VO Natural Computation

Helmut A. Mayer

Department of Computer Sciences
University of Salzburg

SS14

Natural Computation

- Natural Computation *is a branch of computer science simulating the representation, processing and evolution of information in biological systems on computers in order to solve complex problems in science, business and engineering.*
- ABSTRACT concepts from biology to computer science
- Methods and techniques are NOT limited by biology
- Applications are NOT limited to biology
- Biological sources of concepts

Natural Computation Models

- Evolution of Genotype → Adaptation of Phenotype
- Neural Information Processing as Computational Model
- Immune System, Swarms, Ants, Evolutionary Robotics, Fuzzy Reasoning, and much more ...
- Literature
  - Kevin Kelly, *The New Biology of Machines*, 1994
  - Richard Dawkins, *The Blind Watchmaker*, 1996
  - Gerald Edelman and Giulio Tononi, *Consciousness*, 2001

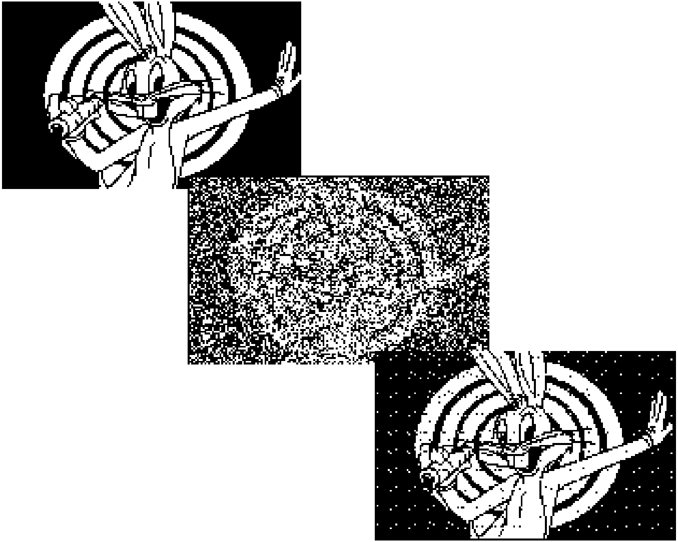
Overview EC

- Evolutionary Computation
- A glimpse at nature
- Global optimization
- Evolution Strategies, Genetic Algorithms, Evolutionary Programming, Genetic Programming

Overview ANN

- Artificial Neural Networks
  - Biological Neural Networks
  - ANN history: Hodgkin–Huxley, McCulloch–Pitts, Perceptron, Adaline
  - Multi–Layer Perceptrons
  - ANN Training, Back–propagation
  - Kohonen's Self Organizing Map
  - Recurrent Networks, Hopfield

Hopfield Image Memory I



Hopfield Image Memory II

Molecular Genetics

- DNA – *D*eoxyribo*N*uclein*A*cid Molecules, the codebook of life
  Watson & Crick ∼1960
- Adenosine (A), Thymidine (T), Cytidine (C), Guanosine (G)
- Basic Organisms
  - *Prokaryotes* – viruses, bacteria, and blue–green algae
    no discrete nucleus, no noncoding segments
  - *Eukaryotes* – plants, animals, discrete nucleus
    subcellular compartments, noncoding segments
- Flow of Information = DNA → mRNA (Uracil for Thymidine)
  → Ribosome, tRNA → Amino Acids → Protein
- Ribosome – Triplets, $4^3 = 64$, but only 20 amino acids!
- Reading Frames, “Wobble Bases”

Amino Acid Codons

First Base	Second Base				Third Base
	U	C	A	G	
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met START	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val(Met)	Ala	Glu	Gly	G

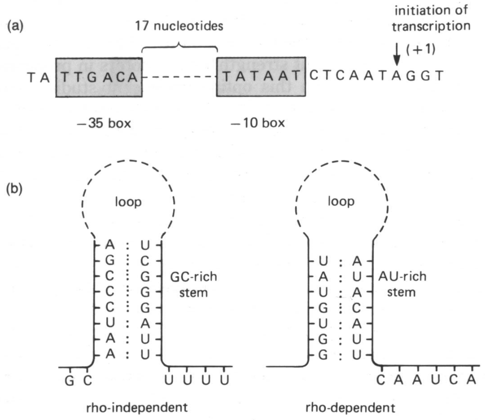
DNA Replication

- Enzyme splits duplex, Helicase unwinds, DNA Polymerase
- Replication fork moves with 800bp/s in E. Coli
  $\rightarrow$ duplication in 40 minutes
- Eukaryotes 50bp/s, Mammalians $\sim 5 \times 10^9$bp/s $\rightarrow$ 1.5 years for duplication $\rightarrow$ 90,000 replicons, massively parallel process
- Error rate $= 10^{-6}$ for replication, 10^{-9} after proof reading

DNA Transcription

- Promoter/Terminator sequences + RNA Polymerase
  $\rightarrow$ hnRNA (heterogenous nuclein) $\rightarrow$ spliced to mRNA
- Gene expression regulated by repressor/activator proteins
- Splicing = cutting out Introns, Alternative Splicing, Exon Shuffling
- Introns Early vs. Introns Late Hypothesis
- E. Coli (4.6×10^6 bp $\rightarrow$ 3,000 proteins)
  Mammals (5×10^9 bp $\rightarrow$ 30,000 proteins)
  $\rightarrow$ noncoding segments

Promoters and Terminators



Coding DNA Content

Species	Genome Size (bp)	Protein Coding DNA (%)
Escherichia Coli	4.5×10^6	~ 100
Drosophila melanogaster	1.5×10^8	33
Homo sapiens	4.0×10^9	$9 - 27$
Protopterus aethiopicus	1.42×10^{11}	$0.4 - 1.2$
Fritillaria assyriaca	1.27×10^{11}	0.02

- Chromosomes – wrapped on coils (Histones), cell specific genes are unpacked
- Example: information content of human DNA?

Natural Evolution

- Lamarck (1744 – 1829): *Inheritance of Acquired Traits*
- Darwin ($\sim$ 1860): *Survival of the Fittest*
- Selection due to limited resources, adaptation, niches
- Baldwin (1896): Phenotypical learning influences evolution of genotype (*Baldwin Effect*)
- Coevolution: *"The evolution of a species is inseparable from the evolution of its environment. The two processes are tightly coupled as a single indivisible process."* Lovelock 1988
- Central Dogma of Molecular Biology: Information only from DNA to Protein

Evolution Prerequisites

- Eigen (1970): Conditions for Darwinian Selection
- Metabolism, Self–Reproduction, and Mutation
- Mutation: new information
- Small error rates: slow progress
  Large error rates: destroys information
- "Optimal" evolution: error rate just below destruction
- Literature
  - Charles Darwin, *On the Origin of Species*, 1859
  - John Maynard Smith, *Evolutionary Genetics*, 1989

Optimization Definitions

- $f : M \subseteq \mathbb{R}^n \to \mathbb{R}, M \neq \{\}, \vec{x}^* \in M$
- $f^* := f(\vec{x}^*) > -\infty$ is a **global** minimum iff $\forall \vec{x} \in M : f(\vec{x}^*) \leq f(\vec{x})$ where
  - $\vec{x}^* \ldots$ global minimum point
  - $f \ldots$ objective function
  - $M \ldots$ feasible region
- $max\{f(\vec{x}) \mid \vec{x} \in M\} = -min\{-f(\vec{x}) \mid \vec{x} \in M\} \ldots$ global maximum
- Constraints, Feasible Region
- $M := \{\vec{x} \in \mathbb{R}^n \mid g_i(\vec{x}) \geq 0 \; \forall i \in \{1 \ldots q\}\}, \; g_i : \mathbb{R}^n \to \mathbb{R}$
  - Satisfied constraint $\Leftrightarrow g_j(\vec{x}) \geq 0$
  - Active constraint $\Leftrightarrow g_j(\vec{x}) = 0$
  - Inactive constraint $\Leftrightarrow g_j(\vec{x}) > 0$
  - Violated constraint $\Leftrightarrow g_j(\vec{x}) < 0$

TSP Problem

- Optimization problem example: Travelling Sales Person
- NP–complete, combinatorial, multimodal, CP–easy (Cocktail Party easy;) D. Goldberg)
- $C = \{c_1 \ldots c_n\} \ldots$ cities
- $\rho_{ij} = \rho(c_i, c_j) \quad i, j \in \{1 \ldots n\},\ \rho_{ii} = 0\ \ldots$ cost
- $\Pi \in S_n = \{s : \{1 \ldots n\} \to \{1 \ldots n\}\}\ \ldots$ feasible tour
- $f(\Pi) = \sum_{i=1}^{n-1} \rho_{\Pi(i),\Pi(i+1)} + \rho_{\Pi(n),\Pi(1)}\ \ldots$ objective function

Optimization Precautions

- In global optimization no general criterion for identification of the global optimum exists (Törn and Žilinskas 1990)
- No Free Lunch Theorem (NFL) *"…all algorithms that search for an extremum of a cost function perform exactly the same, according to any performance measure, when averaged over all possible cost functions."* (Wolpert and Macready 1996)

Evolutionary Computation

- Model of genetic inheritance and Darwinian strife for survival
- Evolutionary Algorithms (EAs) ("History")
  - Genetic Algorithms (GAs)
  - Evolution Strategies (ESs)
  - Evolutionary Programming (EP)
  - Genetic Programming (GP)
- EAs : directional search, no random search!

Application Criteria

- EAs are generic but not universal
- Optimization (broad class of problems)
- "Complex" problems (no conventional algorithm available)
- Features of candidate problems
  - NP–complete problems
  - High–dimensional search space
  - Non–differentiable surfaces (general absence of gradients)
  - Complex and noisy surfaces
  - Deceptive surfaces
  - Multimodal surfaces

Basic EA Components

- Problem Encoding, genotype, chromosome (bitstring, real–valued vector, tree, decoder, ...)
- Population of individuals (generation gap)
- Selection Scheme
- Genetic Operators (mutation, recombination, inversion, ...)
- Fitness Function

Basic EA Pseudo–Code

A Basic Evolutionary Algorithm

```
BEGIN
generate initial population
WHILE NOT terminationCriterion DO
    FOR populationSize
        compute fitness of each individual
        select individuals
        alter individuals
    ENDFOR
ENDWHILE

END
```

Evolution Strategies Basics

- Bienert, Rechenberg, Schwefel: TU Berlin (1964)
- Chromosome: real valued vector
- Specific genetic operators and selection schemes
- Self–adaptation of mutation rate
- First experiments: hydrodynamical problems (shape optimization of a bent pipe)

Early Evolution Strategies

- Simple $(1+1)$–ES (population?)
- Object and strategy parameters
- Heuristic self–adaptation with $\frac{1}{5}$ success rule

$$\sigma(t) = \begin{cases} \sigma(t-1)c & \text{if } p > \frac{1}{5} \\ \sigma(t-1)/c & \text{if } p < \frac{1}{5} \\ \sigma(t-1) & \text{if } p = \frac{1}{5} \end{cases}$$

  with $c = \sqrt[n]{0.85}$
- Multi–membered ES, $(\mu + \lambda)$, (μ, λ) selection
- Covariances (rotation angles)

Self–adaptation

- n–dimensional normal distribution $p(\vec{z}) = \frac{e^{-\vec{z}^T C^{-1} \vec{z}}}{\sqrt{(2\pi)^n |C|}}$
- Mutation of standard deviations σ
  $\sigma_i' = \sigma_i e^{\tau' N(0,1) + \tau N_i(0,1)}$ with $\tau' \cong \frac{1}{\sqrt{2n}}$ and $\tau \cong \frac{1}{\sqrt{2\sqrt{n}}}$
- Mutation of rotation angles α
  $\alpha_j' = \alpha_j + \beta N_j(0,1)$ with $\beta \cong 0.0837 \cong 5^o$
- Mutation of object parameters x_i
  $x_i' = x + \vec{N}(\vec{0}, C(\sigma_i, \alpha_j))$

Recombination

- Recombination, discrete, intermediate, local, global

$$x_i' = \begin{cases} x_{S,i} \\ x_{S,i} \vee x_{T,i} \\ x_{S,i} + u(x_{T,i} - x_{S,i}) \\ x_{Si,i} \vee x_{Ti,i} \\ x_{Si,i} + u_i(x_{Ti,i} - x_{Si,i}) \end{cases}$$

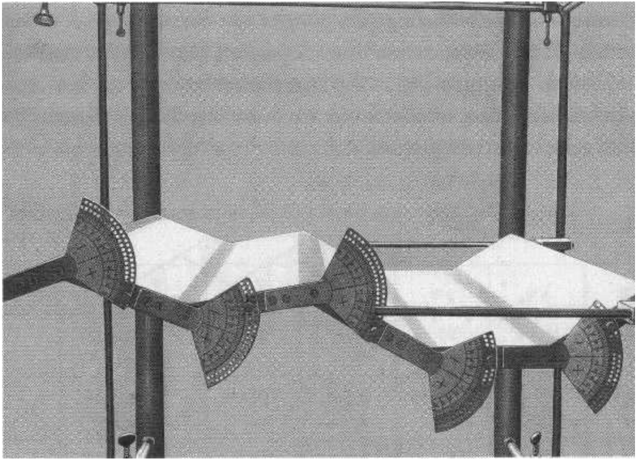
  S, T random parent individuals

- Empiric: object parameters (discrete recombination) strategy parameters (intermediate recombination)

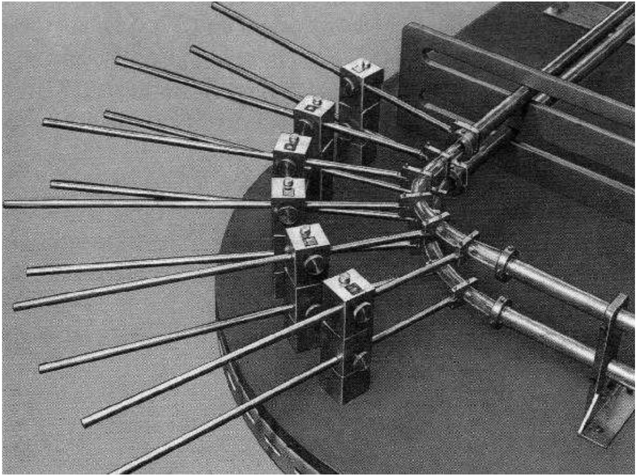
ES History

- First experiments 1964, TU Berlin
- Pictures from: Ingo Rechenberg, *Evolutionsstrategie '94* (1994)
- Further literature
  - Hans–Paul Schwefel, *Evolution and Optimum Seeking*, (1995)
  - Thomas Bäck, *Evolutionary Algorithms in Theory and Practice*, (1996)

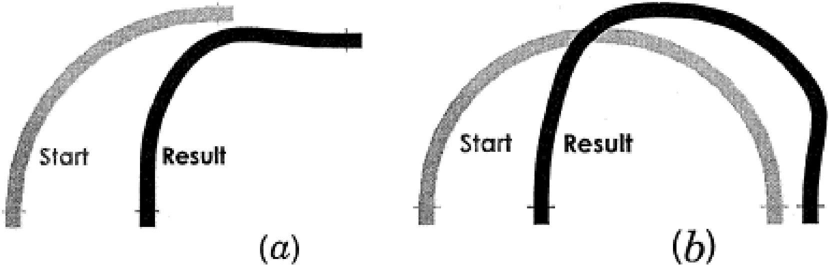
Plate Resistance



Pipe Resistance Experiment



Minimal Resistance



Previously unknown form explored (1965)!

Evolutionary Programming Basics

- No recombination!
- Additional mutation κ when mapping genotype to phenotype
- Genotype (object parameters) mutation
  $x_i' = x_i + N_i(0,1)\sqrt{\beta_i f(\vec{x}) + \gamma_i}$ (standard $\beta_i = 1$ and $\gamma_i = 0$)
- Meta–EP, self–adaptation
- EP selection: every individual scores on q randomly chosen competitors, EP selection $\rightarrow$ $(\mu + \mu)$ ES selection
- Literature
  - L. J. Fogel and A. J. Owens and M. J. Walsh, *Artificial Intelligence through Simulated Evolution* (1966)
  - D. B. Fogel, *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling*, (1991)

Genetic Algorithm Basics

- John Holland, Ann Arbor, Michigan (1960s)
- Chromosome: fixed length bit strings
  $b_i \in \{0, 1\} \qquad i \in \{1, \ldots l\}$
- One–point crossover and bit–flip mutation

```
A Basic Genetic Algorithm

BEGIN
generate initial population
WHILE NOT terminationCriterion DO
  FOR populationSize
    compute fitness of each individual
    record overall best individual
    select individuals to mating pool
    recombine and mutate individuals
  ENDFOR
  new generation replaces old
ENDWHILE
output overall best individual
END
```

- Encoding of solutions, e.g., real values
  $x = c + \frac{d-c}{2^l-1} \sum_{i=1}^{l} b_i 2^{i-1}$, genotype $I = \{b_1, \ldots b_l\}$, phenotype $x \in [c, d]$

Genetic Operators

- Crossover is main GA operator(?)
- k–point, uniform, problem specific, crossover probability $p_c \cong [0.6 - 0.8]$
- Mutation background operator(?)
  - Reintroduction of lost material
  - Mutation probability $p_m \cong [0.01 - 0.001]$
  - Theory $p_m = \frac{1}{l}$, analogue to nature(!)

Fitness Function Design

- Keep it simple, no artificial measures
- Invalid solutions, refine operators or use penalty
- Penalty Functions
  - "Death Penalty"
  - Fixed Penalty
  - Dynamic Penalty
  - Measure of obstruction (no simple count of obstructions)

Knapsack Problem

- Encoding example
- Set of items $g_i, i = 1, \ldots, n$
- Each item has a price p_i and a weight w_i
- Select items (put into knapsack)
  $\{g_k | \sum_{k=1}^{m \leq n} p_k \rightarrow max \wedge \sum_{k=1}^{m \leq n} w_k \leq w_{max}\}$
- Encoding? Fitness Function?
- Incorporate problem knowledge, usually GA + Heuristics > GA (prevents however unconventional solutions)

Selection Methods

- Fitness proportionate selection
  - Roulette Wheel selection (sampling methods)
    $p_{s,i} = \frac{f_i}{\sum_{i=1}^{n} f_i}, \quad i \in \{1, \ldots, N\}$
  - Scaling methods, e.g., Sigma Scaling
    $f_{base} = \overline{f} - g\sigma_f, \quad g \in \mathbb{R}$
    $f_i' = f_i - f_{base}$
- Rank based selection
  - Linear Ranking, Exponential Ranking
    $p_{s,i} = f(r), \quad r \in \{1, \ldots, n\}$
  - Tournament Selection
    tournament size $\Leftrightarrow$ selection pressure
  - Truncation Selection

GA Analysis Definitions

- Schemata, e.g. $1*******$ and $*01****1$ (length $l = 8$)
- Schema Order
  $\sigma = \mid \{i \mid b_i \in \{0,1\} \mid \quad \sigma_1 = 1, \sigma_2 = 3$
- Defining Length
  $\delta = max\{i \mid b_i \in \{0,1\}\} - min\{i \mid b_i \in \{0,1\}\}, \delta_1 = 0, \delta_2 = 6$

Schema Fitness

- Average Schema Fitness $\overline{f}(H^t) = \frac{1}{m(H^t)} \sum_{x_i \in H^t} f(x_i)$
  $m(H^t) \ldots \#$ of specific schema (hyperplane) in generation t
  $f(x_i) \ldots$ fitness of individual x_i
- Average Fitness $\overline{f}^t = \frac{\sum_{i=1}^{n} f(x_i)}{n}$
- Selection probability (proportional selection)
  $p_s(x_i) = \frac{f(x_i)}{\sum_{i=1}^{n} f(x_i)}$
- Combining $\Longrightarrow$
  $\frac{\overline{f}(H^t)}{\overline{f}^t} = \frac{n \sum_{x_i \in H^t} p_s(x_i)}{m(H^t)} = \frac{m(H^{t+1})}{m(H^t)}$
  $\frac{\overline{f}(H^t)}{\overline{f}^t} > 1 \to m(H^{t+1}) = m(H^t)(1+c)$
  $m(H^t) = m(H^0)(1+c)^t$ with $c > 0$ (exponential increase)

Schema Theorem

- Schema survival probability under 1–point crossover
  $1 - p_c \frac{\delta(H^t)}{l-1}$
- Schema survival probability under mutation
  $(1 - p_m)^{\sigma(H^t)}$
- Fundamental Theorem of GAs
  $m(H^{t+1}) \geq m(H^t) \frac{f(H^t)}{\bar{f}^t} (1 - p_c \frac{\delta(H^t)}{l-1})(1 - p_m)^{\sigma(H^t)}$
- ST flaws: finite population sizes, generalization of single generation transition, proportional selection...
- Building Block Hypothesis: *Short, low–order, and highly fit schemata are sampled, recombined, and resampled to form strings of potentially higher fitness.* (Goldberg, 1989)
- Implicit Parallelism: $\mathcal{O}(n^3)$ schemata are processed "simultaneously"

K–Armed Bandit

- Exploration–Exploitation model
- $k = 2$, N trials, $2n$ exploration trials, payoffs $(\mu_1, \sigma_1), (\mu_2, \sigma_2)$
  Expected loss $L(N, n) = |\mu_1 - \mu_2|[(N - n)q(n) + n(1 - q(n))]$
  $q(n)$ ... probability that worse arm is observed best arm
- Minimizing $L \rightarrow n^*$ (optimal experiment size)
  $N \sim e^{n^*}$, exponentially increase trials to the oberved best arm
- GA analogue
  - k schemata with '*' at identical loci $\leftrightarrow$ k–armed bandit
  - all schemata $\leftrightarrow$ multiple k–armed bandit problem
  - optimal strategy $\leftrightarrow$ Schema Theorem

Literature

- John Holland, *Adaptation in Natural and Artificial Systems* (1975)
- David Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning* (1989)
- Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (1992)
- Melanie Mitchell, *An Introduction to Genetic Algorithms* (1996)

Genetic Programming Basics

- John Koza, Stanford (1988)
- Evolution of hierarchical computer programs
- Why use LISP?
  - Programs and data are S–expressions
  - LISP program is its own parse tree
  - EVAL function starts program
  - Dynamic storage allocation and garbage collection
- GP today: Assembler, C, Java

GP Design I

- Function set F and terminal set T, e.g.,
  $F = \{AND, OR, NOT\}$
  $T = \{D_0, D_1\}$
  $C = F \cup T$
- Closure of C, protected functions, e.g.,
  ```
  (defun srt(argument)
     ``The Protected Square Root Funtion''
     (sqrt (abs argument)))
  ```
- Sufficiency of C, problem knowledge
- Initial structures, maximum tree depth
  full, grow, ramped half–and–half

GP Design II

- Fitness function
  - Raw fitness r
    $r(i,t) = \sum_{j=1}^{N} |S(i,j) - C(j)|$
    $N \ldots$ fitness cases
    $S(i,j) \ldots$ program value
    $C(j) \ldots$ correct value
  - Standardized fitness s
    $s(i,t) = r(i,t)$ or $s(i,t) = r_{max} - r(i,t)$
  - Adjusted fitness a
    $a(i,t) = \frac{1}{1+s(i,t)}$
  - Normalized fitness n
    $n(i,t) = \frac{a(i,t)}{\sum_{k=1}^{n} a(k,t)}$

GP Design III

- GP population sizes (1,000 - 10,000)
- Proportionate selection
- Primary GP operator is crossover
- Secondary GP operators
  - Mutation, random subtree insertion
  - Permutation (of arguments)
  - Editing, e.g., `(NOT (NOT X))` $\rightarrow$ `X`
  - Encapsulation, "freezing of subtrees"
  - Decimation (esp. initial population)
- Literature
  - John R. Koza, On the Programming of Computers by means of Artificial Intelligence (1992)

GP Example

- Daida et al., *Extracting Curvilinear Features from Synthetic Aperture Radar Images of Arctic Ice: Algorithm Discovery Using the Genetic Programming Paradigm*, IGARS 1995
  - Classification of Radar Images
  - Sea Ice Analysis, Pressure Ridges
  - Evolved code generates good results but...

GP Example Sea Ice

Table 2. Best-of-Run Individual

(÷ (+ *Laplacian-of-Mean Mean*$_{3\times3}$) (÷ (+ 1.9492742 *Mean*$_{3\times3}$) (× (– (× (÷ (≤ *Mean*$_{5\times5}$ -4.42039 (– (–*Mean*$_{3\times3}$ *Mean*$_{3\times3}$) (≤ (+ *Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (≤ *Image Mean*$_{5\times5}$ *Image Mean*$_{5\times5}$) (÷*Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (÷*Laplacian-of-Mean Mean*$_{5\times5}$))) *Mean*$_{5\times5}$) (÷ *Mean*$_{3\times3}$ -4.87104)) (÷ (÷ -4.721524 *Image*) (÷*Mean*$_{5\times5}$ 3.8854232))) (÷ (– (÷ (–*Mean*$_{3\times3}$ *Laplacian*$_{5\times5}$) (× *Laplacian*$_{5\times5}$ *Laplacian*$_{5\times5}$)) (≤ (+*Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (– (× (× *Laplacian-of-Mean Mean*$_{3\times3}$) (– (÷ (–*Mean*$_{3\times3}$ *Laplacian*$_{5\times5}$) (× *Laplacian*$_{5\times5}$ *Laplacian*$_{5\times5}$)) (≤ (+*Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (≤ *Image Mean*$_{5\times5}$ *Image Mean*$_{5\times5}$) (÷*Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (÷ *Laplacian-of-Mean Mean*$_{5\times5}$)))) *Laplacian*$_{5\times5}$) (÷ *Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (– (÷ (–*Mean*$_{3\times3}$ *Laplacian*$_{5\times5}$) (× *Laplacian*$_{5\times5}$ *Laplacian*$_{5\times5}$)) (≤ (+*Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (≤ *Image Mean*$_{5\times5}$ *Image Mean*$_{5\times5}$) (÷*Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (÷ *Laplacian-of-Mean Mean*$_{5\times5}$))))) (÷ (– (– (× (– (× (÷ (≤ *Mean*$_{5\times5}$ 4.42039 (≤ (+ 1.9492742 *Mean*$_{3\times3}$) (× (≤ *Image Image Laplacian*$_{5\times5}$ *Image*) (+ *Laplacian-of-Mean Laplacian-of-Mean*)) (+ *Image Mean*$_{3\times3}$) (× *Mean*$_{5\times5}$ *Mean*$_{5\times5}$)) *Mean*$_{5\times5}$) (÷ *Mean*$_{3\times3}$ -4.87104)) (÷ (÷ -4.721524 *Image*) (÷*Mean*$_{5\times5}$ 3.8854232))) (÷ (– (– (× -1.2301508 0.2565225) (≤ (+ 3.6199708 *Mean*$_{3\times3}$) (÷ *Image Image*) (÷ -2.134516 *Image*) (+ *Laplacian-of-Mean Image*))) (≤ (+*Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (– *Laplacian*$_{5\times5}$ *Laplacian*$_{5\times5}$) (÷*Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (– (÷ (–*Mean*$_{3\times3}$ *Laplacian*$_{5\times5}$) (÷ *Laplacian-of-Mean* 3.8854232)) (≤ (+*Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (≤ *Image Mean*$_{5\times5}$ *Image Mean*$_{5\times5}$) (÷*Laplacian*$_{5\times5}$ *Mean*$_{3\times3}$) (÷ *Laplacian-of-Mean Mean*$_{5\times5}$))))) (÷ (– (÷*Mean*$_{5\times5}$ 3.8854232) (÷ *Laplacian-of-Mean Mean*$_{3\times3}$)) (× (≤ 2.3869586 *Laplacian-of-Mean* (≤ *Mean*$_{3\times3}$ *Image* -3.3760195 *Laplacian*$_{5\times5}$) *Laplacian*$_{5\times5}$) (÷ -2.9275842 *Image*))))) (– *Laplacian*$_{5\times5}$ *Image*)) *Mean*$_{5\times5}$) (÷ *Laplacian-of-Mean Mean*$_{3\times3}$)) (× (≤ 2.3869586 *Laplacian-of-Mean* (≤ *Mean*$_{3\times3}$ *Image* -3.3760195 *Laplacian*$_{5\times5}$) *Laplacian*$_{5\times5}$) (÷ -2.9275842 *Image*))))) (–*Laplacian*$_{5\times5}$ *Image*))))

(a)

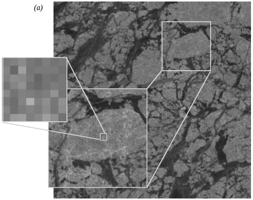
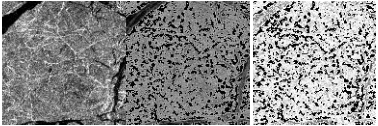


Figure 1. (a) Pressure ridges often appear as low-contrast curvilinear features in low-resolution SAR imagery. (b) 128 × 128 subimage from April 23, 1992, ERS-1 ©ESA 1992. Although contrast-enhanced, the figure still does not show all of the pressure ridge features that can be detected by eye. (c) Solution from best-of-run individual (with image overlay.) Areas where there may be a ridge are darkened. (d) Solution (only) from best-of-run individual.

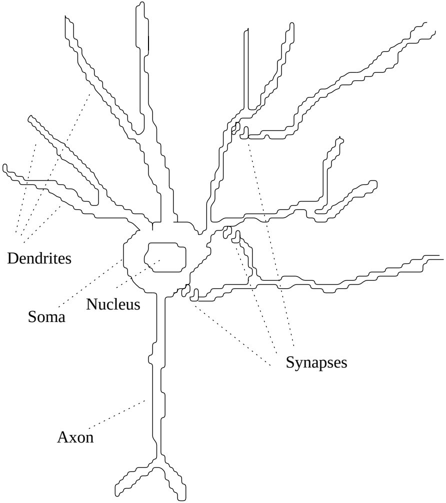


(b) *(c)* *(d)*

Biological Neurons

- Nervous System: Control by Communication
- Massive Parallelism, Redundancy, Stochastic “Devices”
- Humans: 10^{10} neurons, 10^{14} connections (conservative estimation), $10^{1,000,000}$ possible networks(!)
- Neurons: cell body (soma), axon, synapses, dendrites
- Membrane Potential of $-70mV$
  sodium (Na^+) and potassium (Ka^+) ions (chloride Cl^- ions)
- Sodium pump constantly expells Na^+ ions
- Complex interaction of membrane, concentration, and electrical potential

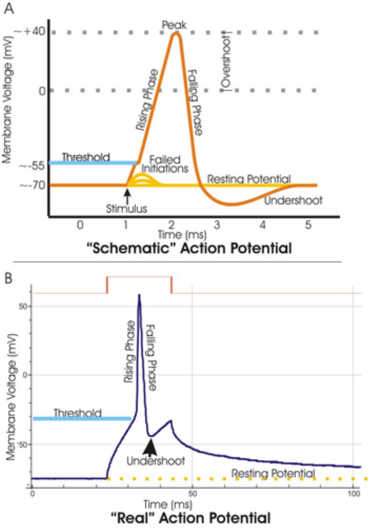
Schematic Neuron



Electrical Signals

- Potassium is in equilibrium, sodium NOT
- Sodium conductance is a function of the membrane potential
- Above *threshold* sodium conductance increases, ion channels, depolarization, polarization, refractory period = action potential
- Frequency coding, is (all) information encoded in action potentials?
- Hodgkin/Huxley–Model

Action Potential (from en.wikipedia.org)



Signal Transmission

- Action potential triggers adjacent depolarization of membrane
  $\rightarrow$ no attenuation
- Speed of action potential $\sim \sqrt{axondiameter}$
  Crab 30 $\mu m - 5\ m/s$
  Squid 500 $\mu m - 20\ m/s$
  Human 20 $\mu m - 120\ m/s$
- *Myelin* insulates membrane, nodes of *Ranvier*
  action potential “jumps” from node to node

Synapses I

- Electrochemical processes, neurotransmitters
- Connect axon–dendrites (but also axon–axon, dendrites–dendrites, synapses–synapses)
- Spatio–temporal integration of action potentials
- Excitatory and inhibitory potentials
  postsynaptic duration $\sim 5\ ms$
- Neurotransmitters influence threshold (permanent changes = learning = closing/opening of ion channels)

Synapses II

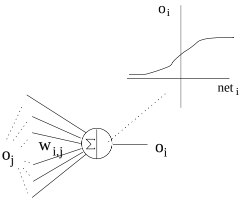
- Slow Potential Theory: Spike frequency codes potential, Vf–converter
- Noise: ionic channels, synaptic vesicles (store neurotransmitter), postsynaptic frequency estimation (in $\sim$100ms a frequency range from 1–100Hz)
- BNN great variety of synapses, ANN mostly one type of "synapse"

Neuron Models

- Level of Simplification?
- *McCulloch* and *Pitts* (1943), Logic Model
  Binary signals, no weights, simple (nonbinary) threshold
  Excitatory and inhibitory connections (absolute, relative)
  Addition of weights
- *Rosenblatt* (1958), Perceptron
  Real–valued weights and threshold

Generic Neuron Model

- Generic Connectionist Neuron
  Nonlinear activation (transfer) function
  Widely used in todays ANNs



- Next generation: spiking neurons, hardware neurons, biological hardware

Networks

- Mysticism of **Neural** Networks
- Functional Model: $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$
  node structure, connectivity, learn algorithm
- "Black Box Syndrom", unexplicable ANN decisions
- Basic Structures: Feed–Forward (MLPs, Kohonen),
  $f = f(g(x))$
  Recurrent (Recurrent MLPs, Hopfield),
  $f = f(x_t, f(x_{t-1}), f(x_{t-2}), \ldots)$

ANN Training

- ANN Training (Learning, Teaching): Adjustment of network parameters
- General Training Methods
  - Supervised Learning (Teacher, I/O–Patterns)
  - Reinforcement Learning (Teacher, Learn Signal)
  - Unsupervised Learning (No Teacher, Self–Organization)

ANN Application Domains

- Constraint Satisfaction (Scheduling, *n*–Queens)
- Content Addressable Memory (Image Retrieval)
- Control (Machines, "ANN Driver")
- Data Compression
- Diagnostics (Medicine, Production)
- Forecasting (Financial Markets, Weather)
- General Mapping (Function Approximation)
- Multi Sensor Data Fusion (Remote Sensing)
- Optimization
- Pattern Recognition (Voice, Image)
- Risk Assessment (Credit Card)

Perceptron

- *Rosenblatt*: Perceptron = Retina + A(ssociation) Layer + R(esponse) Layer, Retina $\rightarrow$ A (partial connections), A $\leftrightarrow$ R (recurrent connections), Threshold Logic Unit (TLU)
- Simplified Perceptron is easier to analyze
- Weight and input vectors, scalar product, threshold as weight
- Linear Separability
  Two sets of points A and B in an n–dimensional space are **linearly separable**, *if there exist $n+1$ real numbers $w_1, \ldots, w_{n+1}$ so that for each point $x = (x_1, \ldots, x_n) \in A$:*
  $\sum_{i=1}^{n} w_i x_i \geq w_{n+1}$
  and for each point $x = (x_1, \ldots, x_n) \in B$:
  $\sum_{i=1}^{n} w_i x_i < w_{n+1}$
- Standard Perceptron demands linearly separable problems

Perceptron Learning

- Perceptron Learn Algorithm

  Start: Random $\vec{w_0}, \ t := 0$

  Test: Random $\vec{x} \in P \cup N$

  If $\vec{x} \in P$ and $\vec{w_t}\vec{x} > 0 \Rightarrow$ **Test**
  If $\vec{x} \in N$ and $\vec{w_t}\vec{x} < 0 \Rightarrow$ **Test**
  If $\vec{x} \in P$ and $\vec{w_t}\vec{x} \leq 0 \Rightarrow$ **Add**
  If $\vec{x} \in N$ and $\vec{w_t}\vec{x} \geq 0 \Rightarrow$ **Sub**

  Add: $\vec{w_{t+1}} := \vec{w_t} + \vec{x}$
  Sub: $\vec{w_{t+1}} := \vec{w_t} - \vec{x}$

  $t := t + 1$
  Exit if no weight update $\forall \vec{x}$

- Perceptron Convergence Theorem
- "Attack" on perceptrons: M. Minsky and S. Papert *Perceptrons* (1969)
- *The XOR–function of two boolean variables* x_1, x_2 *cannot be computed with a* single *perceptron.* (Connectedness)

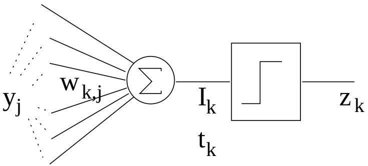
Earlier Models

- Hebbian Learning, *Donald Hebb* (1949)
  "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency as one of the cells firing B is increased."
- Generalized Hebb Rule
  $\Delta w_{i,j} = \eta \, a_i \, b_j$
- Linear Associator, input vector $\vec{a}$, output vector $\vec{b}$
  $\mathbf{W} = \eta \, \vec{b} \, \vec{a}^T$
- ADALINE (Adaptive Linear Element), *Widrow and Hoff* (1960)
  Threshold, error signal, error function has single minimum
  learning rule is special case of backpropagation

Gradient Descent

- Basic Optimization Method
- How to compute the steepest descent? $\rightarrow$ Gradient
- The Nabla Operator (3 dimensions) $\vec{\nabla} = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix}$
- Total Differential of $E(x, y, z)$: $dE = \frac{\partial E}{\partial x} dx + \frac{\partial E}{\partial y} dy + \frac{\partial E}{\partial z} dz$
- Differential Path Element $\vec{ds} = \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix}$
- Gradient $\vec{grad} E = \vec{\nabla} E$
- $dE = \vec{grad} E \ \vec{ds} \rightarrow$ maximal, if $\vec{grad} E || \vec{ds}$
- Note: $div \vec{E} = \vec{\nabla} \vec{E}$ (Divergence), $rot \vec{E} = \vec{\nabla} \times \vec{E}$ (Curl)

Widrow–Hoff Learning Rule

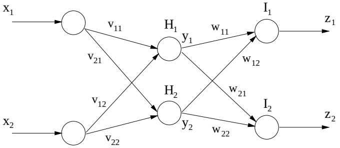


- Gradient Descent, $\Delta w_{k,j} = -\eta \frac{\partial E}{\partial w_{k,j}}$
- Error $E = \sum_{p=1}^{P} E^{(p)}$, $E^{(p)} = \sum_{k=1}^{m} (t_k^{(p)} - I_k^{(p)})^2$
- p training patterns, m output neurons, $t \ldots$ target value
- $\frac{\partial E^{(p)}}{\partial w_{k,j}} = \frac{\partial}{\partial w_{k,j}} (\sum_{k=1}^{m} (t_k^{(p)} - \sum_{j=1}^{h} w_{k,j} y_j^{(p)})^2) =$
  $= -2(t_k^{(p)} - I_k^{(p)}) y_j^{(p)}$
- Omitting pattern index p
  $\Delta w_{k,j} = \eta (t_k - I_k) y_j = \eta \delta_k y_j$

Multi–Layer Perceptron

- Learning as minimization (of network error)
- Error is a function of network parameters
- Gradient descent methods reduce error
- Problem with perceptrons with hidden layers
- Backpropagation = Iterative Local Gradient Descent
  Werbos (1974), *Rumelhart, Hinton, Williams* (1986)
- Error–Backpropagation, output error is transmitted backwards as weighted error, network weights are updated **locally**
- Weight update $\Delta w_{j,i} = \eta \delta_j a_i$
  Generalized error term δ
- Common transfer functions: differentiable, nonlinear, monotonous, easily computable differentiation

Error–Backpropagation I



- $H_j = \sum_{i=1}^{n} v_{j,i} x_i \quad I_k = \sum_{j=1}^{h} w_{k,j} y_j$
  $y_j = f(H_j), z_k = f(I_k)$
- Error $E^{(p)} = \frac{1}{2} \sum_{k=1}^{m} (t_k^{(p)} - z_k^{(p)})^2$
- Output Layer: $\Delta w_{k,j} = -\eta \frac{\partial E}{\partial w_{k,j}}$
  $\frac{\partial E}{\partial w_{k,j}} = \frac{\partial E}{\partial I_k} \frac{\partial I_k}{\partial w_{k,j}} = \frac{\partial E}{\partial I_k} y_j$
  $\frac{\partial E}{\partial I_k} = \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial I_k} = -(t_k - z_k) f'(I_k)$
  $\frac{\partial E}{\partial w_{k,j}} = -(t_k - z_k) f'(I_k) y_j$ mit $\delta_k = (t_k - z_k) f'(I_k)$
  $\Delta w_{k,j} = \eta \delta_k y_j$

Error–Backpropagation II

- Hidden Layer: $\Delta v_{j,i} = -\eta \frac{\partial E}{\partial v_{j,i}}$
  $\frac{\partial E}{\partial v_{j,i}} = \frac{\partial E}{\partial H_j} \frac{\partial H_j}{\partial v_{j,i}} = \frac{\partial E}{\partial H_j} x_i$
  $\frac{\partial E}{\partial H_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial H_j} = \frac{\partial E}{\partial y_j} f'(H_j)$
  $\frac{\partial E}{\partial y_j} = -\frac{1}{2} \sum_{k=1}^{m} \frac{\partial (t_k - f(I_k))^2}{\partial y_j} = -\sum_{k=1}^{m} (t_k - z_k) f'(I_k) w_{k,j}$
  mit $\delta_j = f'(H_j) \sum_{k=1}^{m} \delta_k w_{k,j}$
  $\Delta v_{j,i} = \eta \delta_j x_i$
- Local update rules propagating error from output to input
- Present all p patterns of the training set = 1 *Epoch* (complete training e.g., 1,000 epochs)
- Batch Learning (Off–line): accumulate weight changes for all patterns, then update weights
- On–line Learning: update weights after each pattern

Backpropagation Variants I

- Standard Backpropagation: $\vec{w_t} = \vec{w_{t-1}} - \eta \vec{\nabla} E$
- Gradient Reuse: use $\vec{\nabla} E$ as long as error drops
- BP with variable stepsize (learn rate) η
- BP with momentum: $\Delta \vec{w_t} = -\eta \vec{\nabla} E + \alpha \Delta \vec{w_{t-1}}$

Backpropagation Variants II

- Rprop (Resilient Backpropagation), *Riedmiller/Braun*, 1993

$$\Delta w_{i,j}(t) = \begin{cases} -\Delta_{i,j}(t) & \mathit{if} \frac{\partial E}{\partial w_{i,j}} > 0 \\ +\Delta_{i,j}(t) & \mathit{if} \frac{\partial E}{\partial w_{i,j}} < 0 \\ 0 & \mathit{else} \end{cases}$$

$$\Delta_{i,j}(t) = \begin{cases} \eta^{+}\Delta_{i,j}(t-1) & \mathit{if} \frac{\partial E(t-1)}{\partial w_{i,j}} \times \frac{\partial E(t)}{\partial w_{i,j}} > 0 \\ \eta^{-}\Delta_{i,j}(t-1) & \mathit{if} \frac{\partial E(t-1)}{\partial w_{i,j}} \times \frac{\partial E(t)}{\partial w_{i,j}} < 0 \end{cases}$$

$$0 < \eta^{-} < 1 < \eta^{+}$$

- Second order methods

$$E(\vec{w}) = E(\vec{w_t}) + (\vec{w} - \vec{w_t})\vec{\nabla}E(\vec{w_t}) + \frac{1}{2}(\vec{w} - \vec{w_t})H(\vec{w} - \vec{w_t})^T + \ldots$$

$$\vec{\nabla}E(\vec{w}) = \vec{\nabla}E(\vec{w_t}) + (\vec{w} - \vec{w_t})H \rightarrow \vec{\nabla}E(\vec{w}) = \vec{0}$$

Hessian Matrix H, complex computations, nonlocal informations!

Self Organizing Maps

- Cerebral Cortex: topologically ordered maps (sensory inputs)
- *Willshaw* and *von der Malsburg* (1976)
  two layers, intra–connections (short-range excitatory, long-range inhibitory), inter–connections (Hebbian learning)
- *Kohonen* (1982)
  Topological map, vector quantization, competitive learning

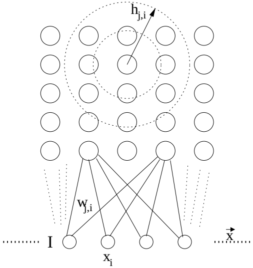


Abbildung: A two–dimensional self organizing map.

SOM Formation

- SOM Learning
  - Competition: input triggers winner–takes–all
  - Cooperation: identify topological neighborhood
  - Synaptic Adaptation: enhance response of winner and neighbors
- Competition, input $\vec{x}$, weight $\vec{w}_j$, winner $i(\vec{x}) = arg\ min_j|\vec{x} - \vec{w}_j|$
- Cooperation, output neuron position $\vec{r}_i$, neigborhood $h_{j,i(\vec{x})}$

  $$d_{j,i} = |\vec{r}_j - \vec{r}_i| \quad h_{j,i(\vec{x})} = e^{\frac{-d_{j,i}^2}{2\sigma(t)^2}} \quad \sigma(t) = \sigma_0 e^{\frac{-t}{\tau_1}}$$
- Adaptation

  $$\Delta\vec{w}_j = \eta(t) h_{j,i(\vec{x})}(t)(\vec{x} - \vec{w}_j(t)) \quad \eta(t) = \eta_0 e^{\frac{-t}{\tau_2}}$$

SOM Properties

- Approximation of input space, topological ordering
- Density matching, nonlinear principal components (PCA)

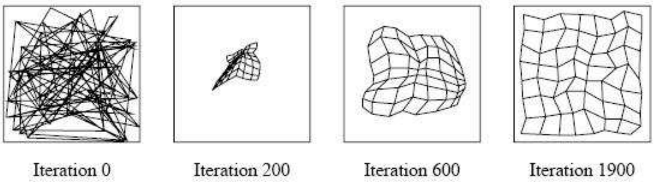


Abbildung: SOM formation of unit square topology (from `www.learnartificialneuralnetworks.com`)