Neuronale Netze (SS 2002), 17.4.

Perceptron for non linearly separable data:

- Cycle theorem: If the perceptron is trained with the perceptron learning algorithm on a pattern set with integer numbers, the following holds: in a finite number of time steps a solution is found or a cycle is reached.

- Alternative perceptron algorithm:

  $\vec{w} := \mathbf{0}$; $\vec{w}^* := \vec{w}$;
  WHILE $\vec{x}$ exists with $\delta(\vec{w}, \vec{x}) \neq \emptyset$ and no cycle is reached
  $\quad$ $\vec{w} := \vec{w} + \delta(\vec{w}, \vec{x})\vec{x}$;
  $\quad$ if $\vec{w}$ better than $\vec{w}^*$ then $\vec{w}^* := \vec{w}$;

  not very efficient!!

- Alternative: Pocket algorithm:

  $\vec{w} := \mathbf{0}$; $l := 0$;
  $P_0 := P$;
  $\vec{w}^* := \vec{w}$; $l^* := 0$;
  WHILE $(P_0 \neq \emptyset$ und ich habe noch Geduld $)$ DO
  $\quad$ Wähle $\vec{x} \in P_0$; $\quad (*)$
  $\quad$ IF $\delta(\vec{w}, \vec{x}) = 0$
  $\quad\quad$ $l := l + 1$; $P_0 := P_0 \backslash \{\vec{x}\}$;
  $\quad\quad$ IF $l > l^*$ THEN
  $\quad\quad\quad$ $\vec{w}^* := \vec{w}$; $l^* := l$;
  $\quad\quad$ END;
  $\quad$ ELSE
  $\quad\quad$ $\vec{w} := \vec{w} + \delta(\vec{w}, \vec{x})\vec{x}$;
  $\quad\quad$ $l := 0$; $P_0 := P$;
  $\quad$ END;
  END;

  Pocket convergence theorem: The pocket algorithm finds an optimum solution after a finite number of time steps if the examples in $(*)$ are chosen at random.

Note: Random selection in $(*)$ is essential!

- The pocket algorithm may take a long time (just as the perceptron algorithm).
- Perceptron training in the presence of errors is difficult in principle for every possible algorithm – NP-complete problem!

Hierarchy of problems:

- ∞: the real problems of life which cannot be formalized or tackled within computer science
- recursively enumerable problems: decision problems such that a program exists which answers 'yes', but may not terminate for 'no'.

  E.g.: decide whether a program terminates.
- decidable problems: decision problems which can be decided by a computer (in arbitrary time).

  E.g.: decide whether a program terminates in at most 2^n steps.
- NP (nondeterministic polynomial): problems which could efficiently be decided with some (nondeterministic) help.

  E.g.: Traveling salesperson problem (does there exist a tour of at most a given length – if the tour was known, it could be tested efficiently that the tour will do)
  SAT (does there exist a solution for a Boolean formula in conjunctive normalform – if a solution was known, it could be tested efficiently that the formula is valid)
- P: problems which can be solved in polynomial time, e.g. sorting
- Among NP: NP-complete problems, which are the hardest problems in NP. Could an NP complete problem be solved efficienly then all problems could be solved efficiently in NP.

Theorem of Cook: SAT is NP-complete.

The standard way of proving NP-completeness is via **reduction** of a known NP-complete problem:

- You know: the problem class A of decision problems is NP-complete.
- You would like to show: the problem class B of decision problems is NP-complete.
- You show: for every instance a in A an instance b in B can efficiently be constructed such that a is solvable if and only if b is solvable.

  (If B was not NP complete, we could then decide every problem in A just computing b from a and deciding b instead of a!)

Hitting set problem: Given a set $S = \{s_1, \ldots, s_n\}$, a set of subsets $C = \{c_1, \ldots, c_m\}$ with $c_i \subset S$, $k \in \mathbb{N}$, does there exist k points in S such that every c_i is hit by at least one of the k points

(In real life: Find a fixed number of representatives such that each group of interest is covered.)