1 Introduction

There will come a time when one will have to forget all that one has learned.

(Ramana Maharshi)

- Machine Learning (ML)
- Knowledge Discovery in Databases (KDD)
- Data mining (DM)

The basic principle of machine learning is the automatic modeling of underlying processes that have generated the collected data.

Knowledge Discovery in Databases

1. Understanding the problem area:
   - methodologies,
   - goals,
   - success criteria.
2. Data understanding:
   - knowing the data,
   - quality check,
   - searching for exceptions.
3. Data preparation:
   - collecting, acquisition,
   - evaluation, visualization,
   - uniformity of data,
   - cleaning, filtering,
   - transformation (discretization, mappings...).
4. Modeling, Machine Learning:
   - selection of ML method(s),
   - generating models,
   - internal evaluation,
   - repeating the process.
5. Evaluation of results:
   - various criteria,
   - accepting best models,
   - evaluation of DM process and determining the next step.
6. Application/usage:
   - who and when shall use the results,
   - problem of applicability (on new data),
   - practical usage of generated knowledge.

Learning from data results in

- rules,
- functions,
- relations,
- equation systems,
- probability distributions, etc.

Models explain the data and can be used for supporting decisions concerning the same underlying process (e.g., forecasting, diagnostics, control, validation, and simulations).

Overview of ML methods

- classification:
  - decision trees, decision rules,
  - naive Bayesian classifiers, Bayesian belief networks,
  - nearest neighbor classifiers,
  - linear discriminant functions,
  - logistic regression,
  - support vector machines, and
  - artificial neural networks.
- regression:
  - linear regression,
  - regression trees,
  - locally weighted regression,
  - support vector machines for regression, and
  - multi-layered feedforward neural networks for regression.
- associations and logical relations:
  - associative neural networks,
  - association rules,
  - inductive logic programming,
- discovery of equation systems,
- unsupervised learning - clustering,
- reinforcement learning.

Applications of ML

- Diagnosing the production process,
- Medical diagnosis and prognosis,
- Risk evaluation of insurance and loan applicants,
- Image classification (5×10^7 galaxies, 2×10^9 stars)
- Predicting the structure of chemical compounds and proteins,
- Game playing.

2 What is learning

Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.

Herbert A. Simon

A learning machine, broadly defined, is any device whose actions are influenced by past experience.

Nils J. Nilsson

- learning = accumulating knowledge from experience
- natural learning : machine learning

Intelligence: ability to adapt and to solve problems.

Knowledge: given in advance or learned.

intelligence =
hardware + background knowledge + learning + ?.

Artificial intelligence

The developement of systems that act intelligently and are able to solve relatively hard problems.

- machine learning,
- knowledge representation,
- machine perception,
- natural language understanding,
- reasoning and theorem proving,
- planning,
- playing games,
- logic programming,
- qualitative modeling,
- knowledge based systems,
- heuristic problem solving,
- robotics,
- cognitive modeling.

Why machine learning

- Cognitive modeling.
- Human learning is relatively slow.
- Automatic generation of knowledge bases for expert systems.
- Make the computer more flexible and adaptive to new situations.
- But: control adaptive systems!

Why machine learning

human	computer
forgetful quickly tired difficult transfer of knowledge	reliable, repeatable 24 hours/day trivial transfer of knowledge
limited memory ? slow ?	enormous data bases ? fast ?
broad knowledge parallel processing learns from errors dynamic knowledge	narrow specialized knowledge sequential processing repeats the same error static knowledge

NATURAL LEARNING

Learning is the progress of activity through experience.
Learnable are morality and intelligence.

Anton Trstenjak

- newborn learns to look and to listen,
- baby learns the meaning of words, the connection between sight and touch, and various moving skills
- child learns to walk, to speak words and later sentences,
- pupil learns to read, write and calculate, trains in abstract reasoning and logic inference,
- schoolgirl assimilates new descriptive knowledge, learns heuristics for problem solving,
- student restructures her knowledge, assimilates new specialized descriptive knowledge and special heuristics,
- at work we learn our profession, with experience we improve our performance and broaden the knowledge

- the psychology of learning
- educational psychology

Innate and learned

- Innate skills are called instincts.
- The (innate) ripening process can have periods when learning is possible and necessary.
- The higher the evolutive stage of species, the more important is the role of learning.
- A higher final level of the learning capability of species implies slower learning in childhood.
- a worm has 13 neurons, a bee 900 neurons
- a gorilla was successfully trained to understand more than one thousand words

Physiological abilities of the human brain

number of all neurons	10^{11}
number of neurons in the cerebral cortex	2×10^{10}
number of impulses per neuron	10 impulses/sec
speed of information flow through channels	30 bits/sec
number of chemical reactions in the brain	$10^5 - 10^6$/sec
number of synapses per neuron	10^4
number of all synapses	$10^{14} - 10^{15}$
number of skin pain points	1.2×10^6
number of fibres in an optic nerve	1.2×10^6
short term memory	7 pointers (addresses)
long term memory – synapses alone	$> 10^{15}$ bits
speed of the long term memory access	2 secs
optimal outside temperature	18°C
number of different aminoacids	20
aminoacid chain length	several thousands
potential number of different proteins	$> 20^{1000}$
genetic code	$> 10^9$ bits = 1G bit
number of produced protein molecules	15.000/sec/neuron

Memory

- Creating new connections between neurons
- Changing the connection strengths on the synapses
- Protein construction

- The human memorizes just everything
- The problem of addressing – associative memory
- Rationalization – a basis for generalization

Types of natural learning

Classified according to:

- learning complexity,
- learning material,
- learning strategy.

Learning complexity types

- Imprinting
- Conditioning and associating
- Probabilistic learning
- Memorizing
- Learning by trial and error
- Imitation
- Learning by understanding and insight (requires memorizing, abstract thinking, logical reasoning, and causal integration)

Learning material types

- sensorial learning,
- learning of motor skills,
- semantic (textual) learning.

Learning of concepts

- **Concepts** are classes or categories of objects, events, experiences or notions, which can be treated in the same way, or have certain common characteristics.
- Concepts can be
  - more or less concrete/abstract and
  - more or less simple/complex (e.g. conjunctive and disjunctive).

- Most of children's learning consists of acquiring existing concepts.
- A teacher provides (positive) examples and (negative) counterexamples and a partial description of a concept.
- Rarely does the learner have to create new concepts.

Learning strategy types

- Breadth first search,
- Depth first search,
- Conservative focusing: systematically changing one variable,
- Focused guessing: changing several variables at once.

Learning, intelligence, consciousness

Trying to understand yourself is like trying to bite your own teeth.

Alan Watts

Knowledge is important, however, much more important is its beneficial use. This depends on human mind and heart.

Dalai Lama

Identity between human and machine is not achieved by transferring the human characteristics to the machine but rather by transferring the mechanical limitations to the human.

Mortimer Taube

Learning, intelligence, consciousness

Intelligence is the ability to adapt to the environment and to solve problems.

intelligence =

hardware + background knowledge + learning + ?

- Amount of intelligence – hundreds of types
- Intelligence – defined according to human intelligence
- Consciousness

Consciousness

When you remove all thoughts, what remains is pure consciousness.

Ramana Maharshi

- Self awareness, differentiation of self from others

awareness of own problems, duties and responsibilities

- We do not know what is consciousness…
- Some quantum physicists relate consciousness with the collapse of the wave-function
- the materialistic explanation assumes consciousness appeared in a certain stage of evolution

States of consciousness

mental content	self awareness	
	YES	NO
YES	wakefulness	dreaming
NO	meditation	dreamless sleep

Levels of consciousness:

- pure consciousness,
- super-consciousness or altered state of consciousness,
- normal consciousness,
- subconsciousness ,
- unconsciousness

Limits of symbolic computability

It is impossible to teach the truth.

Osho

Theory of computability: only a tiny part of all problems, which can be formally defined, can be algorithmically solved.

- Science is limited to describable/ computable/ derivable models
- Formalisms:
  1. Algorithms,
  2. Recursive functions,
  3. Formal Grammars,
  4. Mathematical Logic
- All these formalisms have equivalent expressive power and they all have equivalent limitations

Limits of symbolic computability

- Discrete world: Q – rational numbers
- Continuous world: R – real numbers

$$|R| = 2^{|Q|} >> |Q|$$

- All formalisms have equivalent limitations: they can partially describe the phenomena in the discrete world and practically a negligible part of the continuous world.
- If the world is indeed continuous, then most probably it is undescribable by any of the formalisms which we are able to use with our (rational) mind.

Some important steps by AI

- Lenat's Automatic Mathematician,
- great successes of computers in playing complex games, such as checkers, backgammon, and chess,
- artificial neural networks for modeling the cognitive processes in the brain,
- several successes in generating new and beneficial knowledge from data.

Theory of learnability: all the limitations of computability hold also for learnability…

Possibility of artificial intelligence

Morality and intelligence are learnable.

Anton Trstenjak

Important aspects for understanding the abilities of artificial intelligence are:

- Impact of learning on intelligence,
- Faster is more intelligent,
- Limitations of intelligence,
- Imitating intelligent behavior.

If we omit the consciousness, a machine can in principle be intelligent to induce the sensation of AI.

Of course, such a machine still lacks consciousness.

(Im)possibility of artificial consciousness

- Intelligence: objectively measurable
- Consciousness: subjective,
  objectively cannot be verified!
- A conscous system can be immitated…

Consciousness is fundamentally related
to the following notions:

life, **intelligence**, **free will**.

(Im)possibility of artificial consciousness

Interesting speculations:

- Consciousness = life?
- More intelligence enables higher level of consciousness?
- Consciousness implicates free will?

Science and philosophy

Science without faith is lame, religion without science is blind.

Albert Einstein

- Models empirical, measurable data;
- theories (models) **describe** measurements;
- Repeatability/reliability/objectivity;
- Statistics;
- Experimentor and his intentions need to be excluded from the experiment.
- The purpose of universe/life is out of question – it is considered to be a **random coincidence**

Conclusions

We need education and the sense of moreal ethics – these two need to go together.

Dalai Lama

- Intelligence is the ability; consciousness is connected with ethics.
- Both polarities, **heart** and **intellect**, are inevitably needed.
- Intelligence (natural and AI) is a tool.
- Responsibility remains on your conscience.

3 Basic principles of machine learning (ML)

Definition of machine learning

1. learning as modeling,
2. minimum description length (MDL) principle
3. incremental learning,
4. principle of multiple explanations,
5. estimating probabilities

Definition of machine learning

Learning as modeling

- *learning algorithm*,
- *interpretation algorithm*.

automatically generated knowledge = *model* (*hypothesis*, *theory*)

machine learning = describing or *modeling* of data.

data + background knowledge $\longrightarrow$ description (model, hypothesis, theory)

background knowledge = hypothesis space + optimality criteria + initial hypothesis + heuristics + ?

machine learning = **optimization**:

Given: hypothesis space + criterion function

Find: hypothesis that maximizes the criterion function

Various kinds of models:

Discrete functions: *classification problems*

generalized mapping: codomain is the probability distribution

Continuous functions: continuous (or ordered discrete) class
regression problems.
generalized mapping: codomain is the confidence interval

Relations: more general than functions
greater complexity
ILP and equation systems

Minimum description length (MDL) principle

It is vain to do with more what can be done with fewer.

William of Ockham

Nature is pleased with simplicity, and affects not the pomp of superfluous causes.

Isaac Newton

Hypothesis should fit the background knowledge and the data.
Occam's Razor Principle:
The simplest hypothesis is most reliable (probable).

Criteria:

- Maximize the classification accuracy,
- Minimize the average misclassification cost,
- Minimize the complexity of the hypothesis,
- Maximize the fit to training data,
- Maximize the transparency of the hypothesis,
- Minimize the time complexity of the classification,
- Minimize the number of parameters for classification,
- Minimize the classification cost
- **Maximize the probability of the hypothesis**

Minimum Description Length (MDL) principle

$\mathcal{H}$ - hypothesis space
$H \in \mathcal{H}$ - hypothesis,
B - background knowledge
E - training data (evidence)

Optimal hypothesis:

$$H_{opt} = arg \max_{H \in \mathcal{H}} P(H|E, B)$$

Minimum Description Length (MDL) principle

$P(H|B)$ - prior probability of hypothesis H

Prior information content of H:

$$I(H|B) = -\log_2 P(H|B) \ \ [bit]$$

Posterior information content of H:

$$I(H|E,B) = -\log_2 P(H|E,B) \ \ [bit]$$

Optimal hypothesis:

$$H_{opt} = arg \ \min_{H \in \mathcal{H}} I(H|E,B)$$

Minimum Description Length (MDL) principle

Using the Bayes theorem:

$$I(H|B,E) = I(E|H,B) + I(H|B) - I(E|B)$$

$I(E|B)$ is constant, independent of H:

$$H_{opt} = arg \min_{H \in \mathcal{H}} (I(E|H,B) + I(H|B))$$

Communication channel:

Sender $\longrightarrow$ **data description** $\longrightarrow$ receiver

message = bit string
message length = the number of bits for coding the description (assuming optimal coding)
decoding: the description of the decoder is a part of the message

The task for the sender: **minimize the message length**

The sender can:

- code the data
- code the hypothesis + the derivation of the data

Hypothesis is **compressive** if:
model code length + derivation code length < data code length:

$$I(H|B) + I(E|H, B) < I(E|B)$$

B is available to both, the sender and the receiver

Incremental learning

Nature never says whether the guesses are correct. Scientific success consists of eventually offering a correct guess and never deviating from it thereafter.

(Osherson et al., 1986)

Incremental learning: update the theory after each new data
Problem: the smallest necessary change of the current hypothesis
Storing/forgeting the training instances
The algorithm never "knows" whether the hypothesis is correct.

Principle of multiple explanations

If more than one theory is consistent with the observations, keep all theories.

Epicurus

For each particular model find what its prediction is and then weight its prediction with the probability of that model, and the weighted sum of those predictions is the optimal prediction.

Peter Cheeseman

Keep all hypotheses that are consistent with the data (that are plausible)!

MDL principle:
searching for one best hypothesis
(guiding the search of the learning algorithm)

Principle of multiple explanations:
combine all plausible hypotheses
(guiding the interpretation algorithm)

$\mathcal{R}$ - solution space

$r \in \mathcal{R}$ - solution (prediction, outcome)

$P(r|H)$ - probability of solution r, if the hypothesis H is correct

Solution, provided by the optimal hypothesis:

$$r_{opt1} = arg \max_{r \in \mathcal{R}} P(r|H_{opt})$$

But, it is not the optimal solution!

Optimal interpretation algorithm uses all plausible hypotheses:

$$r_{opt} = arg \max_{r \in \mathcal{R}} \sum_{H \in \mathcal{H}} (P(r|H)P(H|E,B))$$

Seems contradictory: combination of several theories = theory

we changed the space of possible theories: $n \longrightarrow >> 2^n$

Estimating probabilities

(estimate the probability from small number of training instances)
Prior probability density: beta distribution $\beta(a, b)$:

$$p(x) = \begin{cases} \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1} & 0 \leq x \leq 1 \\ 0 & otherwise \end{cases}$$

$$B(a, b) = \int_0^1 x^{a-1}(1-x)^{b-1} dx$$

We interpret $a > 0$ and $b > 0$ as: a successful out of $a + b$ trials
Mathematical expectation of random variable p, distributed according to $\beta(a, b)$:

$$Exp(p) = \frac{a}{a+b}$$

Let us have r successful out of n trials. The probability of success is defined with:

$$p = \frac{r + a}{n + a + b}$$

- initial distribution $\beta(0, 0)$: *relative frequency*

$$p = \frac{r}{n}$$

- initial distribution $\beta(1, 1)$: *Laplace's law of succession*
  k = number of different outcomes:

$$0 < p = \frac{r + 1}{n + k} < 1$$

Estimating probabilities

- $m = a + b$, $p_0 = \frac{a}{a+b}$: *m-estimate*

$$p = \frac{r + mp_0}{n + m} = \frac{n}{n + m} \times \frac{r}{n} + \frac{m}{n + m} \times p_0$$

$m = k$, $p_0 = 1/k \longrightarrow$ Laplace

Estimating the results of learning

1. classification accuracy,
2. confusion matrix,
3. misclassification cost,
4. Brier score,
5. information score,
6. sensitivity and specificity, ROC curve,
7. recall and precision,
8. mean squared error (MSE) and mean absolute error (MAE),
9. correlation coefficient.

Estimating the results of learning

all instances: *training* and *testing* instances

classification accuracy

M_R - the number of classes

N - the number of all possible problems

$N^{(p)}$ - the number of correct solutions

N_t - the number of testing instances

$$A = \frac{N^{(p)}}{N} \times 100\% \sim A_t = \frac{N_t^{(p)}}{N_t} \times 100\%$$

Classification accuracy on N_l training instances = upper bound:

$$A_l = \frac{N_l^{(p)}}{N_l} \times 100\%$$

$A_l >> A_t$: *overfitting*

Majority class: lower bound = **default** accuracy A_d

$N_l^{(i)}$ - the number of learning instances from i-th class

$$A_d = \max_i \frac{N_l^{(i)}}{N_l}$$

$A_t < A_d$: the classifier is unusable

Breast cancer problem: $A_d = 80\%$

Most of the attributes are unusable for prediction.

Prior distribution of classes: $P_1 = 0.8$ in $P_2 = 0.2$

Classification accuracy: $P_1^2 + P_2^2 = 68\%$

Confusion matrix

true class	classified as C1	C2	C3	sum
C1	12.3	2.4	8.5	23.2
C2	5.5	58.7	2.1	66.3
C3	0.0	2.0	8.5	10.5
sum	17.8	63.1	19.1	100.0

Misclassification cost

$N_t^{(ij)}$ - the number of testing instances from i-th class, assigned to the j-th class.

$$N_t^{(p)} = \sum_{i=1}^{M_R} N_t^{(ii)}$$

C_{ij} - misclassification cost

Allow:$C_{ij} \neq C_{ji}$ Usually: $C_{ii} = 0$, $i = 1, ..., M_R$
Average misclassification cost:

$$C_t = \frac{\Sigma_{i,j}(C_{ij}N_t^{(ij)})}{N_t}$$

Brier score

M_R number of classes
$r^{(j)}$ class of the j-th testing instance
$P'_j(r_i), i = 1..M_R$ predicted probability distribution
$C_j(r^{(j)}) = 1$ and $C_j(r_i) = 0, r_i \neq r^{(j)}$ true distribution
N_t number of testing instances:

$$Brier = MSE_P = \frac{\Sigma_{j=1}^{N_t} \Sigma_{i=1}^{M_R} (C_j(r_i) - P'_j(r_i))^2}{N_t}$$

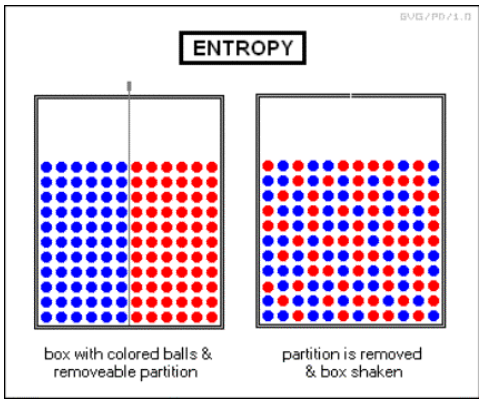
classifier quality: $1 - Brier/2$.

Information score

Answers as probability distributions
Prior and posterior probabilities of classes

domain	A_t	A_d	M_R	$H(R)$
breast cancer	80%	80%	2	0.72 bit
primary tumor	45%	25%	22	3.64 bit

Prior probability of i-th class: $P(r_i) = \frac{N_l^{(i)}}{N_l}$



Information content for i-th class:

$$H(r_i) = -\log_2 P(r_i) \quad [bit]$$

Class entropy: $H(R) = -\sum_{i=1}^{R} (P(r_i) \log_2 P(r_i))$

$H(R)$ is maximal for equally probable classes:

$$P(r_i) = \frac{1}{M_R}, \qquad i = 1, ..., M_R$$

and is minimal, if all instances belong to same class r_j:

$$P(r_j) = 1, \qquad and \qquad P(r_i) = 0, \quad i \neq j$$

Average information score:

$r^{(j)}$ - the correct class of the j-th instance
$P'(r^{(j)})$ - posterior probability of that class

$$Inf = \frac{\sum_{j=1}^{N_t} Inf_j}{N_t} \qquad [bit]$$

and

$$Inf_j = \begin{cases} -\log_2 P(r^{(j)}) + \log_2 P'(r^{(j)}), & P'(r^{(j)}) \geq P(r^{(j)}) \\ -(-\log_2(1 - P(r^{(j)})) + \log_2(1 - P'(r^{(j)}))), & P'(r^{(j)}) < P(r^{(j)}) \end{cases}$$

Let us have 2 classes: $P(r_1) = 0.8$ and $P(r_2) = 0.2$
Posterior distribution, returned by the classifier:
$P'(r_1) = 0.6$ and $P'(r_2) = 0.4$

- If the correct class is r_1:
  classification accuracy: the answer is correct
  information score: the answer is incorrect (misleading)

- If the correct class is r_2:
  classification accuracy: the answer is incorrect
  information score: the answer is correct (useful)

Bounds: $0 \leq Inf \leq H(R)$

Relative information score:

$$RInf = \frac{Inf}{H(R)} \times 100\%$$

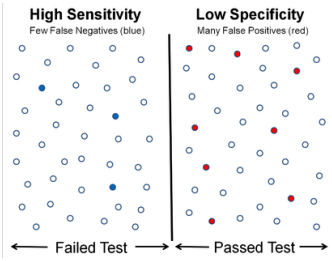
Sensitivity and specificity

true class	classified as P	classified as N	sum
P	TP	FN	POS=TP+FN
N	FP	TN	NEG=FP+TN
sum	PP=TP+FP	PN=FN+TN	n = TP+FP+FN+TN

$$Sensitivity = \frac{TP}{TP + FN} = \frac{TP}{POS}$$

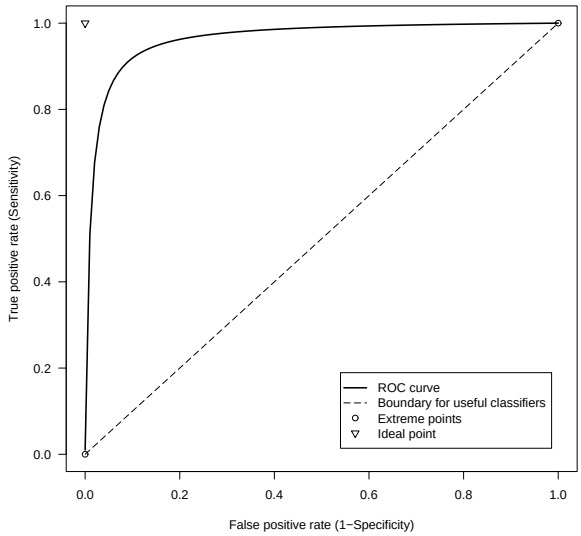
$$Specificity = \frac{TN}{TN + FP} = \frac{TN}{NEG}$$

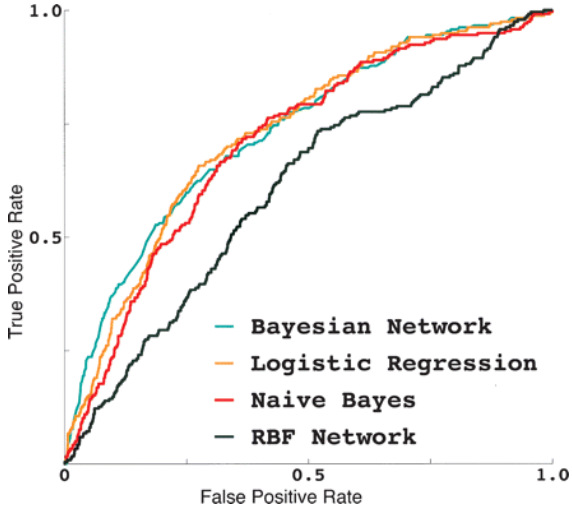
$$Accuracy = \frac{TP + TN}{TN + FP + FN + TN} = \frac{TP + TN}{n}$$



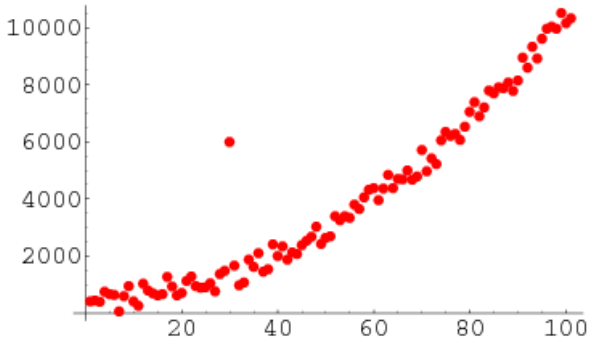
ROC (Reciever Operating Characteristic) curve

Area under the ROC Curve (AUC)





Mean squared error (MSE)



Regression problems:
predicted value: $\hat{f}(i)$ true value: $f(i)$

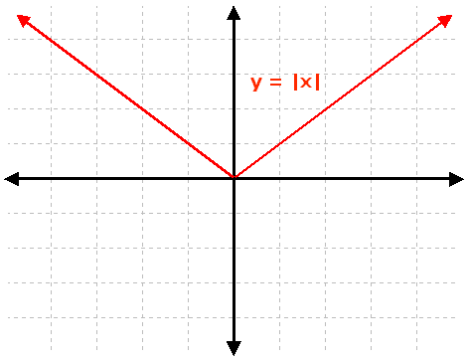
$$E = \frac{1}{N}\sum_{i=1}^{N}(f(i) - \hat{f}(i))^2$$

Relative mean squared error:

$$0 \leq RE = \frac{N \times E}{\Sigma_i (f(i) - \overline{f})^2} \leq 1 \quad where \quad \overline{f} = \frac{1}{N}\sum_i f(i)$$

$\hat{f}(i) = \overline{f} \longrightarrow RE = 1$ trivial f. ($RE > 1 \rightarrow$ unusable)
$\hat{f}(i) = f(i) \longrightarrow RE = 0$ ideal function

Mean absolute error (MAE)



$$MAE = \frac{1}{N} \sum_{i=1}^{N} |f(i) - \hat{f}(i)|$$

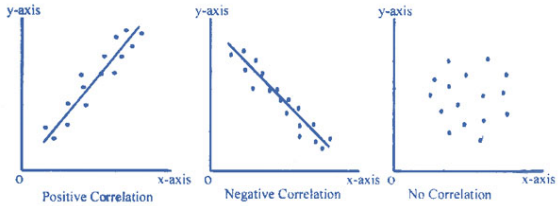
relative MAE:

$$RMAE = \frac{N \times MAE}{\Sigma_i \, |f(i) - \overline{f}|}$$

$$0 \leq RMAE \leq 1$$

Correlation coefficient

$$K = \frac{S_{f\hat{f}}}{S_f S_{\hat{f}}}$$



where:

$$S_{f\hat{f}} = \frac{\Sigma_{i=1}^{N}[(f(i) - \overline{f})(\hat{f}(i) - \overline{\hat{f}})]}{N-1}$$

$$S_f = \frac{\Sigma_{i=1}^{N}(f(i) - \overline{f})^2}{N-1}$$

$$S_{\hat{f}} = \frac{\Sigma_{i=1}^{N}(\hat{f}(i) - \overline{\hat{f}})^2}{N-1}$$

Reasonable only positive values: $0 < K \leq 1$

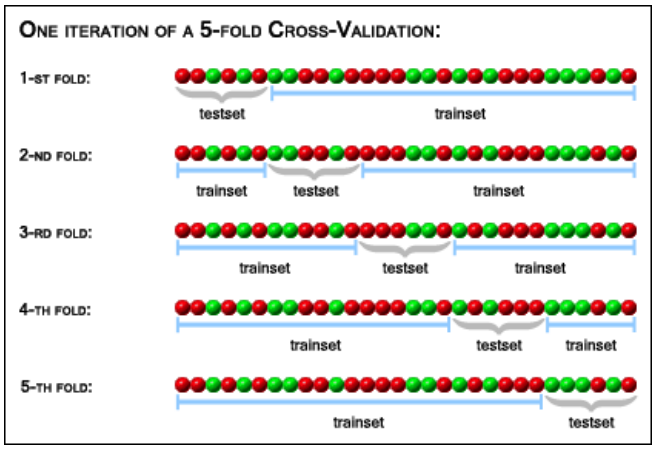
Estimating performance:

1. cross-validation,
2. bootstraping,
3. comparing the performance of different algorithms,
4. combining several ML algorithms.

Cross-validation

- leave-one-out:
  N instances: have to generate $N + 1$ hypotheses
- "leave N/K out": *K-fold cross-validation*
  K - the number of generated hypotheses; usually $K = 10$

- standard error $(1 - \alpha = 68\%)$
- confidence interval (e.g. $1 - \alpha = 95\%$)



Comparing the performance of different ML algorithms

Learning: *learning/training set*
Parameter tuning: *validation set*
Testing: *testing set*
Frequent error: testing set used as validation set...

Significance tests:
evaluate the significance of performance differences

Comparing the performance of different ML algorithms

- Two algorithms on one domain
  - Cross validation: one-tailed t-test
  - leave-one-out or independent testing set: one-tailed z-test
  - Bonferroni correction (N comparisons): $\alpha_B \approx \alpha/N$
- Two algorithms on several domains: Wilcoxon signed rank test
- Several algorithms on several domains: Friedman test
  - One algorithm against the others: Bonferroni-Dunn test
  - Each algorithm against all others: Nemenyi test

Combining several ML algorithms

- different learning algorithms
- the same algorithm with different parameters or on different versions of the learning set

Combining the predictions of different hypotheses:

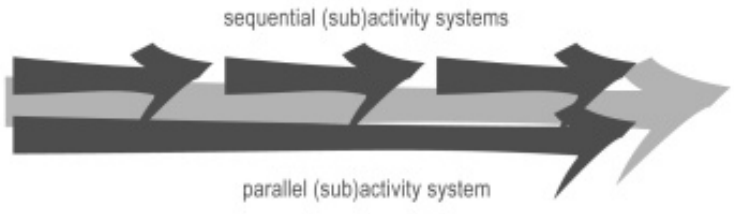
- voting
- weighted voting
- voting, weighted with the prediction reliability
- Naive Bayesian combination
- Meta-learning of the combination function (stacking)
- Locally weighted voting

Combining algorithms

Parallel

Sequential: the first algorithm learns the target function. The next algorithm learns to correct errors of the previous one.

Mixed... (problem of overfitting)



Bagging

- "bootstrap agreggating"(Breiman, 1996): a sequence of hypotheses from different learning sets
- n times randomly pick an example with replacement
- some of the examples are not contained in the set (36.8%)
- All hypotheses vote for the target.
- Improves the performance of unstable algorithms with high variance.
- No overfitting with increasing the number of hypotheses.

Random forests

special version of bagging: uses bootstraped data sets

Better accuracy of decision/regression tree algorithms.

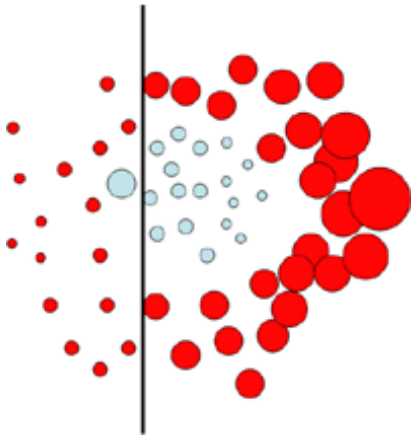
- For each node: randomly select a small subset of attributes as candidates for the best attribute.
- 100 or more trees.
- Classification: voting by all trees.
- Roboust - descreases the variance.
- Weakness: lost transparency.

Boosting

- weighing of learning instances with their "difficulty";
- assumption: algorithm can deal with weighted examples;
- otherwise we generate the learning set as with bagging, only the non-uniform probability distribution.
- In the first iteration all learning instances have weight 1.0. The hypothesis is used to predict the target variable for all instances. The correctly predicted examples decrease their weights and the incorrectly predicted ones increase them.
- e = error of prediction, the weight multiplied by $e/(1-e)$. Finally normalize all the weights to the overall sum n.

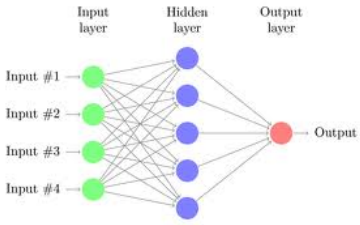
Boosting

- For prediction use all hypotheses, weighted with their accuracy on the (weighted) learning set from which it was generated.
- Weight: $-log(f/(1-f))$, where f is the hypothesis error.
- Boosting often more accurate than bagging and can be used also with stable algorithms with low variance.
- Weakness: sometimes overfitting.



Neural networks: Artificial neuron is an abstraction of the biological one.

$$x_{out} = f\Big(\sum_i w_i x_i + w_{bias}\Big)$$



A possible threshold function f:

$$f(X) = \begin{cases} 1 & X > 0 \\ 0 & X \leq 0 \end{cases}$$

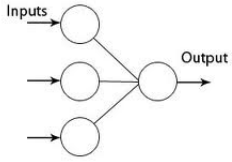
More frequently a sigmoid function is used:

$$f(X) = \frac{1}{1 + e^{-X}}$$

Topology: feedforward multilayered artificial neural networks.

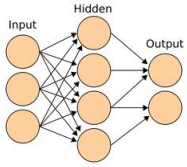
Connections between neurons are called *synapses*.
The matrix of all synapse weights defines the function that is calculated by the neural network.

- classification problem: n_0 output neurons,
- regression problem: one output neuron



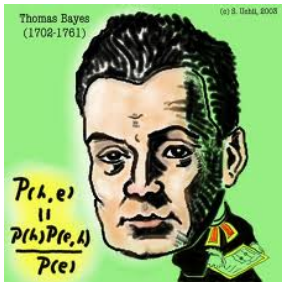
Two-layered networks: linear problems.

Most frequently three/four-layered feedforward networks are used. The number of hidden layers and hidden neurons determines the modelling capabilities of a neural network.



Naive Bayesian classifier

Assumes conditional independence of attributes given the class.



$$P(r_k|V) = P(r_k)\frac{P(V|r_k)}{P(V)}$$

Conditional independence assumption:

$$P(V|r_k) = P(v_1 \;\&\; ... \;\&\; v_a|r_k) = \prod_{i=1}^{a} P(v_i|r_k)$$

$$P(r_k|V) = \frac{P(r_k)}{P(V)} \prod_{i=1}^{a} P(v_i|r_k)$$

Once more the Bayesian rule:

$$P(v_i|r_k) = P(v_i)\frac{P(r_k|v_i)}{P(r_k)}$$

$$P(r_k|V) = P(r_k)\frac{\Pi_{i=1}^{a} P(v_i)}{P(V)} \prod_{i=1}^{a} \frac{P(r_k|v_i)}{P(r_k)}$$

The following factor:

$$\frac{\Pi_{i=1}^{a} P(v_i)}{P(V)}$$

is independent of class and can be omitted:

$$P(r_k|V) = P(r_k) \prod_{i=1}^{a} \frac{P(r_k|v_i)}{P(r_k)}$$

Overview of search algorithms

ML problems that require search:

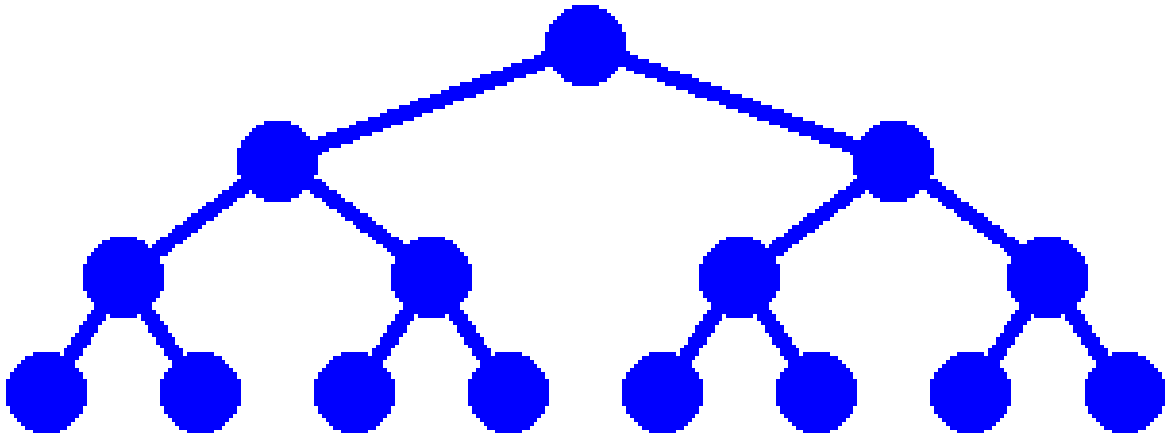
- hypothesis (model) learning
- feature subset selection
- parameter tuning
- constructive induction (searching novel attributes)
- discretization of attributes

Selection of search strategy

Usually, problem spaces are very large.

Search strategy is selected with respect to the size of problem space and its properties.

1. Search for optimal solution
2. Search for suboptimal solution

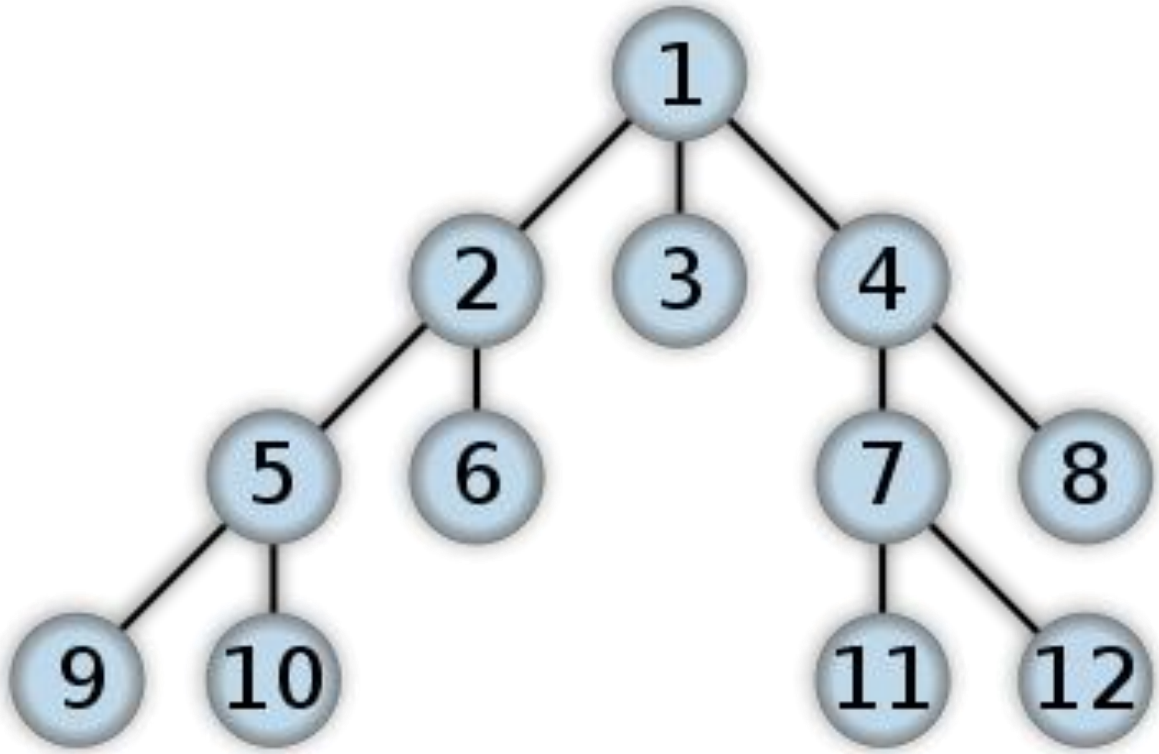


Searching optimal solutions

Basic search strategies

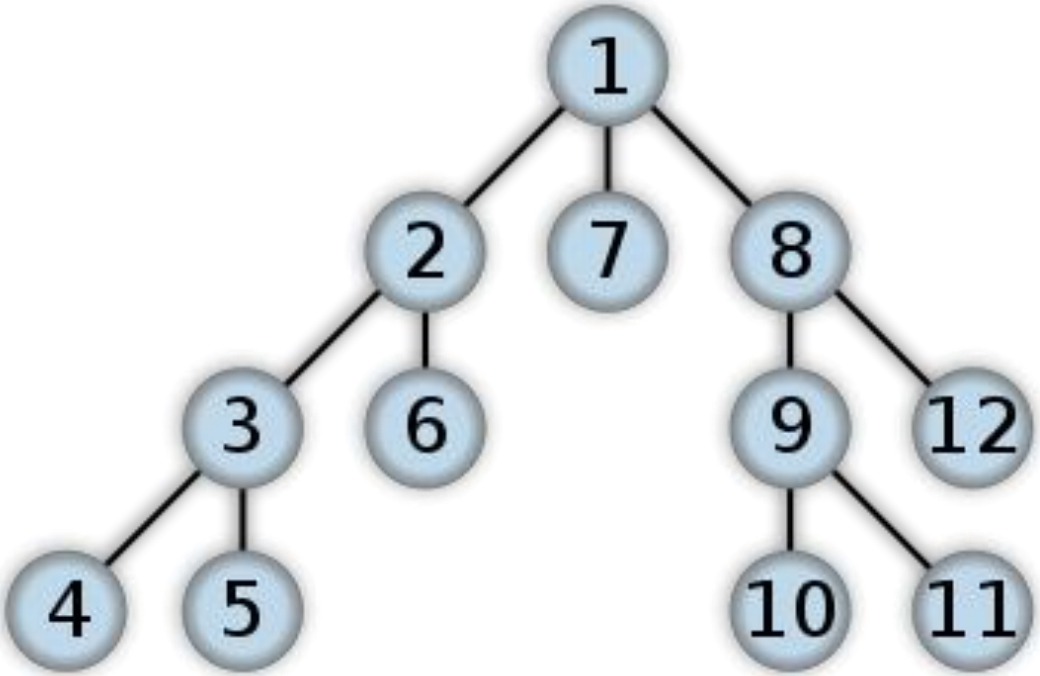
1. exhaustive search
   a. breadth-first search
   b. depth-first search
   c. iterative deepening
2. bounded exhaustive search (branch and bound)
3. best-first search
4. dynamic programming

breadth-first search



+ guarantees shortest path to the solution.
- exponential space complexity!

depth-first search



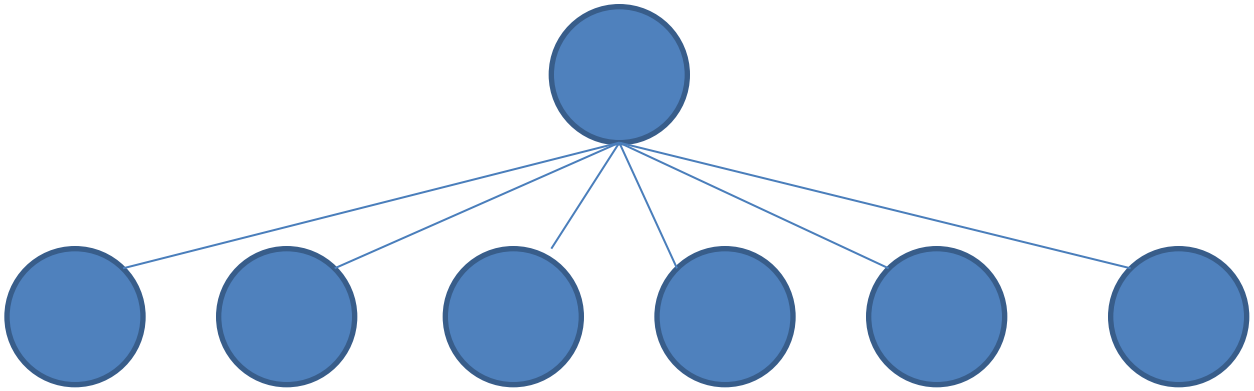
+ linear space complexity.
- can miss the solution close to the initial state
- cannot avoid cycling (lost in infinite branches).

Iterative deepening

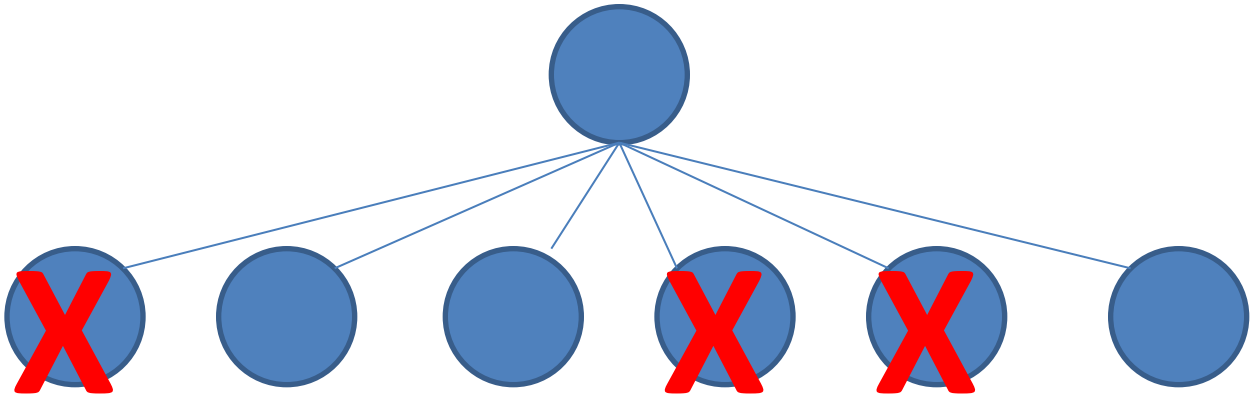
```
max = initial depth;
repeat:
        depth-first search with maxdepth = max;
        max = max + 1;
```

+ guarantees shortest path to the solution
  (no problems with cycling)
+ linear space complexity
- in each iteration again searches the space already searched → exponential time complexity!

Bounded exhaustive search: branch and bound

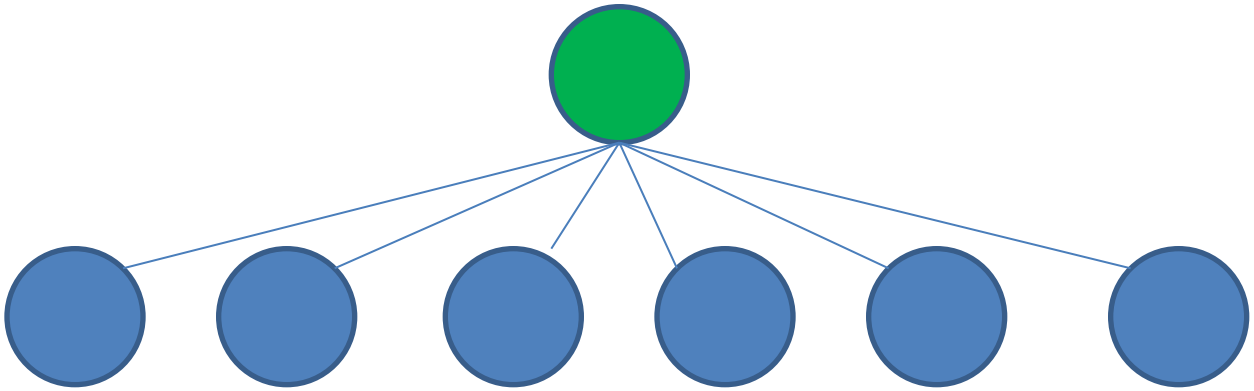


Bounded exhaustive search: branch and bound

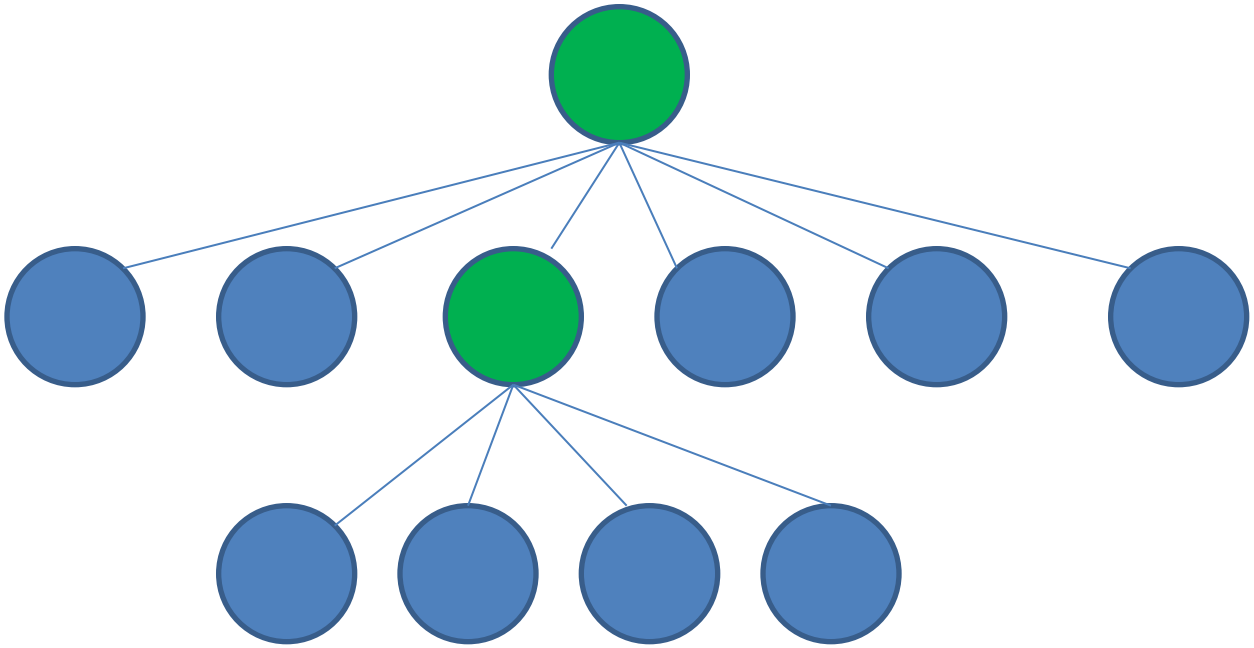


Efficiency of bounding the space depends of our knowledge about the search space.

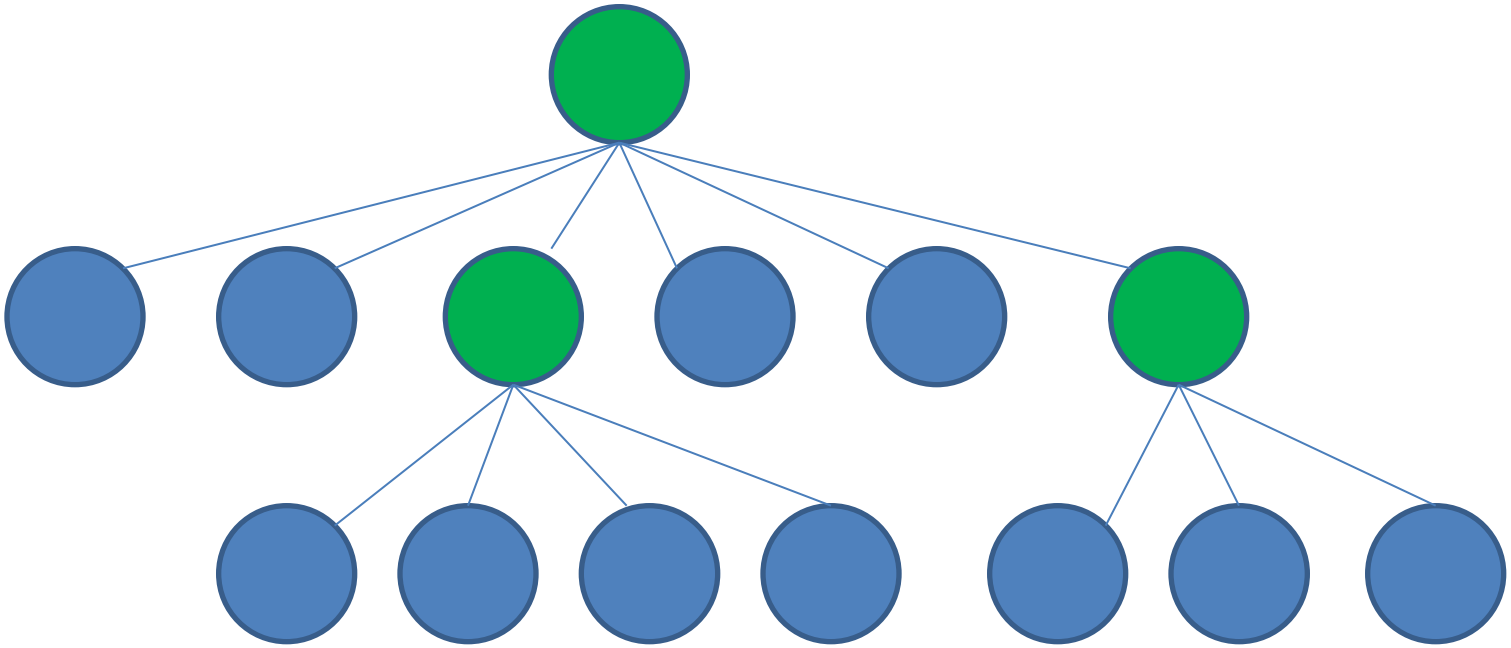
Best-first search



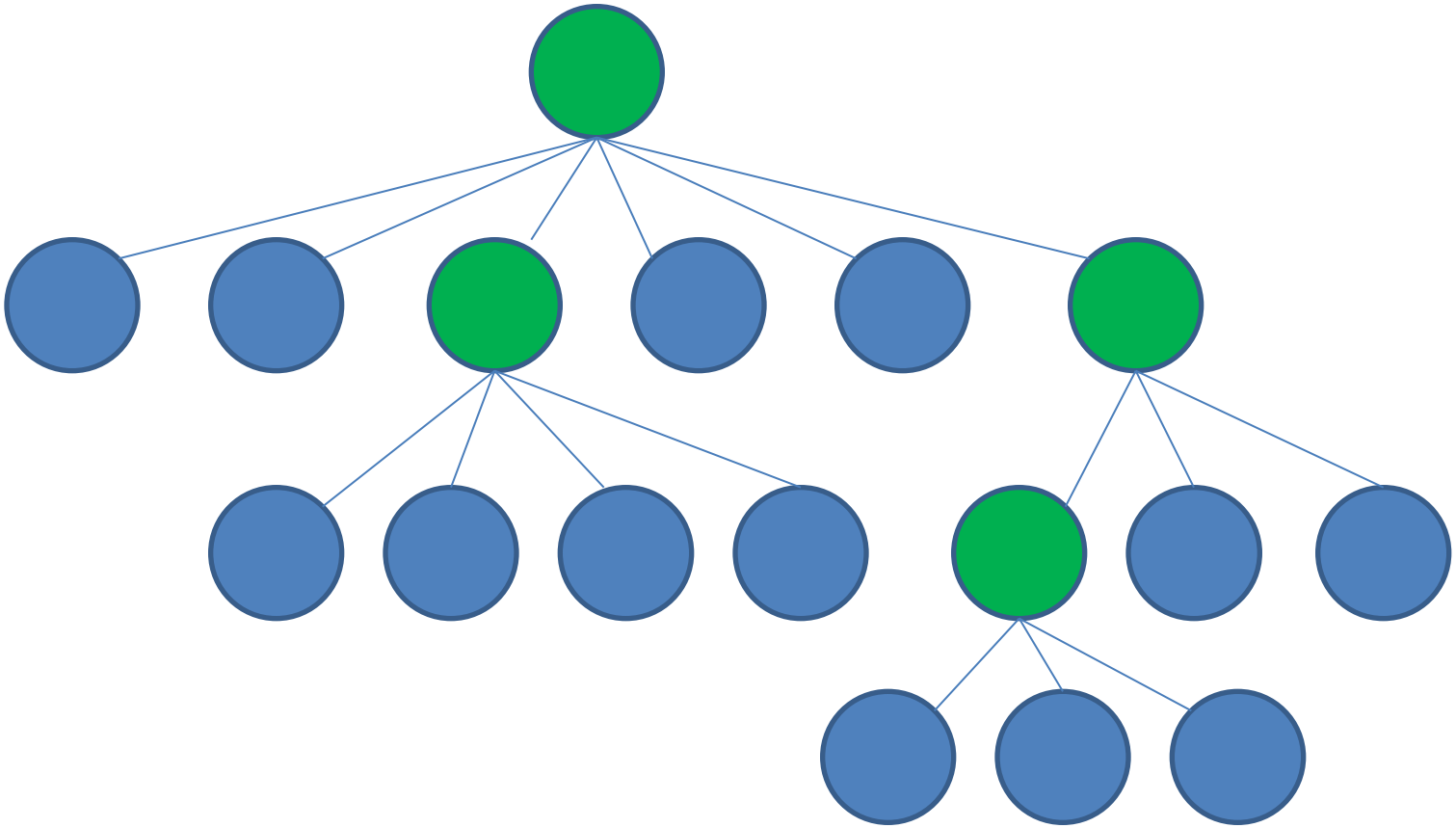
Best-first search



Best-first search

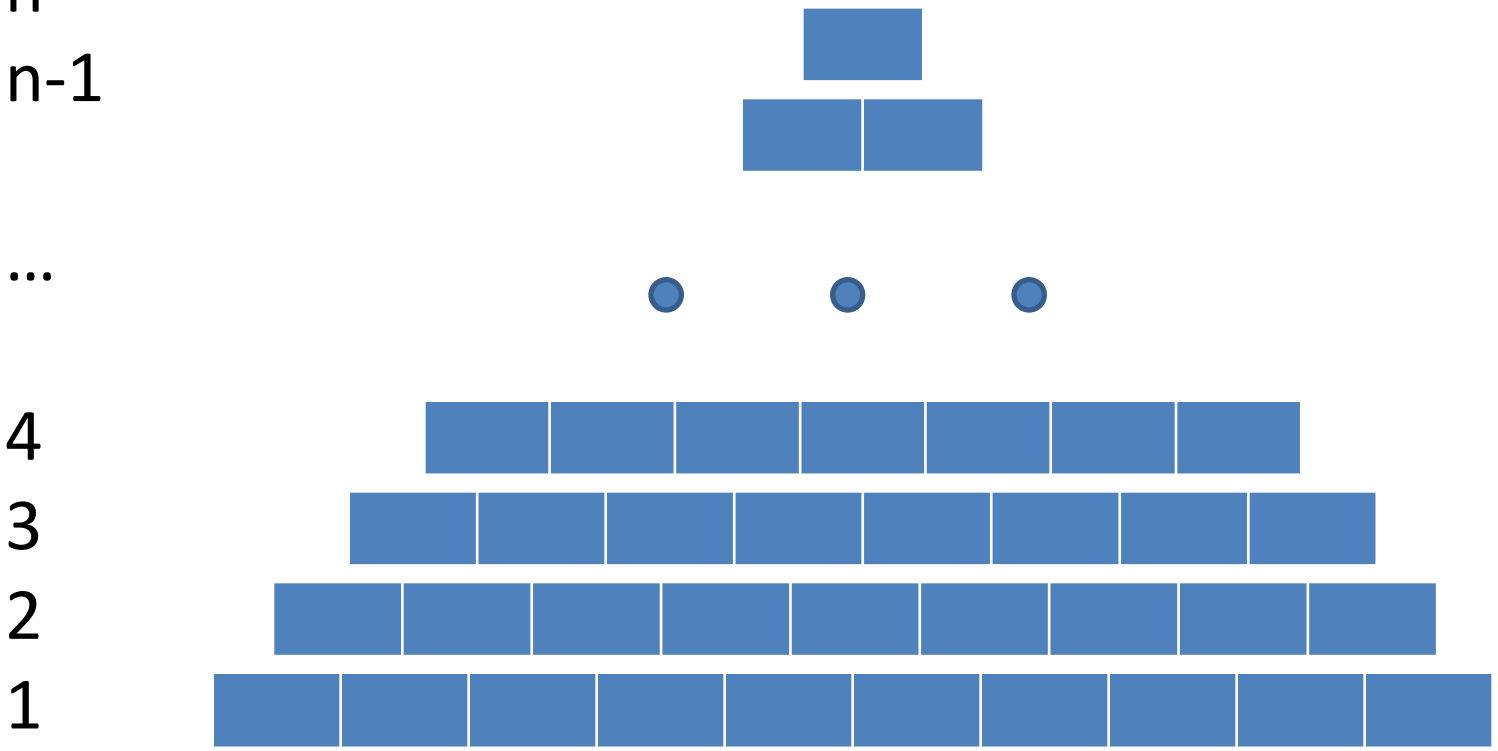


Best-first search



+ in combination with “branch and bound” one of the best strategies.
- exponential space complexity!

Dynamic programming

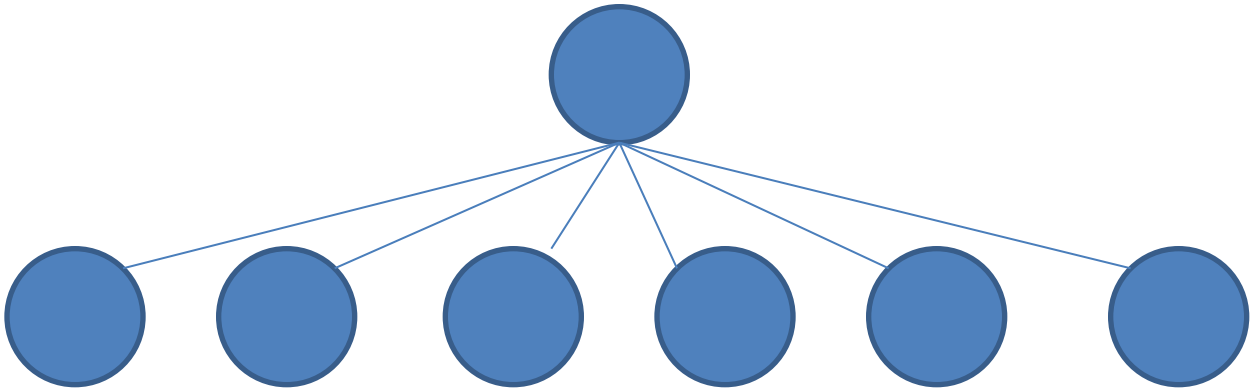


Searching suboptimal solutions

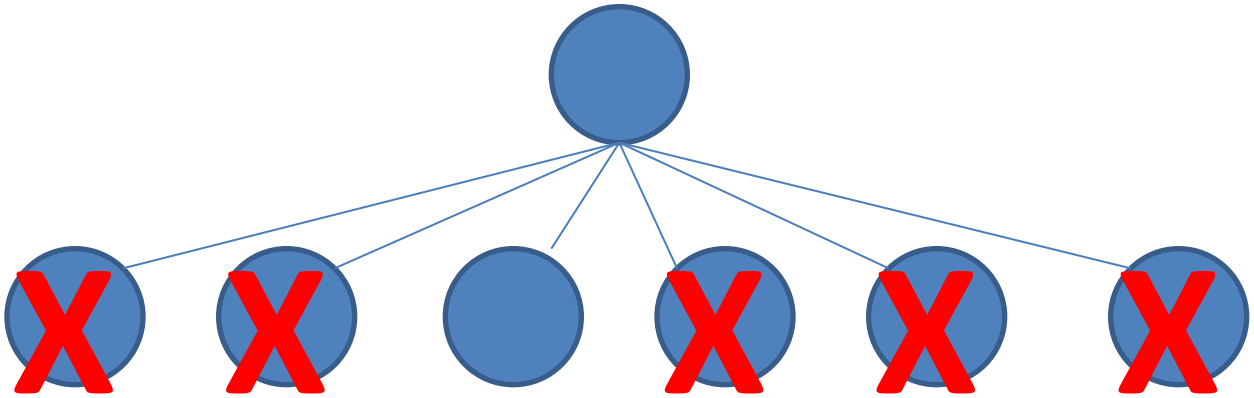
Basic search strategies

1. greedy search (best-only search)
2. beam search
3. Local optimization
4. Gradient search
5. Stochastic algorithms:
    a. Simulated annealing
    b. Genetic algorithms

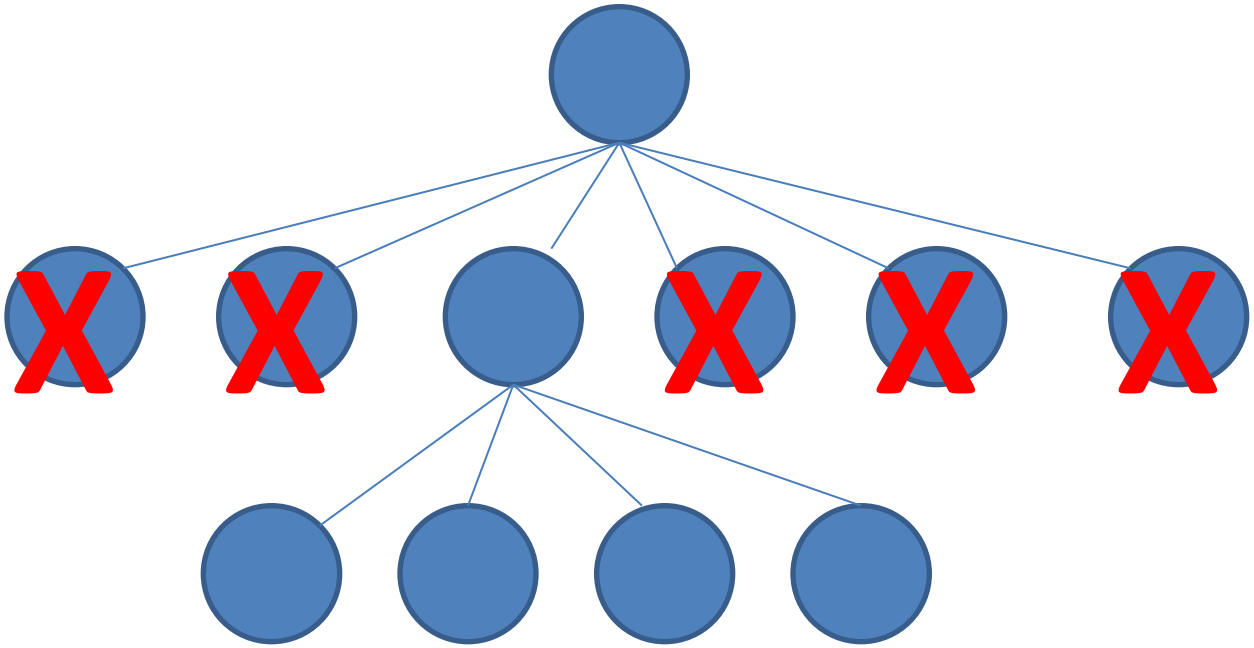
Greedy search



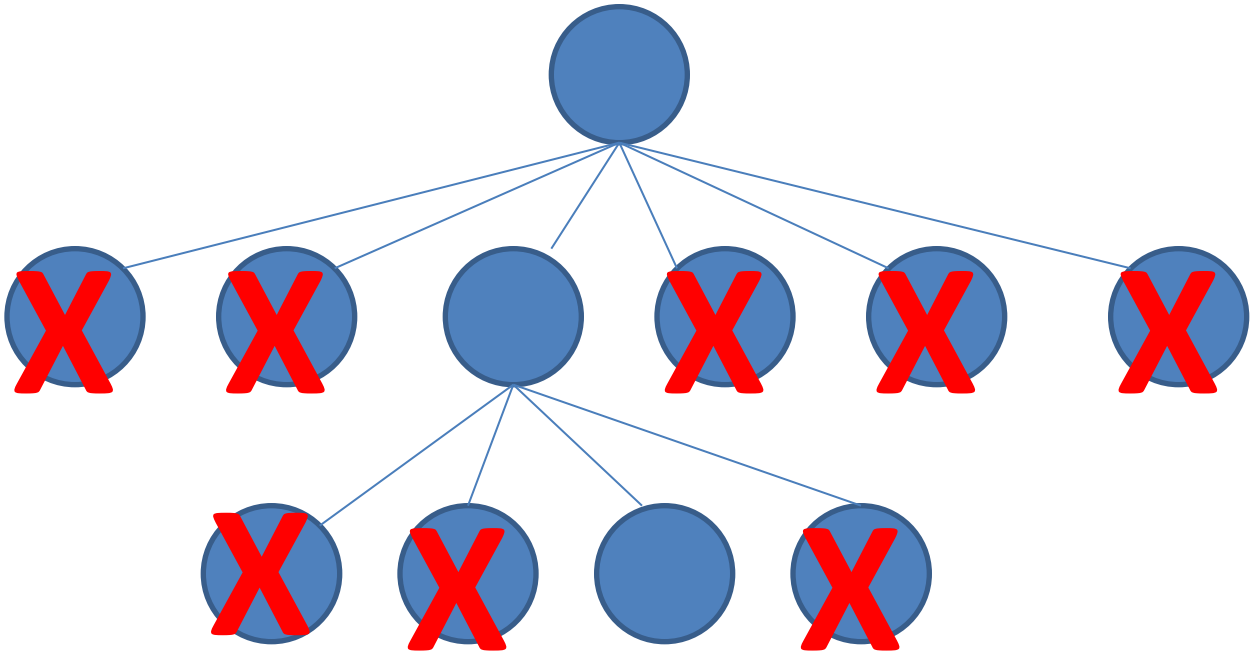
Greedy search



Greedy search

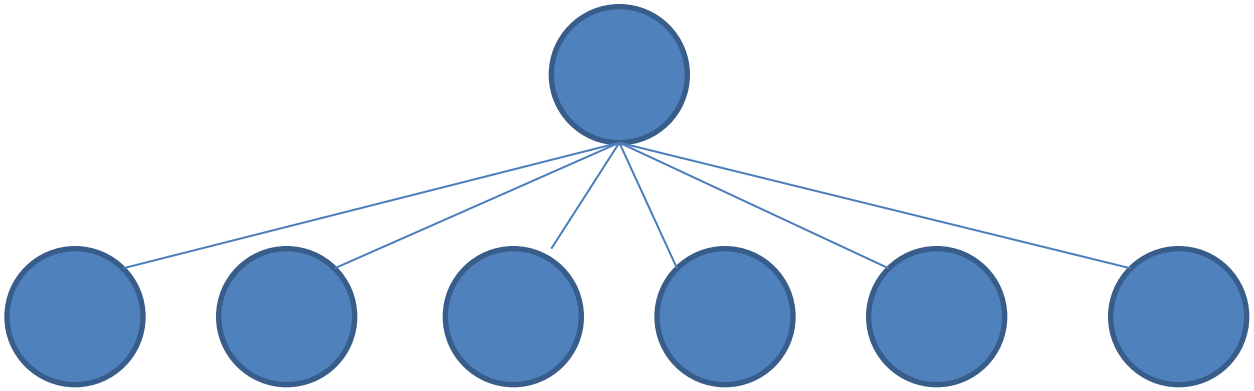


Greedy search

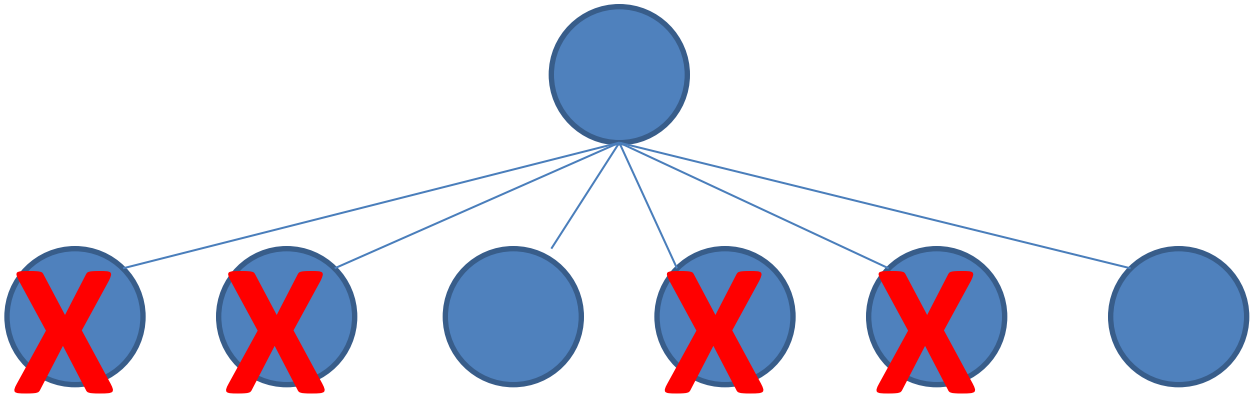


+ search advances very rapidly.
- Algorithm is myopic → solution may be suboptimal
+ in many practical problems the solution is satisfactory.

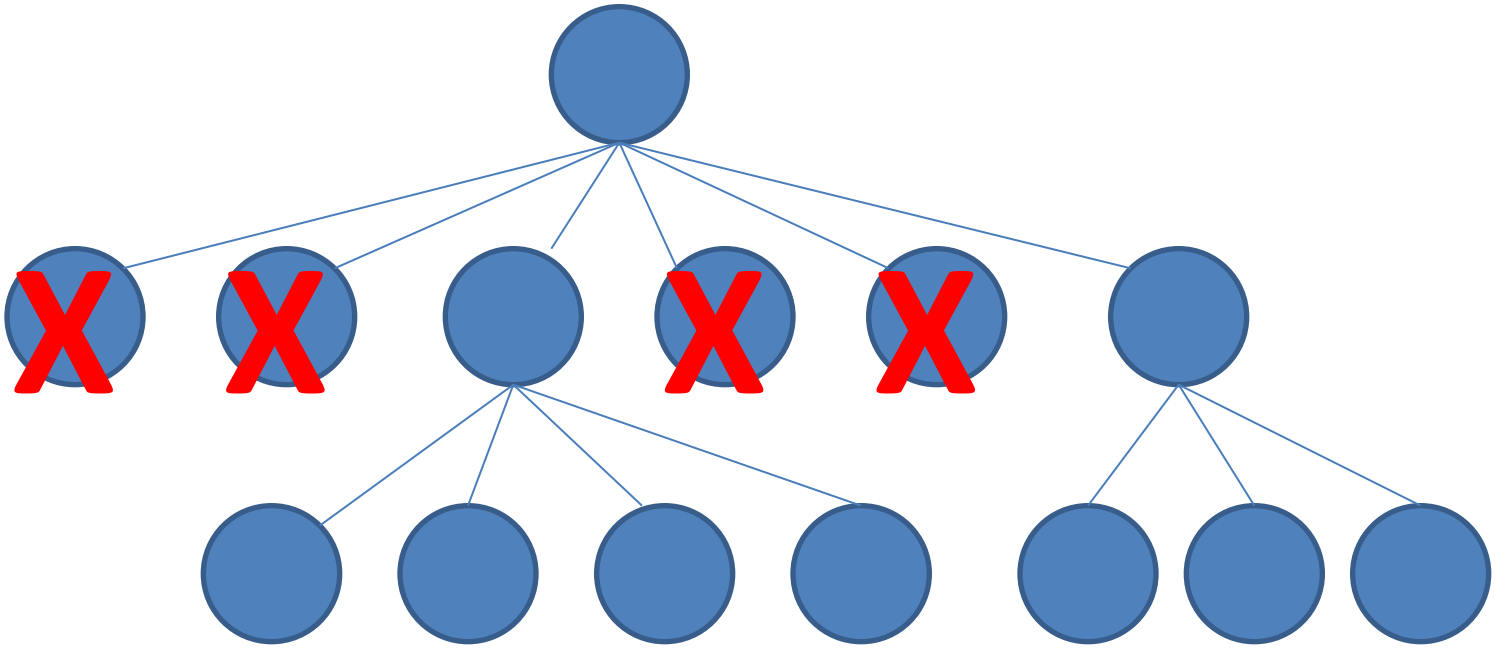
Beam search



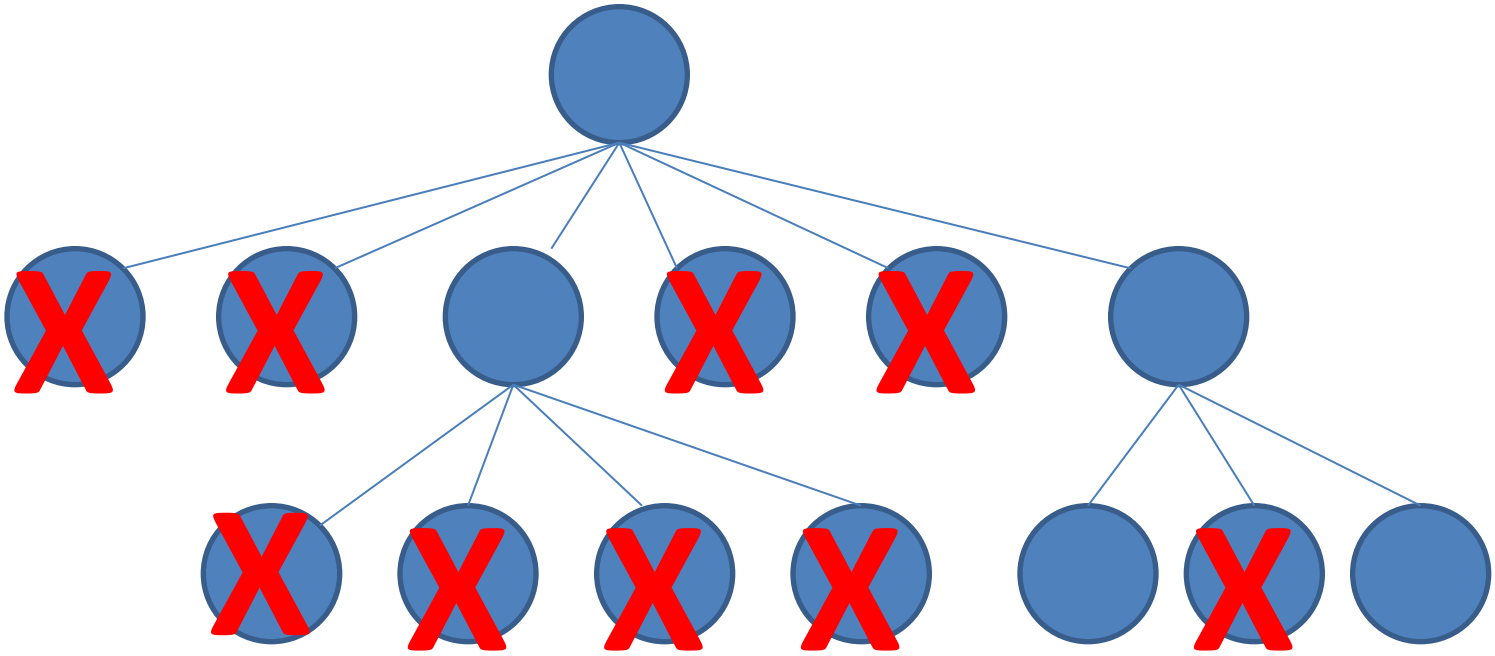
Beam search



Beam search



Beam search



Beam search generalizes greedy search:

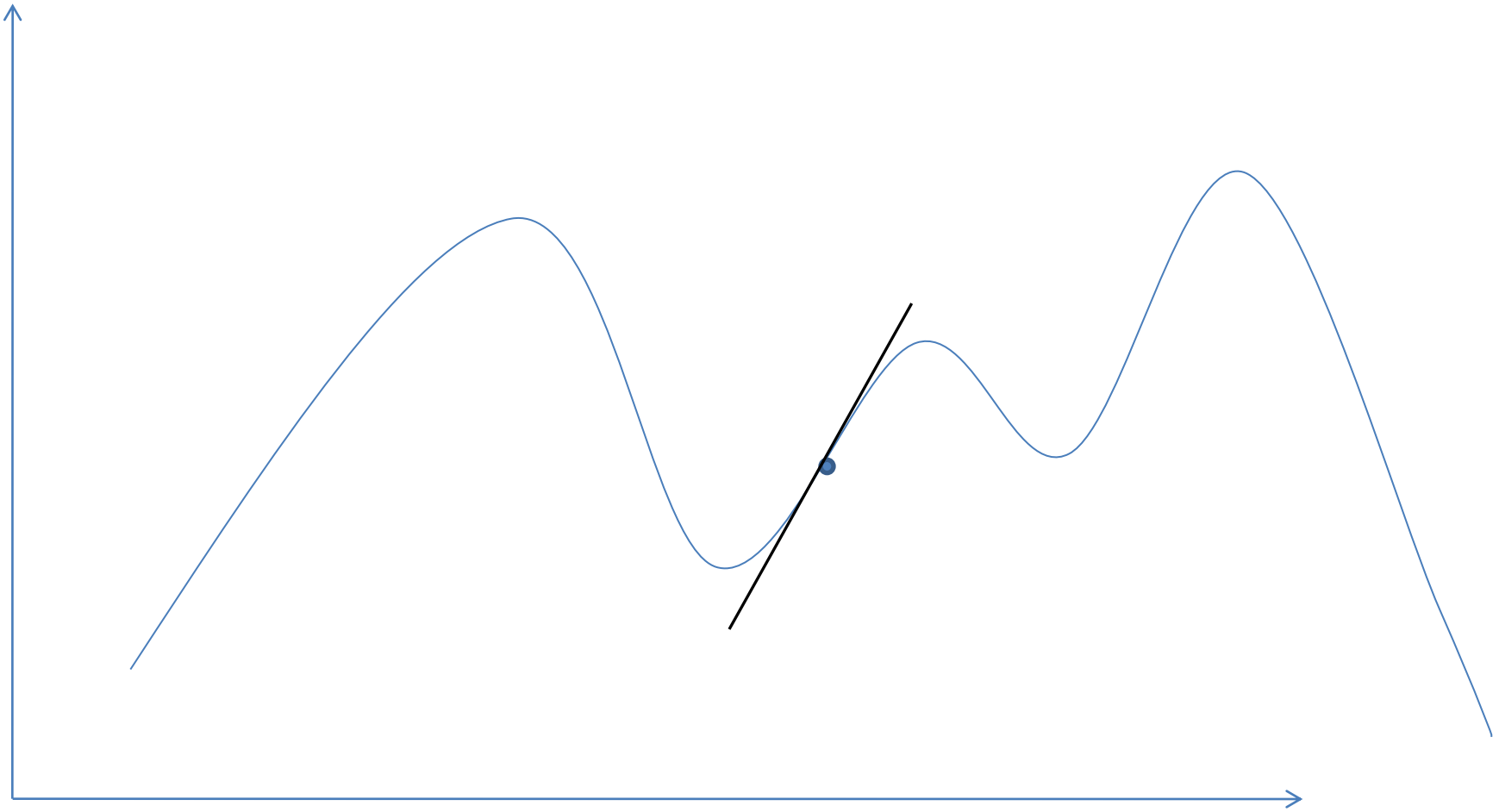
beam size k = 1 → greedy search

Local optimization

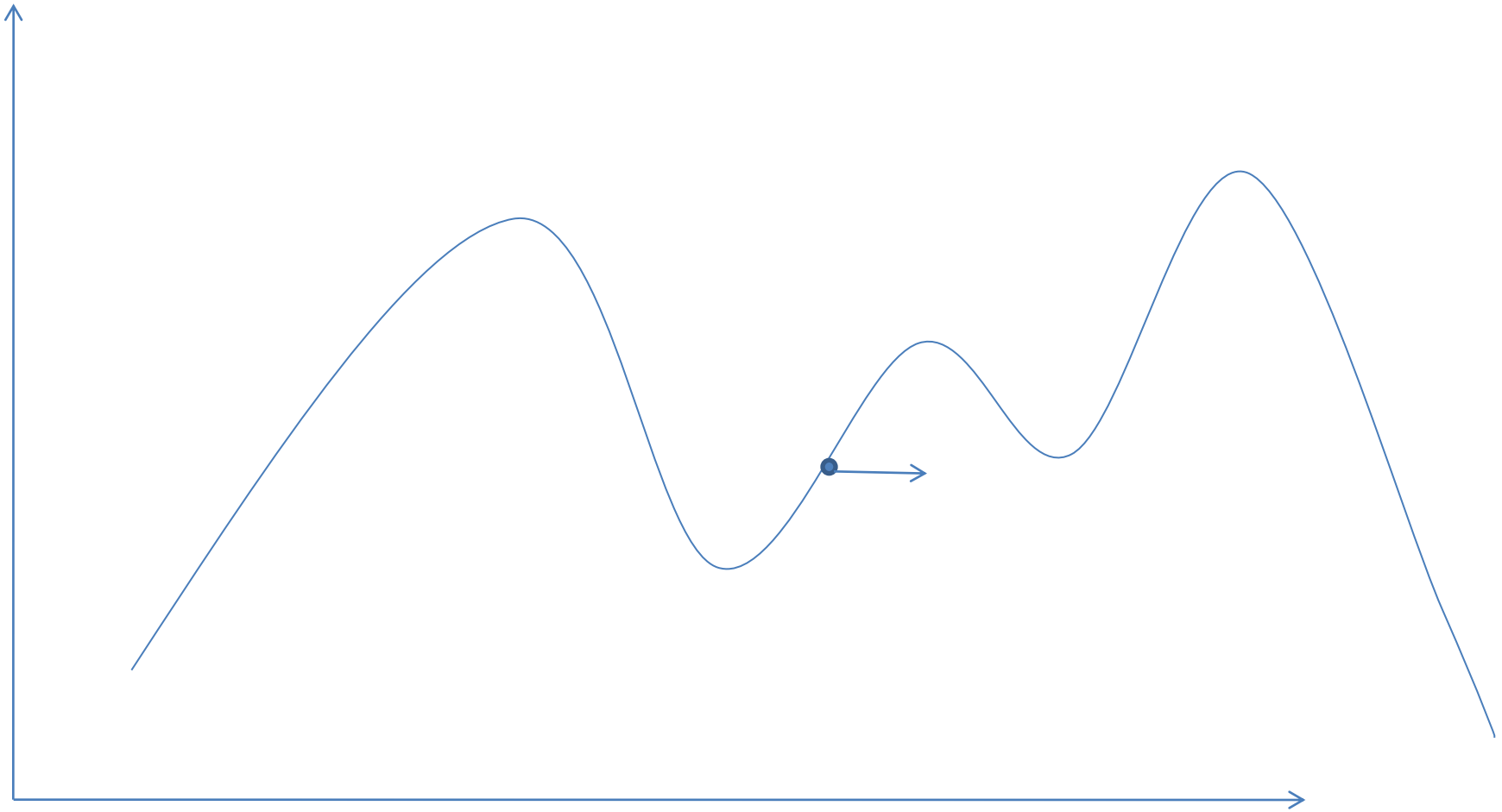
```
repeat
    randomly select a state (random solution);
    repeat
        perform the best step;
    until no improvement;
until timeout;
```

Local optimization generalizes greedy search.
Several times repeats greedy search from different random initial states.
+ One of the best search strategies for huge search spaces.

Gradient search



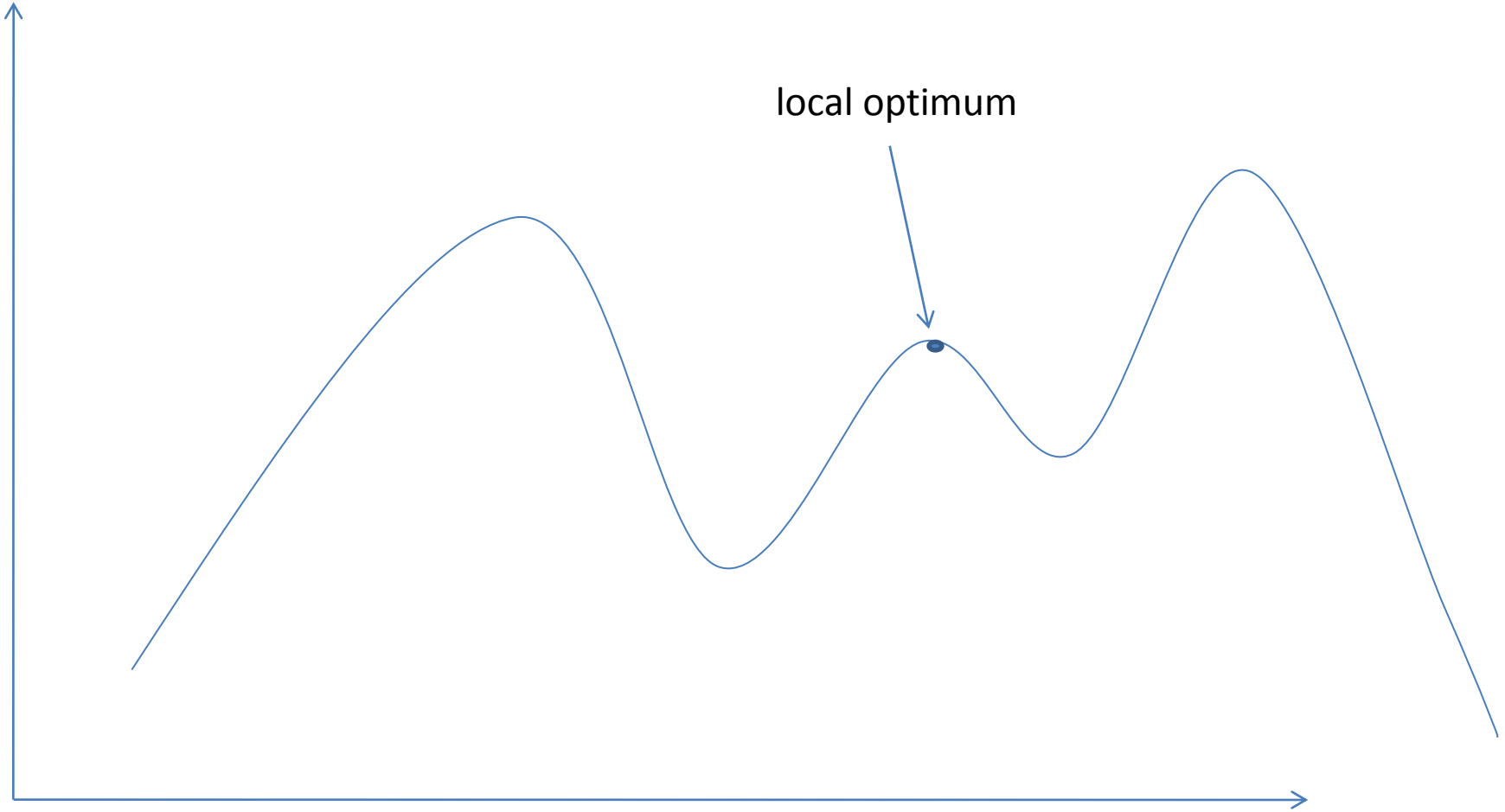
Gradient search



Gradient search

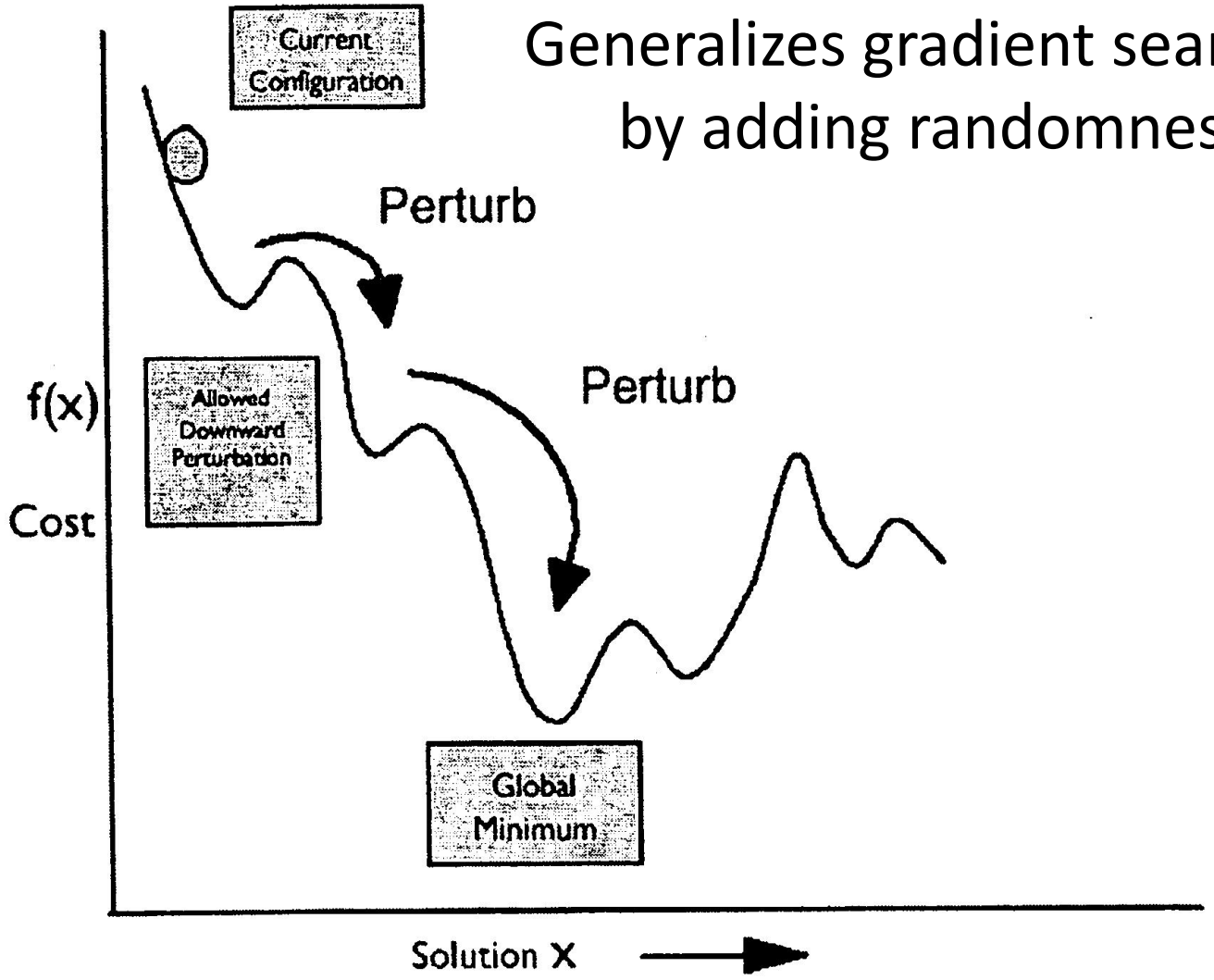
Generalizes greedy search to continuous search spaces.

Again myopia!



Simulated annealing

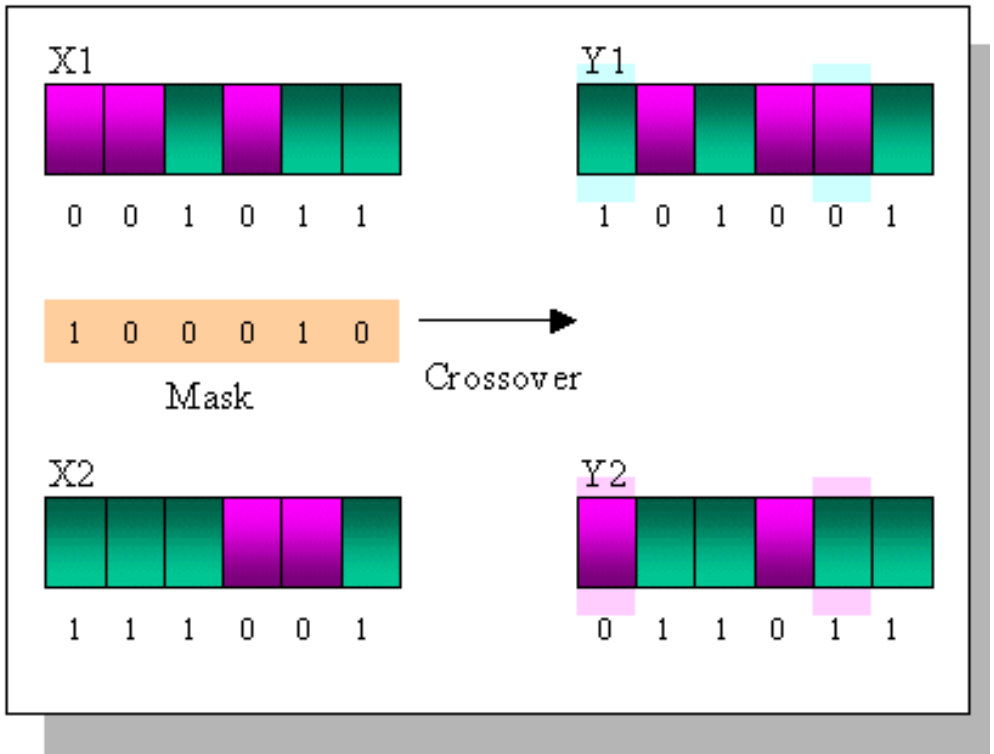
Generalizes gradient search by adding randomness.



Genetic algorithms

Imitate evolution: mating and competing for survival.
Good subjects survive, bad die away. Best have the most successors.

Essential: appropriate coding.



Examples

problem	search strategy
Building decision trees	greedy
Binarization of discrete attribute	greedy /exhaustive
Semi-naive Bayes	bounded exhaustive
learning decision rules	greedy/local opt./genetic alg.
Feedforward neural nets	gradient/simulated annealing
Association rules	bounded exhaustive
Feature subset selection	greedy forward-backward/local opt.

6 Attribute quality measures

1. Impurity function
2. Measures based on information content:
   (a) information gain, gain-ratio, distance-measure
   (b) weight of evidence,
   (c) MDL,
   (d) J-measure,
3. Some other measures:
   (a) χ^2, G, ORT,
   (b) Gini index, ReliefF.

Attributional representation:

- set of attributes A = $\{A_i, i = 0..a\}$;
- for each discrete attribute A_i a set of possible values $\mathcal{V}_i = \{V_1, ...V_{n_i}\}$
- for each continuous attribute A_i an interval of possible values

$$\mathcal{V}_i = [Min_i, ...Max_i]$$

- class A_0: for classification discrete; for regression continuous;
- one learning instance is a vector of attribute values

$$u_j =< r^{(j)}, v^{(1,j)}, ..., v^{(a,j)} >$$

  the class value: $r^{(j)} = v^{(0,j)}$;
- a set of learning examples: $\mathcal{U} = \{u_j, j = 1...n\}$

Let us introduce the following notation:

n – the number of learning examples

$n_{k.}$ – the number of learning examples from class r_k

$n_{.j}$ – the number of learning examples with j–th value of attr. A_i

n_{kj} – the number of lear. ex. from class r_k and with j–th value

$p_{kj} = n_{kj}/n_{..}$

$p_{k.} = n_{k.}/n_{..}$

$p_{.j} = n_{.j}/n_{..}$

$p_{k|j} = p_{kj}/p_{.j} = n_{kj}/n_{.j}$

Measures based on information content

The amount of information:

$$I(X_i) = -\log_2 P(X_i)$$

The average expected amount of information – *entropy*:

$$H(X) = -\sum_i P(X_i) \log_2 P(X_i)$$

Entropy is the impurity measure.
Proof:
1. Maximum:

$$H(X) = -\sum_i^{n_0-1} P(X_i) \log_2 P(X_i) - \left(1 - \sum_i^{n_0-1} P(X_i)\right) \log \left(1 - \sum_i^{n_0-1} P(X_i)\right)$$

$$\frac{\partial H}{\partial P(X_k)} = -\log P(X_k) - \log e + \log \left(1 - \sum_i^{n_0-1} P(X_i)\right) + \log e$$

The first derivative is 0 when: $P(X_k) = 1/n_0$
The second derivative is negative (always):

$$\frac{\partial^2 H}{\partial P(X_k)^2} = -\log e \left(\frac{1}{P(X_k)} + \frac{1}{\log\left(1 - \Sigma_i^{n_0-1} P(X_i)\right)}\right) < 0$$

2. Minimum:

$$H(X) \geq 0 \wedge H(X) = 0 \Leftrightarrow (P(X_i) = 0 \vee P(X_i) = 1)$$

3. H is symmetric.
4. H is concave.
5. H is continuous and continuously derivable.

Let us introduce the entropies:

H_R – class entropy: $H_R = -\Sigma_k\, p_{k.} \log p_{k.}$

H_A – attribute entropy: $H_A = -\Sigma_j\, p_{.j} \log p_{.j}$

H_{RA} – entropy of joint event class-attribute value:

$$H_{RA} = -\sum_k \sum_j p_{kj} \log p_{kj}$$

$H_{R|A}$ – conditional (expected) entropy of class, given A:

$$H_{R|A} = H_{RA} - H_A = -\sum_j p_{.j} \sum_k \frac{p_{kj}}{p_{.j}} \log \frac{p_{kj}}{p_{.j}}$$

$$H_{R|A} = -\sum_j p_{.j} \sum_k p_{k|j} \log p_{k|j}$$

Because $H_X \geq 0$, it holds $H_{RA} \geq H_{R|A}$.

Information gain

$$Gain(A) = H_R + H_A - H_{RA} = H_R - H_{R|A}$$

Properties:

$$Gain(A) \geq 0$$

$$max(Gain(A)) = H_R$$

If we split one value of A into two values to gain attribute A', it holds:

$$Gain(A') \geq Gain(A)$$

Gain ratio

$$GainR(A) = \frac{Gain(A)}{H_A}$$

$$Gain(A_i) \geq \frac{\sum_{j=1}^{a} Gain(A_j)}{a}$$

Distance measure

$$1 - D(R, A) = \frac{Gain(A)}{H_{RA}}$$

$D(R, A)$ is a distance:

1. $D(R, A) \geq 0$
2. $D(R, A) = 0 \Leftrightarrow R = A$
3. $D(R, A) = D(A, R)$
4. $D(R, A_1) + D(A_1, A_2) \geq D(R, A_2)$

J–measure

(inappropriate) attribute quality measure:

$$q_1(V_j) = \phi(P(r_1), ..., P(r_{n_0})) - \phi(P(r_1|V_j), ..., P(r_{n_0}|V_j))$$

Several weaknesses:

- it can be negative and
- does not differentiate between the equal and permuted distributions, e.g.:

$$P(r_1|V_j) = P(r_1), ..., P(r_{n_0}|V_j) = P(r_{n_0}) \Rightarrow q_1(V_j) = 0$$

$$P(r_1|V_j) = P(r_{n_0}), ..., P(r_{n_0}|V_j) = P(r_1) \Rightarrow q_1(V_j) = 0$$

J–measure

Information content of the condition is the expected amount of information obtained from V_j.

$$J_j = p_{.j} \sum_k p_{k|j} \log \frac{p_{k|j}}{p_{k.}}$$

J–measure has nice properties:

Non-negativity: $J_j \geq 0$.

Upper bound of J–measure: derivative equals 0 also:

$$k_0 : p_{k_0|j} = 1 \wedge \forall k \neq k_0 : \ p_{k|j} = 0$$

$$J_j \leq p_{.j} \left(\min \left\{ \max_k -\log p_{k.}, \ -\log p_{.j} \right\} \right) \leq -p_{.j} \log p_{.j} \leq 0.53 \ bit$$

Relation with information gain: $\Sigma_j \, J_j = Gain(A)$

Follows the χ^2 distribution: $2nJ_j \log_e 2 = 1.3863 nJ_j$

Gini–index

Prior Gini-index:

$$Gini\_prior = \sum_k \sum_{l \neq k} p_{k.} p_{l.} = 1 - \sum_k p_{k.}^2$$

Prior Gini-index is an impurity measure.
Proof:

1. Maximum:

$$Gini\_prior = 1 - \sum_k^{n_0-1} p_{k.}^2 - \left(1 - \sum_k^{n_0-1} p_{k.}\right)^2$$

$$\frac{\partial Gini\_prior}{\partial p_{k.}} = -2p_{k.} + 2\left(1 - \sum_k^{n_0-1} p_{k.}\right)$$

The first derivative is 0 when: $p_{k.} = 1/n_0$
The second derivative is negative (always):

$$\frac{\partial^2 Gini\_prior}{\partial p_{k.}^2} = -2 + 2(-1) = -4 < 0$$

2. Minimum:

$$Gini\_prior \geq 0 \wedge Gini\_prior = 0 \Leftrightarrow \exists! k : \ p_{k.} = 1$$

3. $Gini\_prior$ is symmetric.
4. $Gini\_prior$ is concave.
5. $Gini\_prior$ is continuous and continuously derivable.

Attribute quality: difference between the prior and expected Gini–index:

$$Gini(A) = \sum_j p_{.j} \sum_k p^2_{k|j} - \sum_k p^2_{k.}$$

$Gini(A)$ has all properties stemming from impurity measure properties:

Non-negativity: $Gini(A) \geq 0$

Maximum:

$$Gini(A) = Gini\_prior \Leftrightarrow \forall j : \ \exists! k : \ p_{k|j} = 1$$

Increasing the number of attribute values: $Gini(A)$ at most increases.

ReliefF

A_1	A_2	A_3	R
0	0	0	0
0	0	1	0
0	1	1	1
0	1	0	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	0	0

```
function RELIEF(I: array[1..n] of instance): array[1..a] of real;
var inst,att : integer; W : array[1..a] of real;
  M,H : instance; (* nearest miss and hit *)
begin
 for att := 1 to a do W[att] := 0.0; end for
 for inst := 1 to n do
   for instance I[inst] find nearest miss M and hit H;
   for att := 1 to a do
     W[att] := W[att] - diff(att,I[inst],H)/n + diff(att,I[inst],M)/n;
    end for;
  end for;
 return(W);
end;
```

$$diff(A_i, u_j, u_k) = \begin{cases} \frac{|v^{(i,j)} - v^{(i,k)}|}{Max_i - Min_i}, & A_i \; is \; continuous \\ 0, & v^{(i,j)} = v^{(i,k)} \wedge A_i \; is \; discrete \\ 1, & v^{(i,j)} \neq v^{(i,k)} \wedge A_i \; is \; discrete \end{cases}$$

Basic RELIEF:

- estimating dicrete and continuous attributes,
- two-class problems,
- complexity: $O(n^2 \times a) \longrightarrow O(n \times m \times a), \quad m \in [30..200]$

Relations with gini-index:

RELIEF evaluates the probability difference:

$$RELIEF(A_i) = P(diff(A_i, \cdot, \cdot) = 1 | nearest\, inst.\, from\, opposite\, class) -$$

$$-P(diff(A_i, \cdot, \cdot) = 1 | nearest\ instance\ from\ same\ class)$$

Omit the nearness condition and we get a similar function to Gini–index:

$$Relief(A_i) = P(diff(A_i, u_j, u_k) = 0 | r^{(j)} = r^{(k)}) -$$

$$-P(diff(A_i, u_j, u_k) = 0 | r^{(j)} \neq r^{(k)}) =$$

$$= constant \times \sum_j p_{.j}^2 \times Gini'(A_i)$$

Proof:

Let:

$$P_{eqval} = P(diff(A_i, u_j, u_k) = 0)$$

$$P_{samecl} = P(r^{(j)} = r^{(k)})$$

$$P_{samecl|eqval} = P(r^{(j)} = r^{(k)} | diff(A_i, u_j, u_k) = 0)$$

By using the Bayes rule, we get:

$$Relief(A_i) = \frac{P_{samecl|eqval} P_{eqval}}{P_{samecl}} - \frac{(1 - P_{samecl|eqval}) P_{eqval}}{1 - P_{samecl}}$$

$$P_{samecl} = \sum_k p_{k.}^2$$

$$P_{samecl|eqval} = \sum_j \left(\frac{p_{.j}^2}{\Sigma_j \, p_{.j}^2} \times \sum_k p_{k|j}^2 \right)$$

$$\begin{aligned} Relief(A_i) &= \frac{P_{eqval} \times Gini'(A_i)}{P_{samecl}(1 - P_{samecl})} \\ &= \frac{\Sigma_j \, p_{.j}^2 \times Gini'(A_i)}{\Sigma_k \, p_{k.}^2 (1 - \Sigma_k \, p_{k.}^2)} \\ &= constant \times \sum_j p_{.j}^2 \times Gini'(A_i) \end{aligned}$$

where $Gini'(A)$ is similar to Gini–index:

$$Gini'(A) = \sum_j \left(\frac{p_{.j}^2}{\Sigma_j \, p_{.j}^2} \times \sum_k p_{k|j}^2 \right) - \sum_k p_{k.}^2 \qquad (1)$$

The only difference: Gini–index has factor:

$$\frac{p_{.j}}{\Sigma_j \, p_{.j}} = p_{.j}$$

ReliefF implements the following extensions:

Unknown attribute values:

- instance (u_j) lacks the attribute value:

$$diff(A_i, u_j, u_k) = 1 - p_{v^{(i,k)}|r^{(j)}}$$

- both instances lack the attribute value:

$$diff(A_i, u_j, u_k) = 1 - \sum_{l=1}^{n_i} (p_{V_l|r^{(j)}} \times p_{V_l|r^{(k)}})$$

Noisy data: ReliefF searches for k nearest hits and k nearest misses and averages their contributions.

Multi-class problems: ReliefF searches for k nearest misses from all classes. Class contributions are weighted with prior class probs.

function ReliefF(I: array[1..n] of instance; k : integer):
array[1..a] of real;
var
inst,att, kk, cl, r, rr : integer;
W : array[1..a] of real;
M : array[1..k,1..n_0] of instance;

```
begin
 for att := 1 to a do W[att] := 0.0; end for;
 for inst := 1 to n do for cl := 1 to n_0 do
   for instance I[inst] find k nearest instances M[kk,cl] from r_cl;
   r := arg_rr r_rr = r^(inst);
   for att := 1 to a do for kk := 1 to k do
    if cl = r then W[att] := W[att] - diff(att,I[inst],M[kk,cl])/(n*k)
     else W[att] := W[att] + p_cl./(1-p_r.) * diff(att,I[inst],M[kk,cl])/(n*k)
     end if;
    end for; end for;
  end for; end for;
 return(W);
end;
```

Measures for estimating attributes in regression

1. Change of variance
2. Regressional ReliefF (RReliefF)
3. MDL

Change of variance

For impurity measure we use *variance* of the continuous target variable:

$$s^2 = \frac{1}{n} \sum_{k=1}^{n} (r^{(k)} - \overline{r})^2$$

$$\overline{r} = \frac{1}{n} \sum_{k=1}^{n} r^{(k)}$$

If in the two-class classification problem one class changes into value 1.0 of continuous target variable, and the other class into value 0.0, it holds:

$$Gini\_prior = 2s^2$$

Proof:

$$Gini\_prior = p_{1.}p_{2.} + p_{2.}p_{1.} = \frac{2n_{1.}n_{2.}}{n^2}$$

$$\overline{r} = \frac{n_{1.} \times 0 + n_{2.} \times 1}{n} = \frac{n_{2.}}{n}$$

$$\begin{aligned}
s^2 &= \frac{1}{n}\sum_{k=1}^{n}(r^{(k)} - \overline{r})^2 \\
&= \frac{1}{n}\left(\sum_{k=1}^{n_{1.}}\left(0 - \frac{n_{2.}}{n}\right)^2 + \sum_{k=1}^{n_{2.}}(1 - \frac{n_{2.}}{n})^2\right) \\
&= \frac{1}{n}\left(n_{1.}\left(\frac{n_{2.}}{n}\right)^2 + n_{2.}\left(\frac{n_{1.}}{n}\right)^2\right) \\
&= \frac{n_{1.}n_{2.}(n_{1.} + n_{2.})}{n^3} \\
&= \frac{n_{1.}n_{2.}}{n^2} \\
&= \frac{Gini_prior}{2}
\end{aligned}$$

The attribute quality is (non-negative)
expected change of variance:

$$ds^2(A_i) = \frac{1}{n}\sum_{k=1}^{n}(r^{(k)} - \overline{r})^2 - \sum_{j=1}^{n_i}\left(p_{.j}\frac{1}{n_{.j}}\sum_{k=1}^{n_{.j}}(r_j^{(k)} - \overline{r}_j)^2\right)$$

$r_j^{(k)}$ – the value of continuous target var. of k-th instance, which has j-th value of attribute A_i

$$\overline{r}_j = \frac{1}{n_{.j}}\sum_{k=1}^{n_{.j}} r_j^{(k)}$$

7 Data preprocessing

1. Representation of complex structures,
2. discretization of continuous attributes,
3. fuzzy discretization,
4. attribute binarization,
5. transforming discrete attributes into continuous,
6. dealing with missing values,
7. visualization,
8. feature subset selection.

Discretization of continuous attributes

Discretization - we typically lose information.

- optimal number of intervals
- optimal boundaries for each interval

Types of discretization

Start with ordered instances according to att's values:

- *bottom-up*
  n instances: $O(n^2)$
- *top-down*
  k intervals: $O(kn)$

Possible boundaries between intervals

$\phi(P(r_1), ..., P(r_{n_0}))$ - impurity measure
Quality of attribute A given one boundary:

$$q(A) = \phi(P(r_1), ..., P(r_{n_0})) - \sum_{j=1}^{2} P(V_j)\phi(P(r_1|V_j), ..., P(r_{n_0}|V_j))$$

Let Max consecutive instances be from the same class.
Boundary belongs to a set: $x \in [0, Max]$.
It can be shown that $q(x)$ is convex:

$$\frac{\partial^2 q}{\partial x^2} > 0$$

maximum is reached at a boundary point: $x = 0$ OR $x = Max$.

Proof:

$$\frac{\partial q}{\partial x} = -\frac{\phi(p_{1|1},...,p_{n_0-1|1})}{n} + \frac{n_{.1}+x}{n(n_{.1}+x)^2}\sum_{k=1}^{n_0-1} n_{k1}\frac{\partial\phi(p_{1|1},...,p_{n_0-1|1})}{\partial p_{k|1}} +$$

$$\frac{\phi(p_{1|2},...,p_{n_0-1|2})}{n} - \frac{n-n_{.1}-x}{n(n-n_{.1}-x)^2}\sum_{k=1}^{n_0-1} n_{k2}\frac{\partial\phi(p_{1|2},...,p_{n_0-1|2})}{\partial p_{k|2}}$$

$$\frac{\partial q}{\partial x} = -\frac{\phi(p_{1|1},...,p_{n_0-1|1})}{n} + \frac{1}{n(n_{.1}+x)}\sum_{k=1}^{n_0-1} n_{k1}\frac{\partial\phi(p_{1|1},...,p_{n_0-1|1})}{\partial p_{k|1}} +$$

$$\frac{\phi(p_{1|2},...,p_{n_0-1|2})}{n} - \frac{1}{n(n-n_{.1}-x)}\sum_{k=1}^{n_0-1} n_{k2}\frac{\partial\phi(p_{1|2},...,p_{n_0-1|2})}{\partial p_{k|2}}$$

$$\frac{\partial^2 q}{\partial x^2} = \frac{2-2}{n(n_{.1}+x)^2}\sum_{k=1}^{n_0-1} n_{k1}\frac{\partial\phi(p_{1|1},...,p_{n_0-1|1})}{\partial p_{k|1}} -$$

$$-\frac{1}{n(n_{.1}+x)^3}\sum_{k=1}^{n_0-1} n_{k1}^2\frac{\partial^2\phi(p_{1|1},...,p_{n_0-1|1})}{\partial p_{k|1}^2} +$$

$$+\frac{2-2}{n(n-n_{.1}-x)^2}\sum_{k=1}^{n_0-1} n_{k2}\frac{\partial\phi(p_{1|2},...,p_{n_0-1|2})}{\partial p_{k|2}} -$$

$$-\frac{1}{n(n-n_{.1}-x)^3}\sum_{k=1}^{n_0-1} n_{k2}^2\frac{\partial^2\phi(p_{1|2},...,p_{n_0-1|2})}{\partial p_{k|2}^2}$$

$$\frac{\partial^2 q}{\partial x^2} = -\frac{1}{n(n_{.1}+x)^3}\sum_{k=1}^{n_0-1} n_{k1}^2\frac{\partial^2\phi(p_{1|1},...,p_{n_0-1|1})}{\partial p_{k|1}^2} -$$

$$-\frac{1}{n(n-n_{.1}-x)^3}\sum_{k=1}^{n_0-1} n_{k2}^2\frac{\partial^2\phi(p_{1|2},...,p_{n_0-1|2})}{\partial p_{k|2}^2}$$

Therefore, function q is convex:

$$\frac{\partial^2 q}{\partial x^2} > 0$$

and the maximum is reached at one of the boundary points:
$x = 0$ OR $x = Max$.

Measures for controlling discretization

The expected impurity q with a new boundary may only decrease.

Information gain and *Gini-index* are not appropriate.

Use measures which do not overestimate multivalued attributes:

- $1 - D$
- MDL
- $ReliefF$

Fuzzy discretization

Boundary between two intervals = b. between two decisions.

intervals [0.00..0,30] and [0.31..1.00]:
- instances 0.00 and 0.30 are considered equivalent
- instances 0.30 and 0.31 are considered different

Fuzzy boundary:
- probability distribution over intervals
- fuzzy values of the continuous attribute

Attribute binarization

binary decision trees:

- smaller and more optimal:
  - decreasing the replication problem
  - reduces the shattering of data into small subsets
- overcomes the overestimating of multivalued attributes,
- some measures applicable only for binary attributes (ORT).

For binarization of **continuous attributes**
- use the first step of top-down discretization

Binarization of **discrete attribute**:

- For each value of attribute A generate one binary attribute by joining all other values
  OR
- Generate all possible binary splits into two disjunct subsets of values:

$$\frac{1}{2}\sum_{k=1}^{n-1}\binom{n}{k} = 2^{n-1} - 1$$

Two-class problems

Order the values $V_j, j = 1..n$ of the attribute according to increasing conditional probabilities of the first class:

$$P(r_1|V_1) \leq P(r_1|V_2) \leq ... \leq P(r_1|V_n)$$

The quality of the binary version of A:

$$q(A) = \phi(P(r_1), ..., P(r_{n_0})) - \sum_{j=1}^{2} P(\mathcal{V}_j)\phi(P(r_1|\mathcal{V}_j), ..., P(r_{n_0}|\mathcal{V}_j))$$

is the largest for subsets $\mathcal{V}_j, j = 1, 2$, where:

$$\mathcal{V}_1 = \{V_1, ..., V_l\}, \quad \mathcal{V}_2 = \{V_{l+1}, ..., V_n\}$$

for some $0 < l < n$.

Multi-class problems

Greedy search for (suboptimal) binarization:

1. Select the value that maximizes the quality of binary version where all other values are joint.
2. Repeat until the quality stops improving:
   - If there exists a value that, when moved to another subset, improves the attribute's quality, then move it (or move the one that maximizes the quality).

Transforming discrete attributes into continuous

linear and other regressions, discriminant functions and neural networks assume: **all attributes are continuous**.

Binary attribute – special case of the continuous attribute.

Multi-valued discrete attribute is replaced with one binary attribute for each value.

Disadvantage: a lot of mutually dependent attributes.

Dealing with missing values

If the value of attribute A_i is missing, we can either:

- ignore it,
- add to attribute a special (unknown) value,
- supplant it by the most probable attribute value, or
- treat it with a probability distribution of attribute values.

- conditional probabilities given the class;
- machine learning algorithms can learn to predict the missing attribute value from all other attributes and the class label.

Dealing with special values

- "don't care"
- "inapplicable"

Naive Bayesian classifier

conditional independence of attributes given the class:

$$P(r_k|V) = P(r_k) \prod_{i=1}^{a} \frac{P(r_k|v_i)}{P(r_k)}$$

prior probabilities: Laplace law of succession:

$$P(r_k) = \frac{N_k + 1}{N + n_0}$$

conditional probabilities: m-estimate:

$$P(r_k|v_i) = \frac{N_{k,i} + mP(r_k)}{N_i + m}$$

NB properties

- always uses **all available attributes**
- **missing values**: simply excludes instances
- **continuous attributes** (*fuzzy*) discretization in advance
- conditional independence often an acceptable assumption:
  - Patient symptomps depend on a disease
  - LCD display: errors of LCD tubes are independent
- **Probability estimation** quite reliable:
  no overfitting!

NB properties

- If independence is not perfect:
  still enough "space for mistake" (the independence assumption does not spoil the order of class probabilites).
- **Strong dependencies** among attributes: NB fails
- **Explaining decisions of** NB:

  $$-\log_2 P(r_k|V) = -\log_2 P(r_k) - \sum_{i=1}^{a} (\log_2 P(r_k|v_i) - \log_2 P(r_k))$$

  prior needed information to classify into class r_k, minus the sum of information gains of attributes.
- **Incremental learning**

10 Numerical methods

1. k-nearest neigbors (k-NN),
2. discriminant functions,
3. linear regression,
4. support vector machines (SVM)

Nearest neighbors

Learning is simply memorizing training instances - lazy learning.

Prediction: find a subset of *similar* instances.

1. k-nearest neighbors
2. distance-weighted k-nearest neighbors
3. locally weighted regression.

K-nearest neighbors

for new instance u_x, find k nearest u_1, ..., u_k

Classify into the majority class:

$$r_x = argmax_{r \in \{V_1, ..., V_{n_0}\}} \sum_{i=1}^{k} \delta(r, r^{(i)})$$

where

$$\delta(a, b) = \begin{cases} 1, & a = b \\ 0, & a \neq b \end{cases}$$

For regression average the target values:

$$r_x = \frac{1}{k} \sum_{i=1}^{k} r^{(i)}$$

Parameter k

k usually set to an even number

For noiseless data: 1-NN.

By averaging k predictions we fight against noise.

Optimal k: find experimentally

diameter of the neighborhood changes dynamically

distance measure:

k-NN is sensitive to the chosen distance measure.

Euclidean distance metric: $D(u_l, u_j) = \sqrt{\Sigma_{i=1}^{a} d(v^{(i,l)}, v^{(i,j)})^2}$

where for continuous attribute A_i holds:

$$d(v^{(i,l)}, v^{(i,j)}) = |v^{(i,l)} - v^{(i,j)}|$$

and for discrete attribute:

$$d(v^{(i,l)}, v^{(i,j)}) = \begin{cases} 0, & v^{(i,l)} = v^{(i,j)} \\ 1, & v^{(i,l)} \neq v^{(i,j)} \end{cases}$$

the weighted distance is calculated by:

$$D(u_l, u_j) = \sqrt{\sum_{i=1}^{a} q(A_i) d(v^{(i,l)}, v^{(i,j)})^2}$$

Weighted k-nearest neighbors

The impact of the distance can be linear, polynomial, exponential...

For quadratic function, we get for classification problems:

$$r_x = argmax_{r \in \{V_1, \ldots, V_{n_0}\}} \sum_{i=1}^{k} \frac{\delta(r, r^{(i)})}{D(u_x, u_i)^2}$$

and for regression problems:

$$r_x = \frac{\Sigma_{i=1}^{k} \left[r^{(i)} / D(u_x, u_i)^2 \right]}{\Sigma_{i=1}^{k} \left[1 / D(u_x, u_i)^2 \right]}$$

k is redundant: all instances influence the prediction.

Locally weighted regression

Instead of averaging: an arbitrary regression function of k nearest neighbors is used.

linear locally weighted regression is most frequently used, due to:

- danger of overfitting;
- time complexity!

11 Artificial neural networks

Intelligence emerges from the interaction of large numbers of simple processing units.

David Rumelhart & John McClelland

1. Uvod
   (a) A simple example NN
   (b) Distributed memory
   (c) Properties of NN
   (d) Applications
   (e) Relation with symbolic AI

2. Types of NN
   (a) Topology
   (b) Purpose
   (c) Learning rule
   (d) Combination function
3. Hopfield NN
4. Bayesean NN
5. Perceptron
   (a) Two-layered perceptron
   (b) Multilayered perceptron

A simple example NN

a simple matrix calculus:
Task: recognize the following two patterns:

$$X_1 = (1, 1, 1)^T$$

and

$$X_2 = (1, -1, -1)^T$$

Let us construct a (memory) matrix:

$$M = X_1 X_1^T + X_2 X_2^T = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix}$$

Let us introduce a decision (threshold) function:

$$f(X) = \begin{cases} 1, & X > 0 \\ 0, & X = 0 \\ -1, & X < 0 \end{cases}$$

It holds:

$$f(MX_1) = f((2,4,4)^T) = (1,1,1)^T = X_1$$

$$f(MX_2) = f((2,-4,-4)^T) = (1,-1,-1)^T = X_2$$

Let us try with partially known vectors:

$$X_1' = (1, 1, 0)^T$$

$$X_2' = (1, 0, -1)^T$$

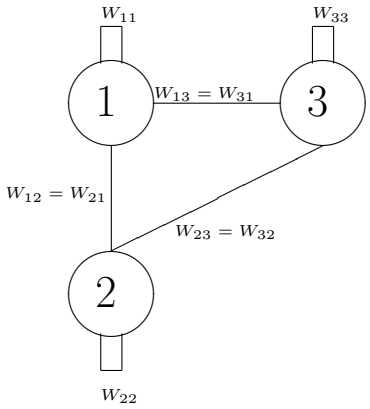
$$f(MX_1') = f((2, 2, 2)^T) = (1, 1, 1)^T = X_1$$

$$f(MX_2') = f((2, -2, -2)^T) = (1, -1, -1)^T = X_2$$

Let us try also with a wrong value:

$$X_1'' = (1, 1, -1)^T = X_2''$$

$$f(MX_1'') = f((2, 0, 0)^T) = (1, 0, 0)^T$$



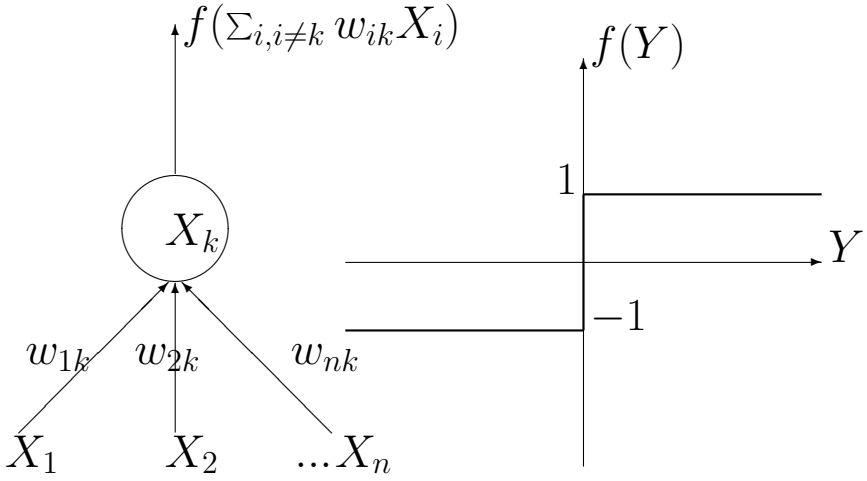
Two possible neuron states, 1 or –1
Each neuron is connected to all the neurons.
The bidirectional connections – synapses are associated with w.

Initially, all the weights are set to zero.
For each training example:
    IF equal values of two neurons
      increment the weight
      ELSE decrement the weight

Processing: neurons calculate their output:

$$Y_i = f(\sum_j W_{ji} X_j)$$

X_j – current state of the j^{th} neuron,
W_{ji} – weight synapse between j^{th} and i^{th} neuron
Y_i – new state of the i^{th} neuron
f – output function



- (a)synchronous processing
- convergence of learning/processing
- synapse may be excitatory/inhibitory

Distributed memory

Properties:

- Automatic generalization
- Adaptability to changing environment
- increasing distributivity = improving the storage accuracy
- Constructive character
- Robustness
- Spontaneous recollection of forgotten data
- content addressable memory: datum = address

Properties of ANN

- Similarity to biological systems
- Parallelism
- Multidirectional execution
- Robustness
  - Neuron damage nevronov
  - synapse damage
  - incomplete input data
- Synapse represents a relation between activity of two neurons

- Relation between hardware and software:
  - no software
  - "relaxator" instead of "computer"
  - hardware can be static or dynamic,
    topology of a network can be randomly generated.
- Mathematical foundation: linear algebra
- biggest **deficiency**: No explanation of decisions

Application examples

- quality control in various production processes,
- medical diagnosis from various measurements/images,
- recognition and classification of images,
- recognition of manually written text and signatures,
- identification of objects with sonar,
- recognition of voice and speech,
- computer vision
- etc

Analogy with brain

- The brain is content-addressable
- not possible to separate the memory function from processors
- The memory in cortex is distributed
- synaptic connections contribute significantly to memory
- The speed of neurons is measured in milliseconds ... Perception, language processing, intuitive thinking etc require about 100 milliseconds – about 100 steps.
- performance gradually decreases with destroyed neurons

- Human perception is to a certain extent invariant to height, translation, and rotation.
- automatically generate a reduced representation
- preserving the topology of spatial signals (self-organization).
- 10^{10} neurons 10^4 connections per neuron: enough capacity to memorize every experience we can encounter in 100 years
- genetic code cannot determine all the connections in the brain no hardware in the strict sense of the word, nor software
- recursive statements are hard to understand by humans.

Differences:

- synapse can be excitatory XOR inhibitory
- the size of the signal is given with the impulse frequency
- ANN models do not use global communication.

Relation to symbolic AI methods

Symbolic Artificial Intelligence	Artificial Neural Networks
symbolic level	subsymbolic level
transparent decisions	only sometimes
explicitly stored rules	dynamically created rules
sequential processing	parallel processing
logic, conscious level	intuitive, subconscious level
psychological analogy	also
no similarity to biological systems	similarity to biological systems
left hemisphere	right hemisphere

TYPES OF ANN

ANN can be classified by various criteria:

- topology of ANN,
- application purpose,
- learning rule, and
- combination function.

Combination function:

Weighted sum:

$$A(X_j) = \sum_i W_{ij} X_i + C_j$$

sigma-pi **function**:

$$sp(X_1, ..., X_n) = \sum_{S_i \in P} W_i \prod_{k \in S_i} X_k$$

P is a powerset of the set of all indices of neurons

Naive Bayes

Output function: – deterministic binary,

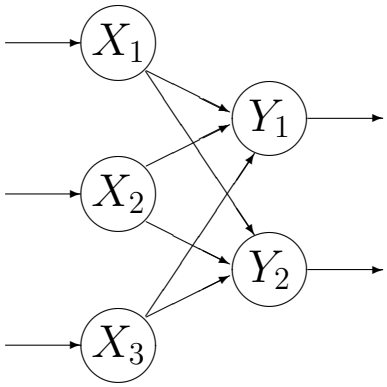
– deterministic continuous:

$$f(X) = \frac{1}{1 + e^{-X}}$$

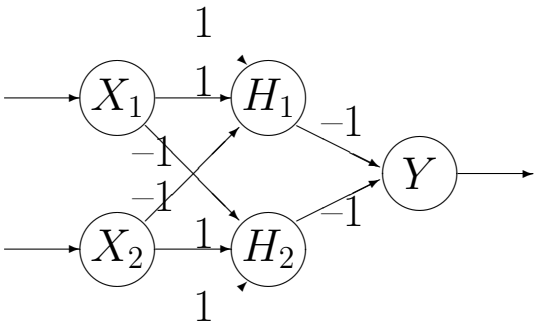
– stochastic.

Topology

- without layers
- Two-layered feedforward ANN:



- Multi-layered feedforward ANN:



X_1	X_2	Y
1	1	−1
1	−1	1
−1	1	1
−1	−1	−1

Application purpose

- Auto-associative memory
- Hetero-associative memory
- Temporal associative memory
- Classification and regression
- clustering
- Self-organization

Learning rule

- basic Hebbian learning rule
- generalized Hebbian learning
- Delta learning rule
- generalized delta rule – backpropagation of errors
- Competitive learning rule
- Forgetting

Perceptron

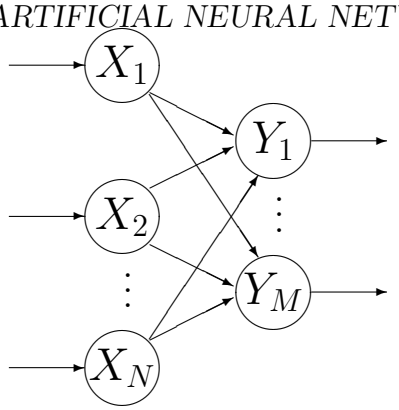
Two-layered perceptron

Two layered feedforward network:

$$Y = WX$$

$$Y_i = \sum_{j=1}^{N} W_{ji} X_j + \theta_i$$

Can solve only linear problems!



Discussion can be limited to a single output neuron.

$$Y = W^T X$$

$Y \geq 0$, then classifies to the first class
$Y < 0$, then classifies to the second class

Learning

1. Initialization: random weights (or set to 0).
2. Present the n-th learning example $X(n)$ to the network.
3. In one step the output is calculated.
4. If correctly classified, the weights are not modified,

otherwise the weights are modified:

4.1. If the correct class is the first one and $W^T(n)X(n) < 0$:

$$W(n+1) = W(n) + \eta X(n)$$

4.2. If the correct class is the second one and $W^T(n)X(n) \geq 0$:

$$W(n+1) = W(n) - \eta X(n)$$

η – *learning rate*

Delta learning rule

Takes into account the difference between Y and desired d:

$$W(n+1) = W(n) + \eta\Big(d(n) - Y(n)\Big)X(n)$$

$$W(n+1) = W(n) + \eta\Big(d(n) - W^T(n)X(n)\Big)X(n)$$

The delta rule uses a *gradient search.*

Squared error $E(n)$ is given with:

$$E(n) = \Big(d(n) - W^T(n)X(n)\Big)^2$$

and its derivative is:

$$\frac{dE(n)}{dW(n)} = -2\Big(d(n) - W^T(n)X(n)\Big)X(n)$$

Batch delta learning rule:
Calculates the difference for all learning examples, $X(1)$, ..., $X(m)$.
The error is: $E(n) = \Sigma_{i=1}^{m} \Big(d(i) - W^T(n)X(i)\Big)^2$
its derivative is:

$$\frac{dE(n)}{dW(n)} = -2 \sum_{i=1}^{m} \Big(d(i) - W^T(n)X(i)\Big)X(i)$$

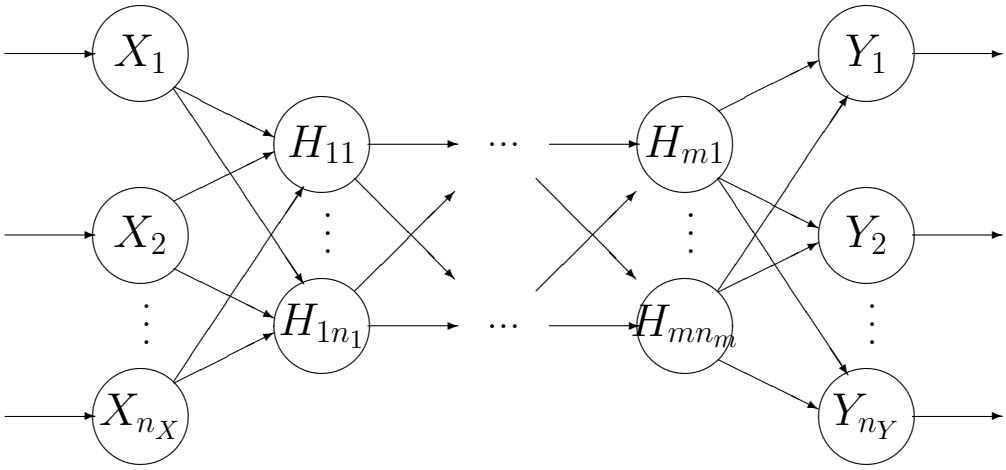
Batch learning rule:

$$W(n+1) = W(n)+\eta\frac{dE(n)}{dW(n)} = W(n)+\eta \sum_{i=1}^{m} \Big(d(i)-W^T(n)X(i)\Big)X(i)$$

Takes into account more information at once.
Not appropriate for an implementation with a NN hardware.

Multi-layered perceptron

Can calculate any nonlinear function.



generalized delta learning rule **or** *backpropagation of error*

1. We start with random weights.
2. Example sets input neurons and NN calculates its output.
3. The difference of actual from the target output is calculated.
4. The modification starts with weights of the output layer.

5.1 Then the error is propagated back to the last hidden layer.

5.2 Its weights are appropriately modified.

5.3 Continues backwards until it modifies the weights of the first hidden layer.

Calculating the output (forward propagation)

The output function: continuous and continuously derivable:

$$f(X) = \frac{1}{1 + e^{-X}}$$

Its derivative is:

$$f'(X) = \frac{e^{-X}}{(1 + e^{-X})^2} = f(X)(1 - f(X))$$

A neuron from the first hidden layer calculates its output for a given learning example $X(n)$ as follows:

$$H_{1i}(n) = f\Big(\sum_{j=1}^{n_X} W_{ji}^{(1)}(n) X_j(n) \Big)$$

Let: $A_{1i}(n) = \Sigma_{j=1}^{n_X} W_{ji}^{(1)}(n) X_j(n)$
On the k-hidden layer, $k > 1$, neurons calculate their outputs:

$$H_{ki}(n) = f(A_{ki}(n))$$

where

$$A_{ki}(n) = \sum_{j=1}^{n_k} W_{ji}^{(k)}(n) H_{k-1,j}(n)$$

The output neurons calculate their outputs:

$$Y_i(n) = f(A_i(n))$$

where

$$A_i(n) = \sum_{j=1}^{n_m} W_{ji}^{(m+1)}(n) H_{m,j}(n)$$

Update of output neurons:

The error of the i-th output neuron for the n-th learning example:

$$e_i(n) = d(n) - Y_i(n)$$

The error of the whole network for that instance:

$$E(n) = \frac{1}{2} \sum_{i=1}^{n_Y} e_i^2(n)$$

Derivative with respect to weights $W_{ji}^{(m+1)}$ uses the chain rule:

$$\frac{\partial E(n)}{\partial W_{ji}^{(m+1)}(n)} = \frac{\partial E(n)}{\partial e_i(n)} \frac{\partial e_i(n)}{\partial Y_i(n)} \frac{\partial Y_i(n)}{\partial A_i(n)} \frac{\partial A_i(n)}{\partial W_{ji}^{(m+1)}(n)}$$

Particular partial derivatives are:

$$\frac{\partial E(n)}{\partial e_i(n)} = e_i(n)$$

$$\frac{\partial e_i(n)}{\partial Y_i(n)} = -1$$

$$\frac{\partial Y_i(n)}{\partial A_i(n)} = f'(A_i(n))$$

$$\frac{\partial A_i(n)}{\partial W_{ji}^{(m+1)}(n)} = H_{mj}(n)$$

The rule for updating the weights $W^{(m+1)}$ of output neurons:

$$W_{ji}^{(m+1)}(n+1) = W_{ji}^{(m+1)}(n) - \eta \frac{\partial E(n)}{\partial W_{ji}^{(m+1)}(n)}$$

$$W_{ji}^{(m+1)}(n+1) = W_{ji}^{(m+1)}(n) - \eta \frac{\partial E(n)}{\partial A_i(n)} H_{mj}(n)$$

$$W_{ji}^{(m+1)}(n+1) = W_{ji}^{(m+1)}(n) - \eta e_i(n) f'(A_i(n)) H_{mj}(n)$$

Back-propagating of errors of the output neurons:
we calculate partial derivatives of error with respect to the activation levels:

$$\frac{\partial E(n)}{\partial A_{mi}(n)} = \frac{\partial E(n)}{\partial H_{mi}(n)} \frac{\partial H_{mi}(n)}{\partial A_{mi}(n)}$$

$$\frac{\partial H_{mi}(n)}{\partial A_{mi}(n)} = f'(A_{mi}(n))$$

For the first factor we use the chain rule:

$$\frac{\partial E(n)}{\partial H_{mi}(n)} = \sum_{j=1}^{n_Y} \frac{\partial E(n)}{\partial A_j(n)} \frac{\partial A_j(n)}{\partial H_{mi}(n)}$$

$$\frac{\partial E(n)}{\partial H_{mi}(n)} = \sum_{j=1}^{n_Y} \frac{\partial E(n)}{\partial A_j(n)} \frac{\partial \left(\sum_{l=1}^{n_m} W_{lj}^{(m+1)}(n) H_{m,l}(n) \right)}{\partial H_{mi}(n)}$$

$$\frac{\partial E(n)}{\partial H_{mi}(n)} = \sum_{j=1}^{n_Y} \frac{\partial E(n)}{\partial A_j(n)} W_{ij}^{(m+1)}(n)$$

Therefore for the last (m-th) hidden layer we get:

$$\frac{\partial E(n)}{\partial A_{mi}(n)} = f'(A_{mi}(n)) \sum_{j=1}^{n_Y} \frac{\partial E(n)}{\partial A_j(n)} W_{ij}^{(m+1)}(n)$$

$$\frac{\partial A_{mi}(n)}{\partial W_{ji}^{(m)}(n)} = H_{m-1,j}(n)$$

The updating rule for the last hidden layer is:

$$W_{ji}^{(m)}(n+1) = W_{ji}^{(m)}(n) - \eta \frac{\partial E(n)}{\partial W_{ji}^{(m)}(n)}$$

$$W_{ji}^{(m)}(n+1) = W_{ji}^{(m)}(n) - \eta \frac{\partial E(n)}{\partial A_{mi}(n)} H_{m-1,j}(n)$$

$$W_{ji}^{(m)}(n+1) = W_{ji}^{(m)}(n) - \eta f'(A_{mi}(n)) \left(\sum_{l=1}^{n_Y} \frac{\partial E(n)}{\partial A_l(n)} W_{il}^{(m+1)}(n) \right) H_{m-1,j}(n)$$

Other hidden layers:

For k-th hidden layer,$1 < k < m$, it holds:

$$\frac{\partial A_{ki}(n)}{\partial W_{ji}^{(k)}(n)} = H_{k-1,j}(n)$$

Therefore the updating rule is:

$$W_{ji}^{(k)}(n+1) = W_{ji}^{(k)}(n) - \eta f'(A_{ki}(n)) \left(\sum_{l=1}^{n_{k+1}} \frac{\partial E(n)}{\partial A_{k+1,l}(n)} W_{il}^{(k+1)}(n) \right) H_{k-1,j}(n)$$

For the first hidden layer ($k = 1$) it holds:

$$\frac{\partial A_{1i}(n)}{\partial W_{ji}^{(1)}(n)} = X_j(n)$$

Therefore the updating rule is:

$$W_{ji}^{(1)}(n+1) = W_{ji}^{(1)}(n) - \eta f'(A_{1i}(n)) \left(\sum_{l=1}^{n_2} \frac{\partial E(n)}{\partial A_{2,l}(n)} W_{il}^{(2)}(n) \right) X_j(n)$$

Problems with backpropagation

1. The learning process does not always converge to an optimal network (it can get stuck into a local optimum).
2. The selection of an appropriate topology.
3. Danger of overfitting.
4. The learning rate η influences the convergence of learning.
5. Requires a large number of passes through the learning set.
6. The generalized delta learning rule has no biological analogy.

The first three problems can be partially solved...

1. Let us introduce a momentum term for persistence:

$$W_{ji}(n+1) = W_{ji}(n) - \Delta W_{ji}(n+1) - \alpha \Delta W_{ji}(n)$$

2. The weight elimination method adds a term that penalizes large weights.
   - The learning starts with an overly large network and the learning process itself eliminates redundant neurons.
   - By automatically finding an appropriate topology the network avoids overfitting.
   - However, the method introduces additional parameters which cannot be easily set.