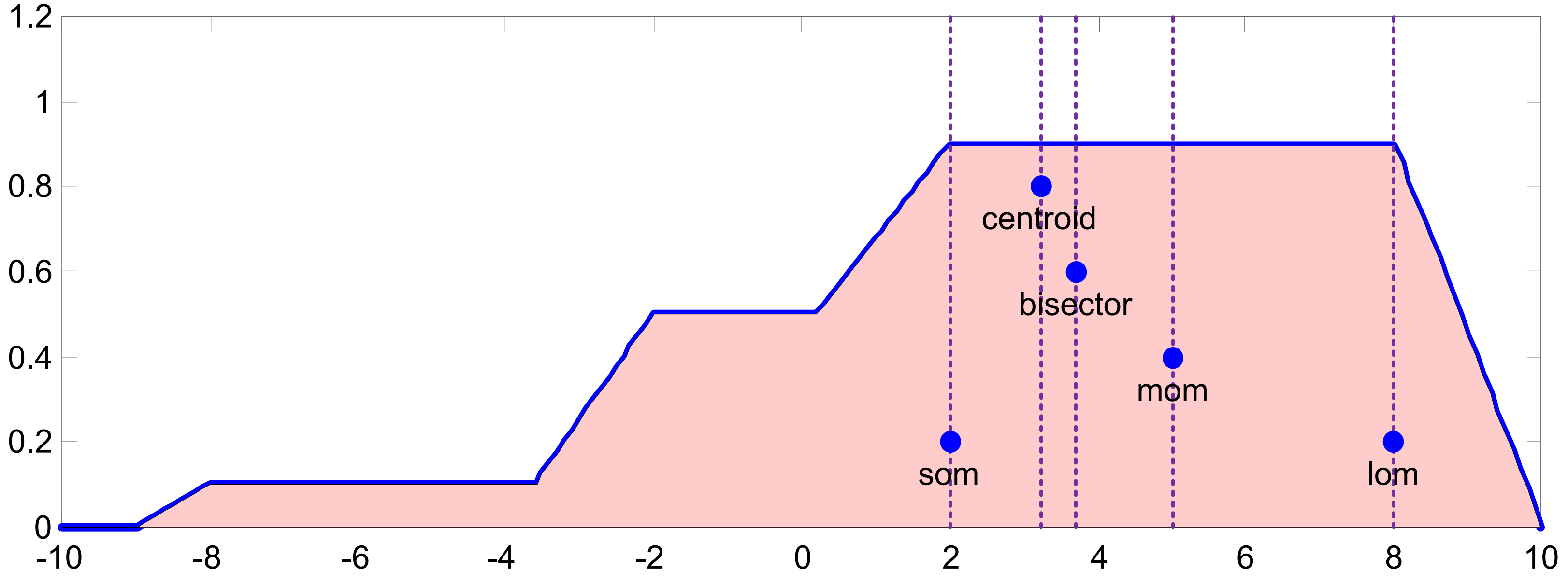
Rule 2

  - For a mapping $y = f(x)$, producing the fuzzy set $B = f(A_1, A_2, \ldots, A_n)$
  - Input is an n-dimensional vector $x = (x_1, x_2, \ldots, x_n)$ and $x \in X$

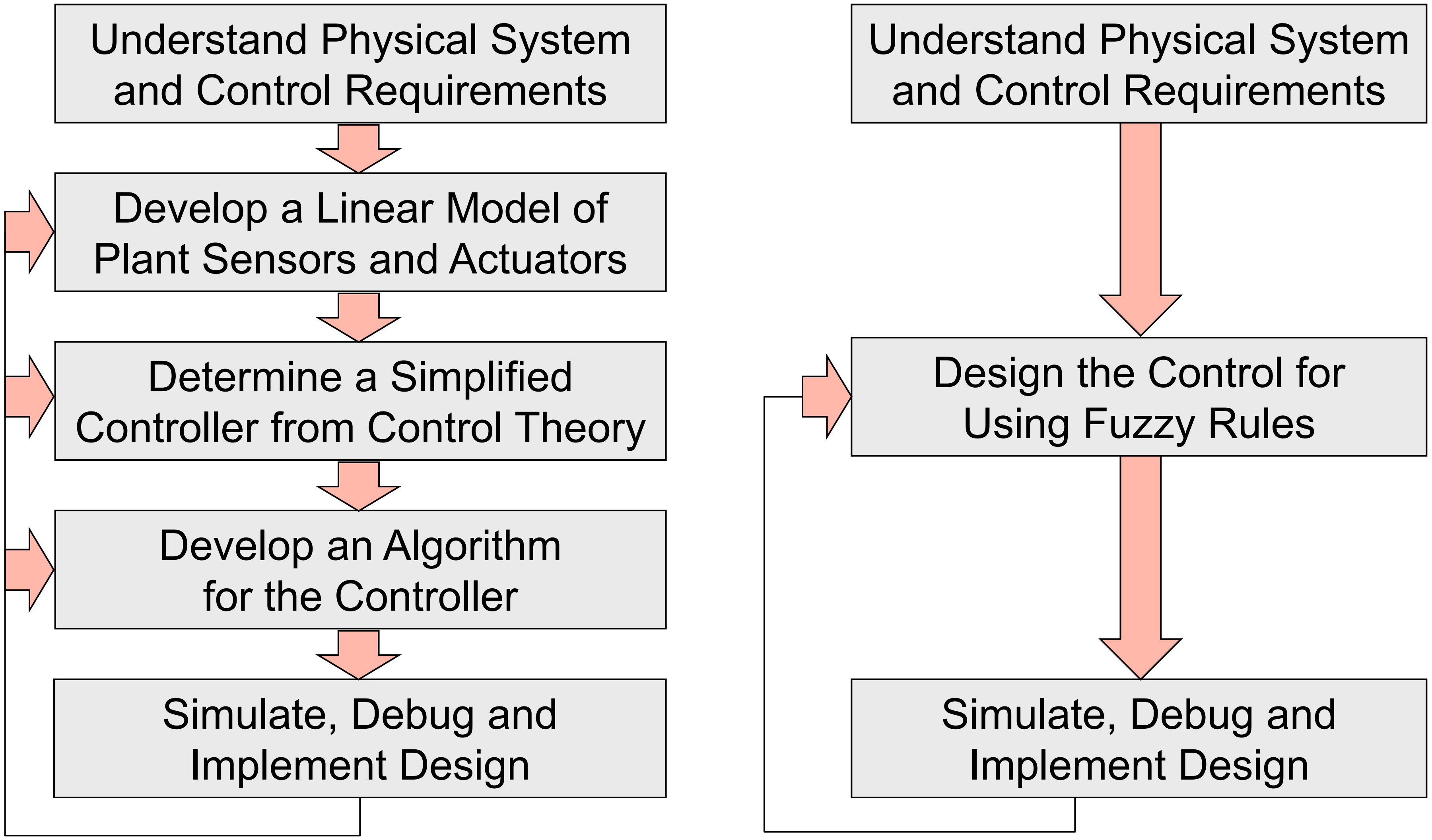


Defuzzification Methods

- Transforms fuzzy output of the inference engine to crisp output using membership functions analogous to the fuzzifier
- Commonly used techniques:
  - ***centroid*** of area
  - ***bisector*** of area
  - ***mom***: mean of maximum
  - ***som***: smallest of maximum
  - ***lom***: largest of maximum
  - …



Benefits of Using Fuzzy Logic



Understand Physical System and Control Requirements
Develop a Linear Model of Plant Sensors and Actuators
Determine a Simplified Controller from Control Theory
Develop an Algorithm for the Controller
Simulate, Debug and Implement Design
Understand Physical System and Control Requirements
Design the Control for Using Fuzzy Rules
Simulate, Debug and Implement Design

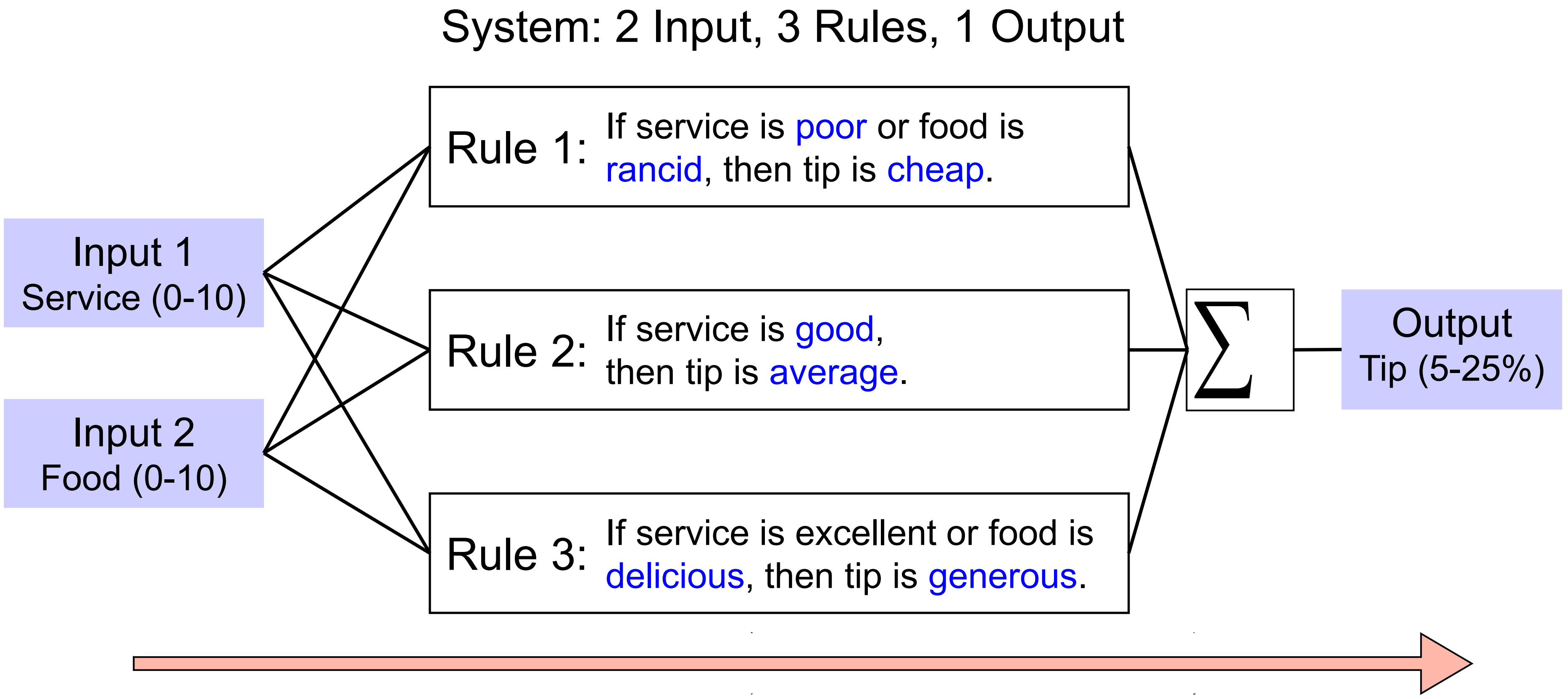
Example: Fuzzy Reasoning for Restaurant Tipping

The Basic Tipping Problem: Given a number between 0 and 10 that represents the quality of service at a restaurant (where 10 is excellent), what should the tip be?

- Fuzzy Rules

1. IF the Service is poor or the Food rancid
   THEN Tip is cheap
2. IF Service is Good
   THEN Tip is average
3. IF Service is excellent or the Food is delicious
   THEN Tip is generous

Restaurant Tipping: Fuzzy Inference Process


System: 2 Input, 3 Rules, 1 Output
Input 1
Service (0-10)
Input 2
Food (0-10)
Rule 1: If service is poor or food is rancid, then tip is cheap.
Rule 2: If service is good, then tip is average.
Rule 3: If service is excellent or food is delicious, then tip is generous.
Σ
Output
Tip (5-25%)

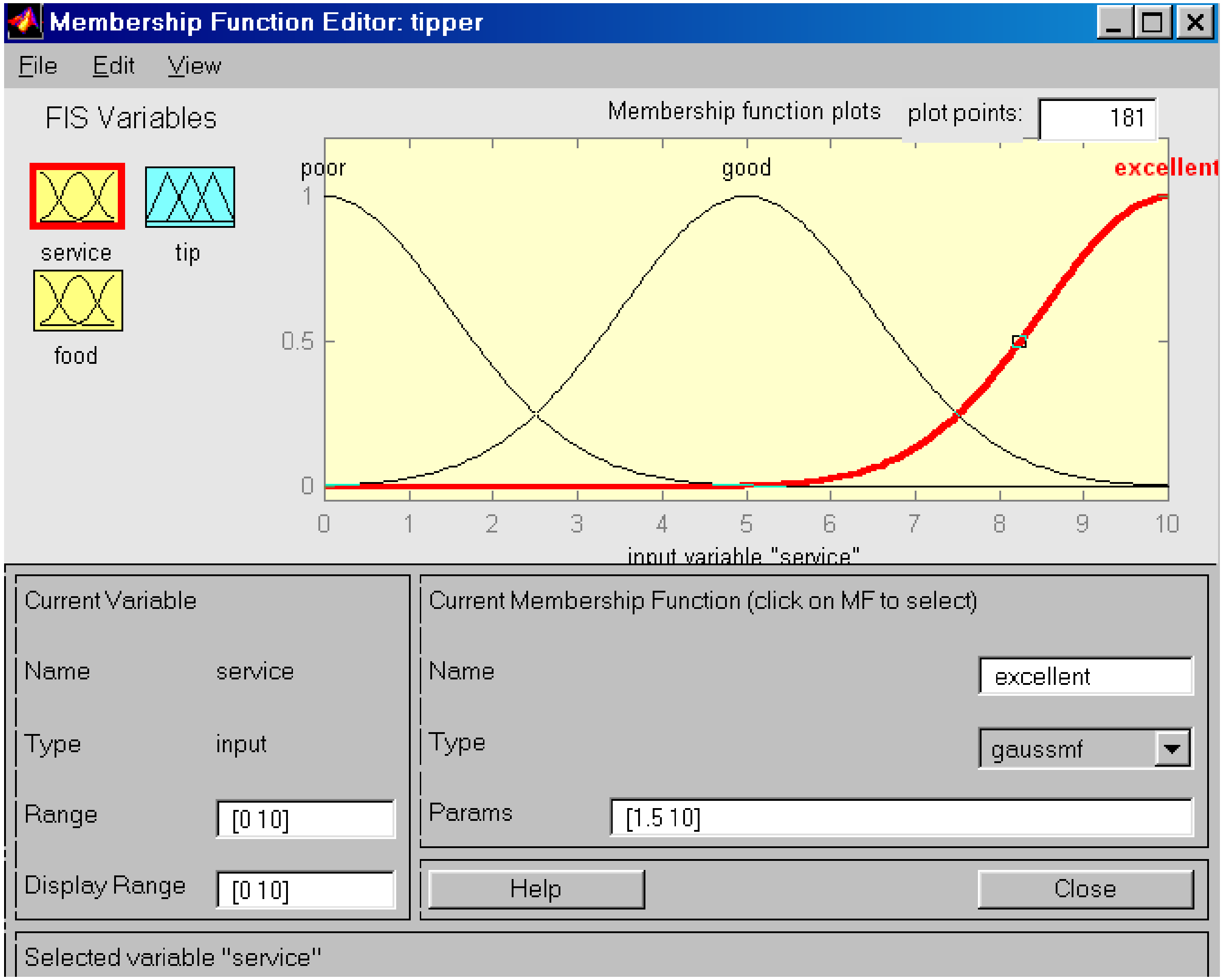
The inputs are crisp (non-fuzzy) numbers limited to a specific range

All rules are evaluated in parallel using fuzzy reasoning

The results of the rules are combined and distilled (defuzzified)

The result is a crisp number

Fuzzy Inferencing: Input Variable “service”



plot points: 181

Current Variable

Name service

Type input

Range [0 10]

Display Range [0 10]

Current Membership Function (click on MF to select)

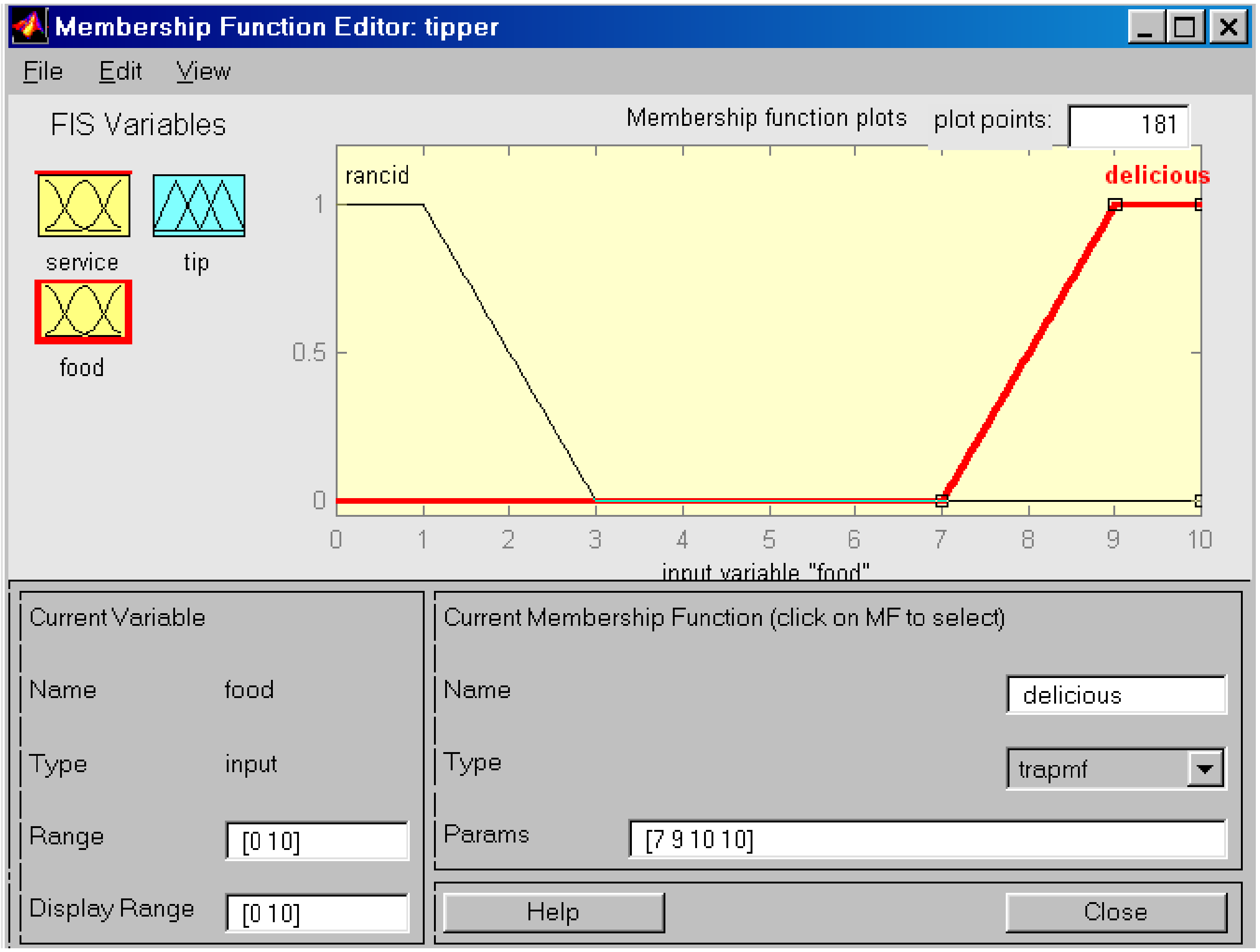
Name excellent

Type gaussmf

Params [1.5 10]

Selected variable "service"

Fuzzy Inferencing: Input variable "Food"



plot points: 181

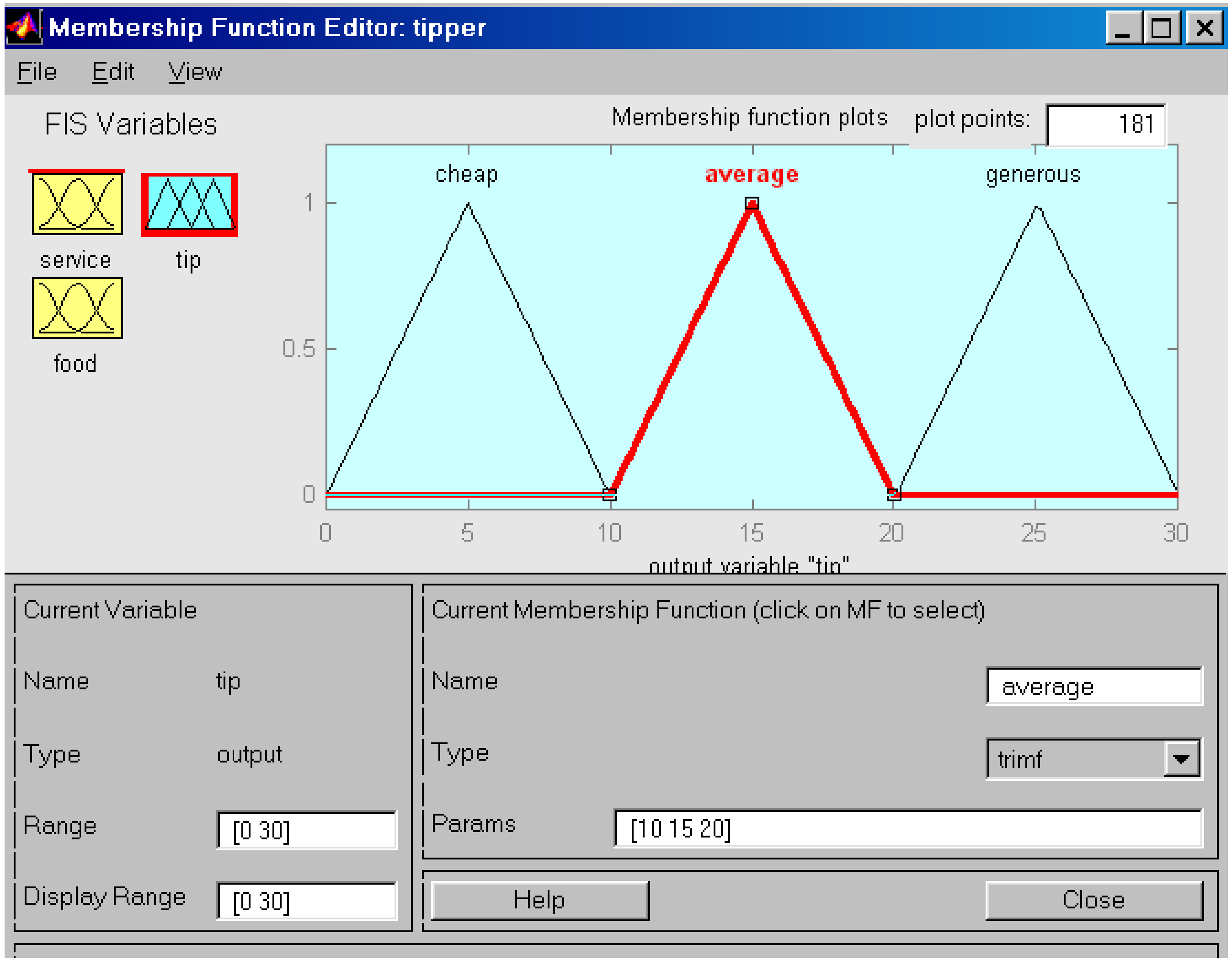
Current Variable

Name	food
Type	input
Range	[0 10]
Display Range	[0 10]

Current Membership Function (click on MF to select)

Name	delicious
Type	trapmf
Params	[7 9 10 10]

Fuzzy Inferencing: Output Variable “tip”



Current Variable

Name tip

Type output

Range [0 30]

Display Range [0 30]

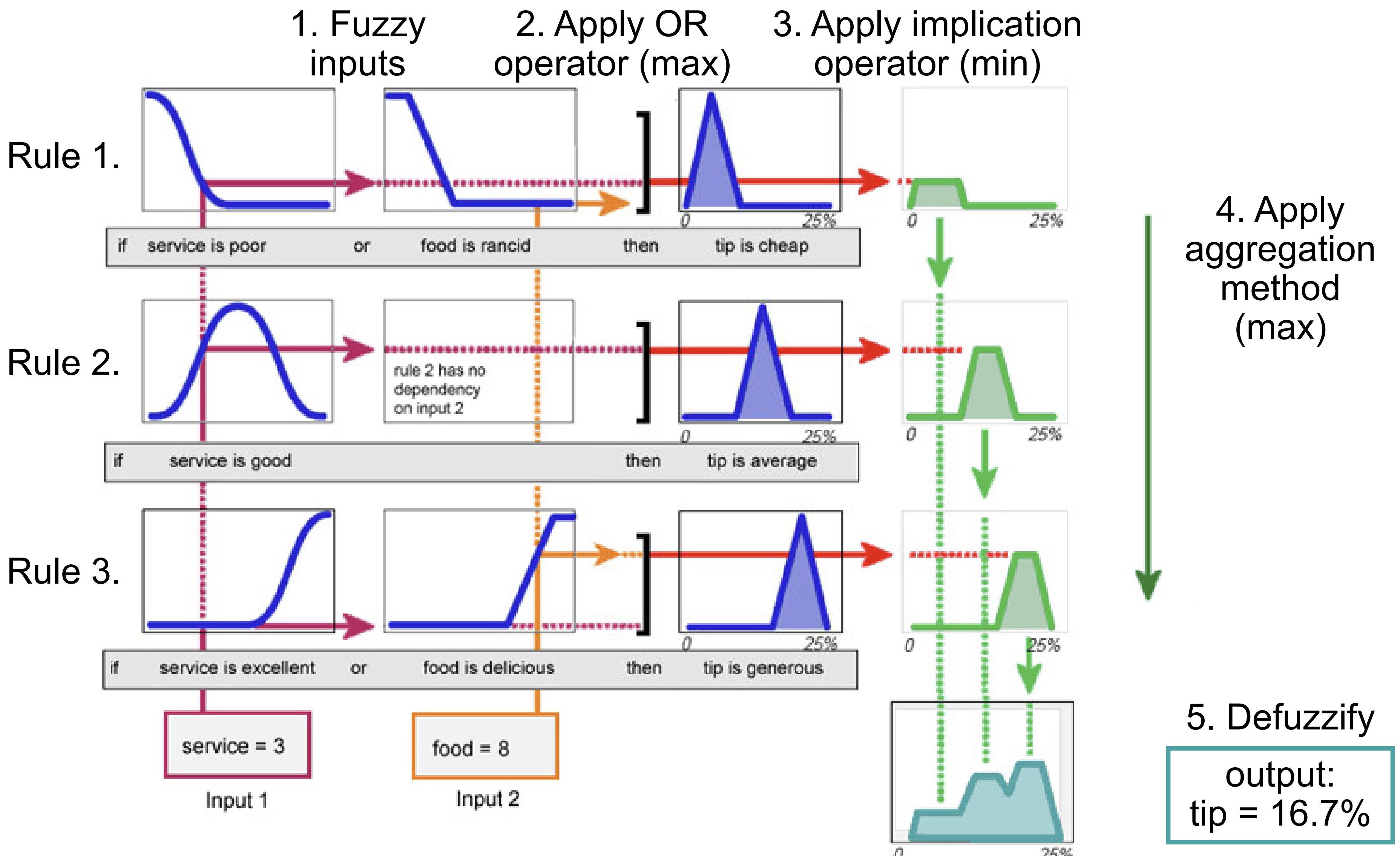
Current Membership Function (click on MF to select)

Name average

Type trimf

Params [10 15 20]

Fuzzy Inferencing: Mamdani's Method



1. Fuzzy inputs
2. Apply OR operator (max)
3. Apply implication operator (min)
4. Apply aggregation method (max)
5. Defuzzify
Rule 1.
Rule 2.
Rule 3.
if service is poor or food is rancid then tip is cheap
if service is good then tip is average
if service is excellent or food is delicious then tip is generous
rule 2 has no dependency on input 2
0
25%
service = 3
Input 1
food = 8
Input 2
output: tip = 16.7%

Example: Fuzzy Logic Control Landing

Fuzzy Logic and Neural Networks: Link to previous Lecture – Part 2

- Neural Networks: system ***learned*** from data and feedback
  - Difficult to develop an insight about the meaning associated with each neuron and each weight
  - Viewed as “black box” approach: *know what the box does but not how it is done conceptually*

- Fuzzy Logic:
  - Fuzzy rule-based ***models*** are easy to ***comprehend***
  - does not come with a learning algorithm

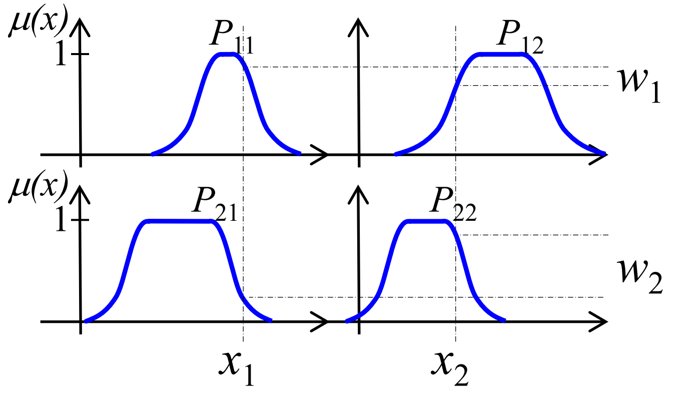
- Fuzzy Neural Network system: Learning and identification of fuzzy models
  - Example: ANFIS (adaptive-network-based fuzzy inference system) Matlab: http://www.mathworks.de/de/help/fuzzy/anfis.html

Fuzzy Logic and Neural Networks: ANFIS

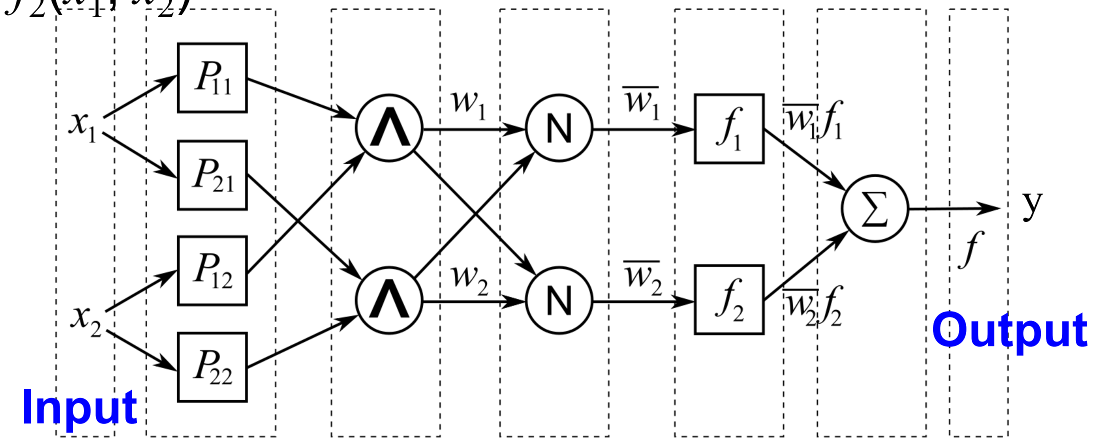
- ANFIS: construct input-output mapping based on human knowledge and input-output data pairs
  - ***Activation functions*** represent ***Gaussian*** fuzzy predicates
  - Parameters for the ***conclusions*** of the systems can be learned with ***Backpropagation*** variant
- **Example**: Rule system with input variables x_1, x_2:

If $P_{11}(x_1)$ **and** $P_{12}(x_2)$ **then** $f_1(x_1, x_2)$

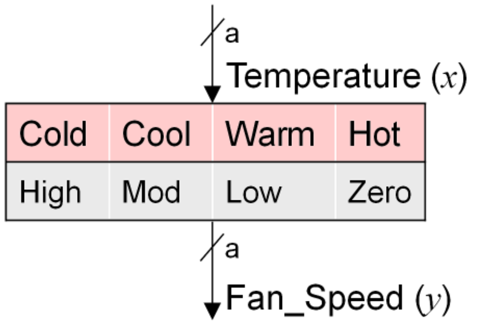
If $P_{21}(x_1)$ **and** $P_{22}(x_2)$ **then** $f_2(x_1, x_2)$



Be aware: different notation



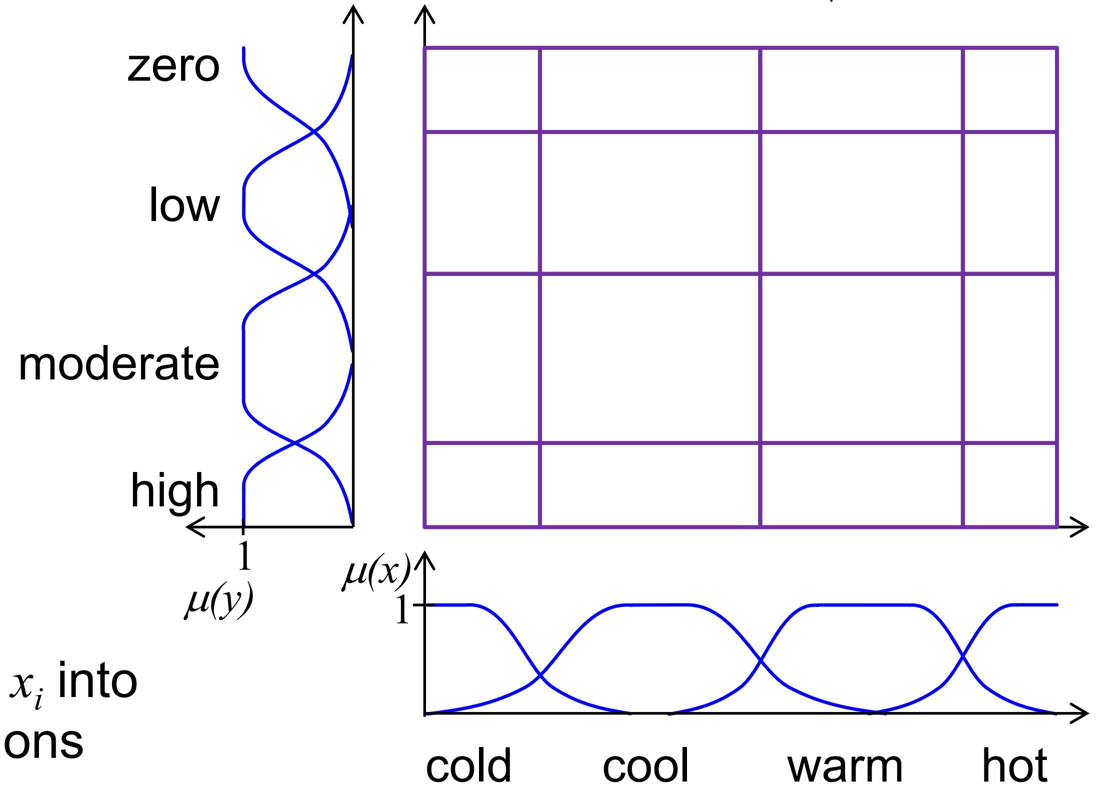
Classification: Fuzzy Rule Extraction from Data


a
Temperature (x)
a
Fan_Speed (y)

Cold	Cool	Warm	Hot
High	Mod	Low	Zero

- Prediction and classification tasks:
  - Need to derive fuzzy models from data

- *Mamdani* model:
  - Assume input granulation is fixed
    - E.g. four linguistic values
  - Granulate the input and output space
    - Divide each variable x_i into n_i membership functions



Classification: Fuzzy Rule Extraction from Data

- *Mamdani* model (cont.):
  - Analyze the entire data set in the granulated space
  - Generate fuzzy rules from given data

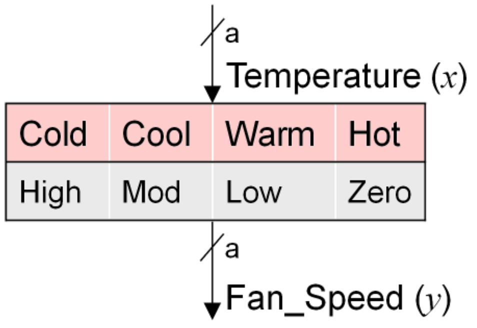
R_1: IF x is **cold**
THEN y is **moderate**

R_2: IF x is **cool**
THEN y is **moderate**

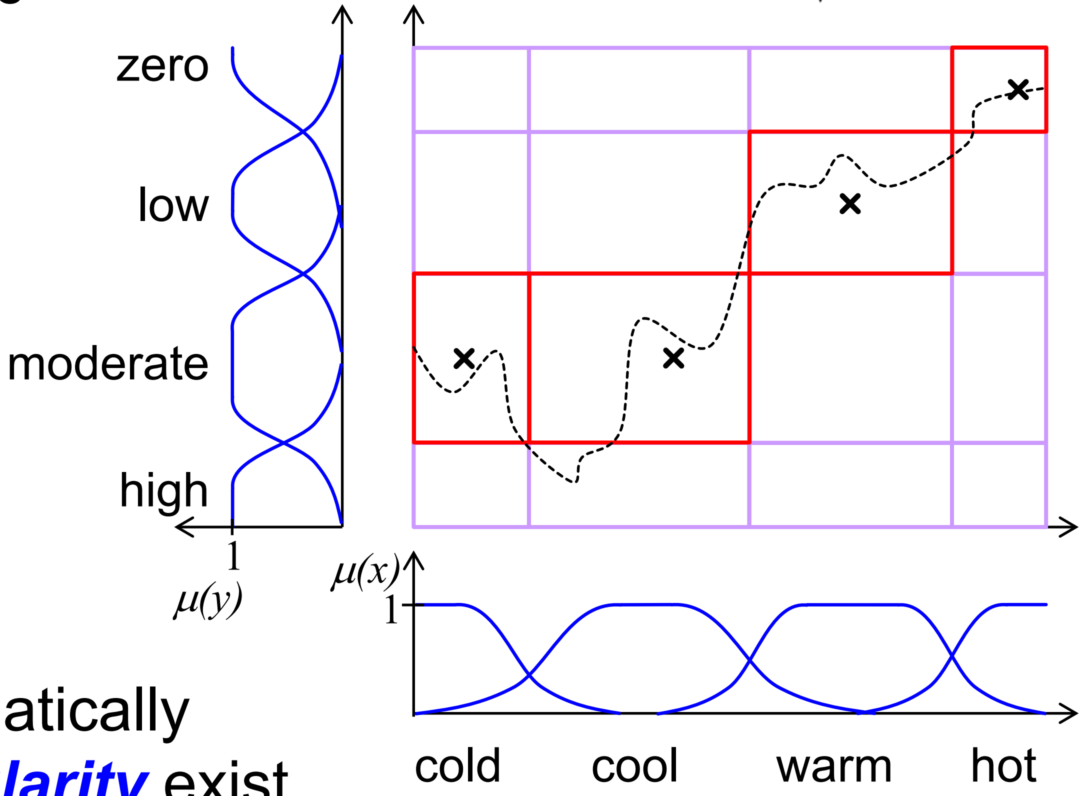
R_3: IF x is **warm**
THEN y is **low**

R_4: IF x is **hot**
THEN y is **zero**

- Other models for automatically ***determining*** the ***granularity*** exist

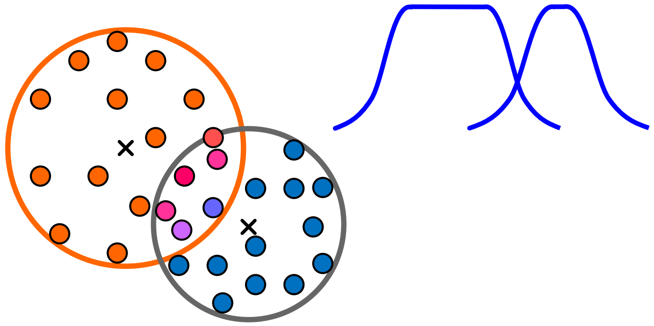


Cold	Cool	Warm	Hot
High	Mod	Low	Zero



Clustering: Fuzzy Set and Fuzzy Cluster

- Some tasks may ***need*** for fuzzy or ***soft cluster*** assignment
  - **Example**: A *computer game* could belong to both *entertainment* and *software*



- Methods: fuzzy clusters and probabilistic model-based clusters
- Fuzzy cluster: A fuzzy set *S: F*$_S$: *X* → [0, 1] (value between 0 and 1)
  - **Example**: Popularity of digital cameras, defined as a fuzzy mapping:

$$\text{popularity}(o) = \begin{cases} 1 & \text{if } i \geq 1000 \text{ of } o \text{ are sold} \\ i/1000 & \text{if } i < 1000 \text{ of } o \text{ are sold} \end{cases}$$

  - Fuzzy Set:

$$DG = \{A(0.05), B(1.0), C(0.86), D(0.27)\}$$

Cam.	Sales
A	50
B	1320
C	860
D	270

Clustering: Fuzzy Clusters and Cluster Quality

- ***Fuzzy clustering*** of k fuzzy clusters can be represented as partition matrix $M = [w_{ij}]$
  - **Example**: Clustering of "reviews" in: C$_1$:"digital camera" C$_2$: "computer"

$$M = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ \frac{2}{3} & \frac{1}{3} \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Rev.-ID	Keywords
R1	digital camera, lens
R2	digital camera
R3	lens
R4	digital camera, lens, computer
R5	computer, CPU
R6	computer, computer game

- Measure how well a clustering fits the data:

$$SSE(C) = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij}^{p} \cdot dist(o_i, c_j)^2$$

center of cluster

similarity measure

sum of squared error

for all objects o_j

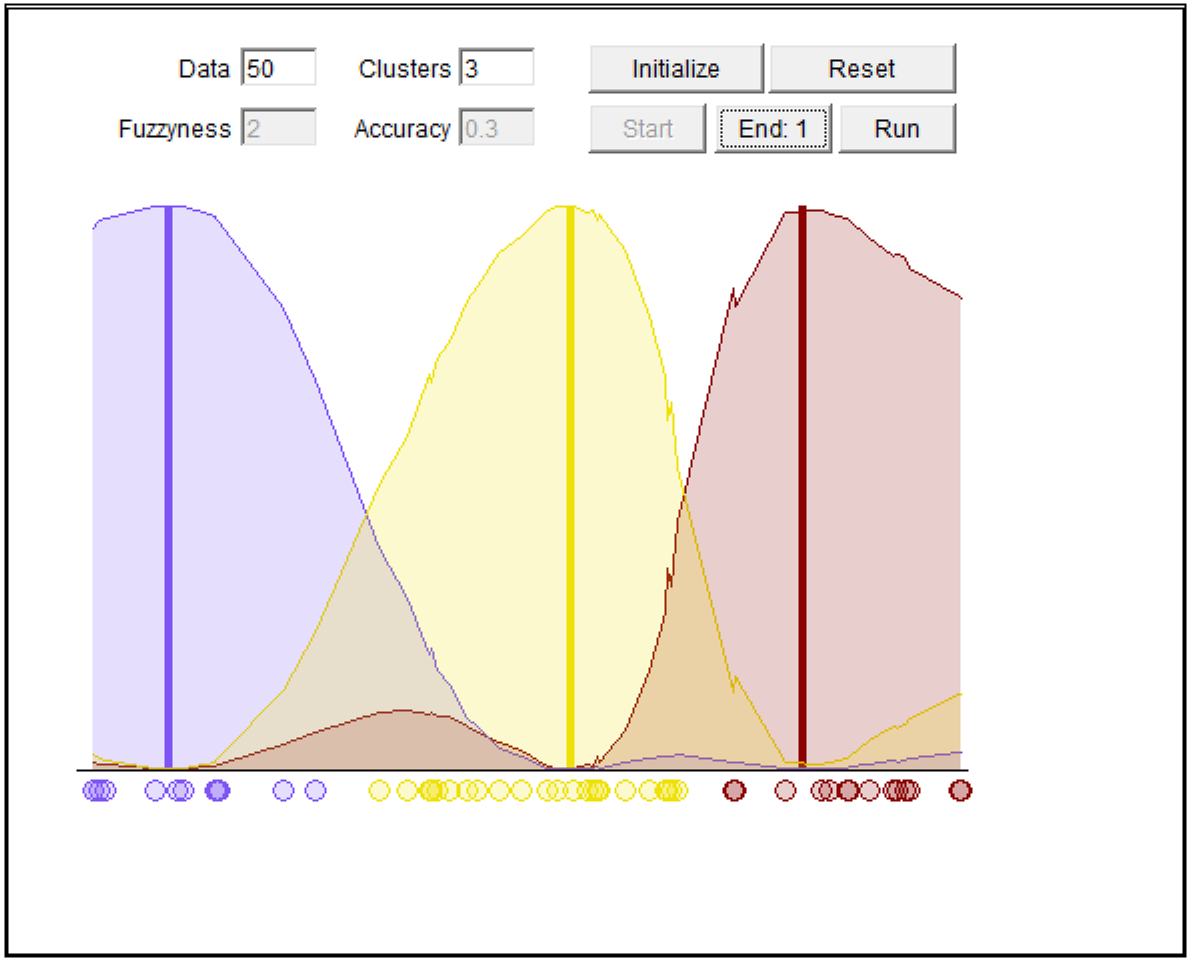
for all cluster C_j

Clustering: Probabilistic Model-Based Clusters

- Probabilistic Model-Based Clusters
  - ***Categories*** a data points belong too, are mostly ***inherently hidden*** (latent) and can not directly be observed
  - Fuzzy clusters be can defined as probability density functions
  - Set of observed objects is a mixture if instances from multiple probabilistic lusters (***mixture models***).
  - Goal: estimate the parameters of these distributions
- Fuzzy Clustering Algorithm: Expectation-Maximization
  1. Start with initial values for the parameters
  2. Calculate the cluster probabilities for each instance
  3. Re-estimate the values for the parameters
  4. Repeat

Example: Clustering with Fuzzy C-Means

http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/AppletFCM.html



Data 50 Clusters 3
Fuzzyness 2 Accuracy 0.3

Fuzzy Systems – Discussion

- Strengths
  - Less rules are required within a knowledge base
  - Membership functions can be used to represent intuitive knowledge from experts
  - Outputs terms can be familiar to humans
- Weaknesses
  - Still requires the writing of many rules
  - Knowledge acquisition and representation problems
  - Can be difficult to maintain and upgrade
  - Not adaptive in their pure form (but neural networks can extend!)
    - neuro-fuzzy systems such as ANFIS overcome this problem

Summary

- Genetic algorithms:
  - Generate-and-test approach, suitable for many problems
  - Anytime algorithms: stop at any time with a (suboptimal) solution
- Fuzzy logic:
  - Provides an alternative way to represent linguistic and subjective attributes of the real world in computing
  - Can be applied e.g. to improve efficiency and simplicity of the design process in control systems or various clustering tasks
- Further reading
  - Eiben, A.E., Smith, J.E., *Introduction to Evolutionary Computing*, Springer, 2003
  - http://www.mathworks.com/help/toolbox/fuzzy/fp351dup8.html (Practical example)