Artificial Intelligence II
Summary

Thilo Kratzer

July 16, 2018

DISCLAIMER: THIS IS NOT A COMPLETE SUMMARY. ERRORS ARE LIKELY. THE SOLE SOURCE ARE THE LECTURE SLIDES OF THE LECTURE "KÜNSTLICHE INTELLIGENZ II" OF THE SUMMER TERM 2018 AT THE FRIEDRICH-ALEXANDER UNIVERSITY.

1 Quantifying Uncertainty

Definition 1 (Agent)**.** An agent is an entity that perceives and acts. It is modeled as a function from percept histories to actions.

$$\mathcal{P}^* \mapsto \mathcal{A}$$

An agent is anything that perceives its environment via sensors and acts on it with actuators.

Definition 2 (World Model)**.** A stateful reflex agent has a world model consisting of

belief state that has information about the possible states the world may be in

transition model that updates the belief state based on sensor information and actions

Definition 3 (Probability Theory)**.** A probability theory is an assertion language for talking about possible worlds and an inference method for quantifying the degree of belief in such assertions.

Definition 4 (Probabiltiy Model)**.** A probabiltiy model $\langle \Pi, P \rangle$ consists of a set Π of possible worlds called the sample space and a probability function $P : \Omega \to R$, such that $0 \leq P(\omega) \leq 1$ for all $\omega \in \Omega$ and $\sum_{\omega \in \Omega} P(\omega) = 1$.

Definition 5 (Random Variable)**.** A random variable (also called random quantity, aleatory variable, or stochastic variable) is a variable quantity whose value depends on possible outcomes of unknown variables and processes we do not understand.

Definition 6 (Probability)**.** Given a random variable X, $P(X = x)$ denotes the prior probability, or unconditional probability, that X has value x in the absence of any other information.

Definition 7 (Event)**.** We will refer to the fact $X = x$ as an event, or an outcome. The notation uppercase "X" for a variable, and lowercase "x" for one of its values will be used frequently.

Definition 8 (probability distribution)**.** The probability distribution for a random variable X, written $\mathbf{P}(X)$, is the vector of probabilities for the (ordered) domain of X.

Definition 9 (Joint Probability Distribution)**.** Given a subset $\mathbf{Z} \subseteq \{X_1, \ldots, X_n\}$ of random variables, an event is an assignment of values to the variables in $\mathbf{Z}$. The joint probability distribution, written $\mathbf{P(Z)}$, lists the probabilities of all events.

Definition 10 (Atomic Event)**.** Given random variables $\{X_1, \ldots, X_n\}$, an atomic event is an assignment of values to all variables.

Definition 11 (Full Joint Probability Distribution)**.** Given random variables $\{X_1, \ldots, X_n\}$, the full joint probability distribution, denoted $\mathbf{P}(\{X_1, \ldots, X_n\})$, lists the probabilities of all atomic events.

Definition 12 (Proposition)**.** Given random variables $\{X_1, \ldots, X_n\}$, a propositional formula, short proposition, is a propositional formula over the atoms $X_i = x_i$ where x_i is a value in the domain of X_i. A function P that maps propositions into $[0, 1]$ is a probability measure if

- $P(\top) = 1$
- for all propositions A, $P(A) = \sum_{e \models A} P(e)$ where e is an atomic event.

Theorem 1 (Kolmogorow)**.** A function P that maps propositions into $[0, 1]$ is a probability measure iff

- $P(\top) = 1$ and
- for all propositions A, B:
  $P(a \vee b) = P(a) + P(b) - P(a \wedge b)$.

Definition 13 (Conditional Probability)**.** Given propositions A and B where $P(b) \neq 0$, the conditional probability, or posterior probability, of a given b, written $P(a \mid b)$, is defined as:

$$P(a \mid b) = \frac{P(a \wedge b)}{P(b)}$$

Definition 14 (Conditional Probability Distribution)**.** Given random variables X and Y, the conditional probability distribution of X given Y, written $\mathbf{P}(X \mid Y)$, is the table of all conditional probabilities of values of X given values of Y.

Definition 15 (Independence)**.** Events a and b are (stochastically) indepentent if $P(a \wedge b) = P(a) \cdot P(b)$. Random variables X and Y are indepentent if $\mathbf{P}(X, Y) = \mathbf{P}(X) \cdot \mathbf{P}(Y)$.

Proposition 1 (Product Rule)**.** Given propositions A and B, $P(a \wedge b) = P(a \mid b) \cdot P(b) = P(b \mid a) \cdot P(a)$.

Definition 16 (System of Equation)**.** $\mathbf{P}(X, Y) = \mathbf{P}(X \mid Y) \cdot \mathbf{P}(Y)$ is a system of equations. Similar for unconditional distributions, $\mathbf{P}(X, Y) = \mathbf{P}(X) \cdot \mathbf{P}(Y)$.

Proposition 2 (Chain Rule)**.** Given random variables $X_1, \ldots, X_n$, we have:

$$\mathbf{P}(X_1, \ldots, X_n) = \mathbf{P}(X_n \mid X_{n-1}, \ldots, X_1) \cdot \mathbf{P}(X_{n-1}, \ldots, X_1)$$

Proposition 3 (Marginalization)**.** Given sets $\mathbf{X}$ and $\mathbf{Y}$ of random variables: $\mathbf{P}(\mathbf{X}) = \sum_{y \in \mathbf{Y}} \mathbf{P}(\mathbf{X}, y)$ where $\sum_{y \in \mathbf{Y}}$ sums over all possible value combinations of $\mathbf{Y}$.

Definition 17 (Normalization Constant)**.** Given a vector $\langle w_1, \ldots, w_k \rangle$ of number in $[0, 1]$ where $\sum_{i=1}^{k} w_i \leq 1$, the normalization constant α is $\alpha \langle w_1, \ldots, w_k \rangle := 1/\sum_{i=1}^{k} w_i$.

Proposition 4 (Normalization)**.** Given a random variable X and an event $\mathbf{e}$, we have $\mathbf{P}(X \mid \mathbf{e}) = \alpha \mathbf{P}(X, \mathbf{e})$.

Proposition 5 (Bayes' Rule)**.** Given propositions A and B where $P(a) \neq 0$ and $P(b) \neq 0$, we have:

$$P(a \mid b) = \frac{P(b \mid a) \cdot P(a)}{P(b)}.$$

Definition 18 (conditionally independent)**.** Given sets of random variables $\mathbf{Z}_1$, $\mathbf{Z}_2$, and $\mathbf{Z}$, we say that $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are conditionally independent given $\mathbf{Z}$ if:

$$\mathbf{P}(\mathbf{Z}_1, \mathbf{Z}_2 \mid \mathbf{Z}) = \mathbf{P}(\mathbf{Z}_1 \mid \mathbf{Z}) \cdot \mathbf{P}(\mathbf{Z}_2 \mid \mathbf{Z}).$$

We alternatively say that $\mathbf{Z}_1$ is conditionally independent of $\mathbf{Z}_2$ given $\mathbf{Z}$. If $\mathbf{Z} = \emptyset$ we say $\mathbf{Z}_1$ is (stochastically) independent to $\mathbf{Z}_2$ (Def. 15).

Proposition 6. If $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are conditionally independent given $\mathbf{Z}$ then:

$$\mathbf{P}(\mathbf{Z}_1 \mid \mathbf{Z}_2, \mathbf{Z}) = \mathbf{P}(\mathbf{Z}_1 \mid \mathbf{Z}).$$

Definition 19 (Naive Bayes Model)**.** A Bayesian network in which a single cause directly influences a number of effects, all of which are conditionally independent, given the cause is called a naive Bayes model or Bayesian classifiers.

Remark 1. In a naive Bayes model, the full joint probability distribution can be written as:

$$\mathbf{P}(\text{cause} \mid \text{effect}_1, \ldots \text{effect}_n) = \mathbf{P}(\text{cause}) \cdot \prod_i \mathbf{P}(\text{effect}_i \mid \text{cause})$$

2 Bayesian Networks

Definition 20 (Bayesian Network)**.** Given random variables $X_1, \ldots, X_n$ with finite domains $D_1, \ldots, D_n$, a Bayesian network (also belief network or probabilistic network) is an acyclic directed graph $BN = \langle \{X_1, \ldots, X_n\}, E \rangle$. We denote $\text{Parents}(X_i) := \{X_j \mid (X_j, X_i) \in E\}$. Each X_i is associated with a function $\text{CPT}(X_i) : D_i \times \prod_{X_j \in \text{Parents}(X_i)} D_j \to [0, 1]$, the conditional probability table.

Definition 21 (conditionally independent)**.** Given a Bayesian network $BN = \langle \{X_1, \ldots, X_n\}, E \rangle$, we identify BN with the following two assumptions:

- for $1 \leq i \leq n$, X_i is conditionally independent of $\text{NonDesc}(X_i)$ given $\text{Parents}(X_i)$, where $\text{NonDesc}(X_i) := \{X_j \mid (X_i, X_j) \notin E^*\} \backslash \text{Parents}(X_i)$ where E^* is the transitive-reflexive closure of E.
- for $1 \leq i \leq n$, all values x_i of X_i and all value combinations of $\text{Parents}(X_i)$, we have $P(x_i \mid \text{Parents}(X_i)) = \text{CPT}(x_i, \text{Parents}(X_i))$.

Definition 22. The size of a Bayesian network is not a fixed property of the domain. It depends on the skill of the designer.

Procedure BN construction algorithm

1: Initialize $BN = \langle \{X_1, \ldots, X_n\}, E \rangle$, where $E = \emptyset$
2: Fix any order of the variables $X_1, \ldots, X_n$
3: **for** $i := 1, \ldots, n$ **do**
4: Choose a minimal set $\text{Parents}(X_i) \subseteq \{X_1, \ldots, X_{i-1}\}$ so that $\mathbf{P}(X_i \mid X_{i-1}, \ldots, X_1) = \mathbf{P}(X_i \mid \text{Parents}(X_1))$ `/* use (cond.) indpendence */`
5: **for** each $X_j \in \text{Parents}(X_i)$ **do**
6: insert (X_j, X_i) into E
7: **end loop**
8: Associate X_i with $\text{CPT}(X_i)$ corresponding to $\mathbf{P}(X_i \mid \text{Parents}(X_i))$
9: **end loop**

Definition 23 (size)**.** Given random variables $X_1, \ldots, X_n$ with finite domains $D_1, \ldots, D_n$ the size (total number of entries in the CPTs) of $BN = \langle \{X_1, \ldots, X_n\}, E \rangle$ is defined as $\text{size}(BN) := \sum_{i=1}^{n} \#(D_i) \cdot \prod_{X_j \in \text{Parents}(X_i)} \#(D_j)$

Definition 24 (deterministic)**.** A node X in a Bayesian network is called deterministic if its value is completely determined by the values of $\text{Parents}(X)$.

Definition 25 (noisy disjunction node)**.** The CPT of a noisy disjunction node X in a Bayesian network is given by $P(x_i \mid \text{Parents}(X_i)) = \prod_{\{j \mid X_j = \top\}} q_j$, where q_i are the inhibitation factors of $X_i \in \text{Parents}(X)$.

Definition 26 (Probabilistic Inference Task)**.** Given random variables $\{X_1, \ldots, X_n\}$, a probabilistic inference task consists of a set $\mathbf{X} \subseteq \{X_1, \ldots, X_n\}$ of query variables, a set $\mathbf{E} \subseteq \{X_1, \ldots, X_n\}$ of evidence variables, and an event $\mathbf{e}$ that assigns values to $\mathbf{E}$. We wish to compute the posterior probability distribution $\mathbf{P}(\mathbf{X} \mid \mathbf{e})$. $\mathbf{Y} := \{X_1, \ldots, X_n\} \setminus (\mathbf{X} \cup \mathbf{E})$ are the hidden variables.

Definition 27 (Polytree)**.** A graph is calles singly connected, or a polytree, if if there is at most one undirected path between any two nodes in the graph.

Definition 28 (Decision Theory)**.** Decision theory investigates how an agent a deals with choosing among actions based on the desirability of their outcomes.

Definition 29 (Lottery)**.** Lottery $L = [p, A; (1 - p), B]$. An agent chooses among prizes (A, B, etc.) and lotteries, i.e., situations with uncertain prizes.

Definition 30 (Expected Utility)**.** Treat the result of an action a as a random variable $R(a)$ whose variables are the possible outcome states. The expected utility $EU(a)$ of an action a (given evidence $\mathbf{e}$) is then:

$$EU(a \mid \mathbf{e}) = \sum_{s'} P(R(a) = s' \mid a, \mathbf{e}) \cdot U(s')$$

3 Making Decisions Rationally

Definition 31 (Preferences)**.** An agent decides weather

- $(A \prec B)$ when A preferred to B
- $(A \sim B)$ when indifference between A and B
- $(A \preceq B)$ when B not preferred to A.

Definition 32 (Constraints)**.**

Orderability $(A \prec B) \vee (B \prec A) \vee (A \sim B)$

Transitivity $(A \prec B) \wedge (B \prec C) \Rightarrow (A \prec C)$

Continuity $(A \prec B \prec C) \Rightarrow (\exists p.\ [p, A; (1-p), C] \sim B)$

Substitutability $(A \sim B) \Rightarrow$
$([p, A; (1-p), C] \sim [p, B; (1-p), C])$

Monotonicity $(A \prec B) \Rightarrow$
$(p \geq q) \Leftrightarrow ([p, A; (1-p), B] \preceq [q, A; (1-q), B])$

Theorem 2 (Ramseys, 1931)**.** Given preferences satisfying the constraints there exists a real-valued function U such that

$$(U(A) \geq U(B)) \Leftrightarrow A \preceq B \text{ and}$$
$$U([p_1, S_1, \ldots, p_n, S_n]) = \textstyle\sum_i p_i U(S_i).$$

Remark 2. Agent behavior is invariant w.r.t. positive linear transformation, i.e.

$$U'(x) = k_1 U(x) + k_2 \text{ where } k_1 > 0$$

behaves exactly like U.

Definition 33 (value function)**.** We call a total ordering on states a value function or ordinal utility function.

Definition 34 (MEU principle)**.** Choose the action that maximizes expected utility.

Definition 35 (Standard approach to assessment of human utilities)**.** Compare a given state A to a standard lottery L_p that has

- "best possible prize" $u_\top$ with probability p
- "worst possible catastrophe" $u_\bot$ with probability $1-p$

Adjust lottery probability p until $A \sim L_p$. Then $U(A) = p$.

Definition 36 (Normalized utility)**.** $u_\top = 1, u_\bot = 0$

Definition 37 (Micromorts)**.** One-millionth chance of death (useful for Russian roulette, paying to reduce product risks, etc.).

Definition 38 (QALYs)**.** Quality-adjusted life years (useful for medical decisions involving substantial risk).

Definition 39 (Dominance)**.** Choice B strictly dominates choice $A \Leftrightarrow X_i(B) \geq X_i(A)$ for all i (and hence $U(B) \geq U(A)$).

Definition 40 (Stochastical Dominance)**.** Distribution p_1 stochastically dominates distribution p_2 if the cummulative distribution of p_2 dominates that for p_1 for all t:

$$\int_{-\infty}^{t} p_1(x)dx \leq \int_{-\infty}^{t} p_2(x)dx$$

Definition 41 (Positive Influence)**.** $X \xrightarrow{+} Y$ (X positively influences Y) means that $\mathbf{P}(Y \mid x_1, \mathbf{z})$ stochastically dominates $\mathbf{P}(Y \mid x_2, \mathbf{z})$ for every value z of Y's other parents $\mathbf{Z}$ and all x_1 and x_2 with $x_1 \geq x2$.

Definition 42 (Independence)**.** X_1 and X_2 (preferentially) independent of X_3 iff preference between $\langle x_1, x_2, x_3\rangle$ and $\langle x_1', x_2', x_3'\rangle$ does not depend on x_3.

Theorem 3 (Leontief, 1947)**.** If every pair of attributes is preferentially independent of its complement, then every subset of attributes is preferentially independent of its complement: mutual preferential independence.

Theorem 4 (Debreu, 1960)**.** Mutual preferential independence implies that there is an additive value function: $V(S) = \sum_i V_i(X_i(S))$, where V_i is a value function referencing just one variable X_i.

Definition 43 (utility-independence)**.** $\mathbf{X}$ is utility-independent of $\mathbf{Y}$ iff preferences over lotteries in $\mathbf{X}$ do not depend on particular values in $\mathbf{Y}$.

Definition 44 (mutually utility-independence)**.** A set $\mathbf{X}$ is mutually utility-independent, iff each subset is utility-independent of its complement.

Theorem 5. For mutually utility-indepenent sets there is a multiplicative utility function:

$$\begin{aligned} U = {} & k_1 U_1 + k_2 U_2 + k_3 U_3 \\ & + k_1 k_2 U_1 U_2 + k_2 k_3 U_2 U_3 \\ & + k_3 k_1 U_3 U_1 + k_1 k_2 k_3 U_1 U_2 U_3 \end{aligned}$$

Definition 45 (value nodes)**.** Add action nodes $\square$ and utility nodes $\diamond$ (also called value nodes) to belief networks to enable rational decision making.

Definition 46 (Expected utility)**.** With current evidence E, current best action α and possible action outcomes S_i the expected utility is

$$EU(\alpha \mid E) = \max_a (\sum_i U(S_i) \cdot P(S_i \mid E, a))$$

Definition 47 (VPI)**.** Value of perfect information.

$$\mathrm{VPI}_E(E_j) =$$
$$\sum_k P(E_j = e_{jk} \mid E) \cdot EU(\alpha_{e_{jk}} \mid E, E_{jk} = e_{jk}) - EU(\alpha \mid E)$$

Definition 48 (Properties of VPI)**.**

- nonnegative: $\text{VPI}_E(E_j) \geq 0$ for all j and E
- nonadditive:

$$\text{VPI}_E(E_j, E_k) \neq \text{VPI}_E(E_j) + VPI_E(E_k)$$

- order-independent:

$$\begin{aligned}\text{VPI}_E(E_j, E_k) &= \text{VPI}_E(E_j) + \text{VPI}_{E,E_j}(E_k) \\ &= \text{VPI}_E(E_k) + \text{VPI}_{E,E_k}(E_j)\end{aligned}$$

Definition 49. A simple Information-Gathering Agent:

Function INFORMATION-GATHERING-AGENT(percept)
Returns:
an action
Locals:
D, a decision network
integrate percept into D
$j := \underset{k}{\text{argmax}}(\text{VPI}_E(E_k)/Cost(E_k))$
if $\text{VPI}_E(E_j) > Cost(E_j)$ **then**
  return Request(E_j)
else
  return the best action from D

4 Temporal Probability Models

Definition 50 (temporal probability model)**.** A temporal probability model is a probability model, where possible worlds are indexed by a time structure $\langle S, \preceq \rangle$.

Definition 51 (Markov property)**.** $\mathbf{X}_t$ only depends on a bounded subset of $\mathbf{X}_{0:t-1}$.

Definition 52 (Markov process)**.** A (discrete-time) Markov process (also called Markov chain) is a sequence of random variables with the Markov property.

- First-order Markov process:

$$\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$$

- Second-order Markov process:

$$\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-2}, \mathbf{X}_{t-1})$$

Definition 53. We divide the random variables in a Markov process M into a set of (hidden) state variables $\mathbf{X}_t$ and a set of (observable) evidence variables $\mathbf{E}_t$. We call $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$ the transition model and $\mathbf{P}(\mathbf{E}_t \mid \mathbf{E}_{t-1})$ the sensor model of M.

Definition 54 (Stationarity)**.** A Markov process is called stationary if the transition model is independent of time, i.e. $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$ is the same for all t.

Definition 55 (Sensor Property)**.** We say that a sensor model has the sensor Markov property, iff $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$.

Remark 3 (Transition & Sensor Models)**.**

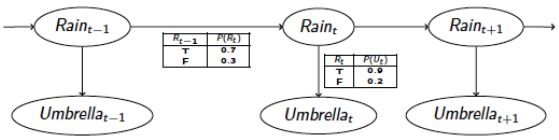

Figure 1: Markov process example with transition and sensor probabilities (models)

Remark 4. If we additionally know the initial prior probabilities $\mathbf{P}(\mathbf{X}_0)$ (time $t = 0$), then we can compute the full joint probability distribution as

$$\mathbf{P}(\mathbf{X}_{0:t}, \mathbf{E}_{0:t}) = \mathbf{P}(\mathbf{X}_0) \cdot \prod_{i=1}^{t} \mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1}) \cdot \mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)$$

Definition 56 (Filtering or monitoring)**.** $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$, computing the belief state input to the decision process of a rational agent.

$$\begin{aligned}&\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) \\ &= \alpha \cdot \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \cdot \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}) \\ &= \alpha \cdot \underbrace{\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1})}_{\substack{\text{sensor} \\ \text{model}}} \cdot \sum_{\mathbf{x}_t} \underbrace{\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t)}_{\substack{\text{transition} \\ \text{model}}} \cdot \underbrace{\mathbf{P}(\mathbf{x}_t \mid \mathbf{e}_{1:t})}_{\substack{\text{recursive} \\ \text{call}}}\end{aligned}$$

Definition 57 (Prediction or state estimation)**.** $\mathbf{P}(\mathbf{X}_{t+k} \mid \mathbf{e}_{1:t})$ for $k > 0$, evaluation of possible action sequences (filtering without new evidence).

$$\begin{aligned}&\mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{e}_{1:t}) \\ &= \sum_{\mathbf{x}_{t+k}} \underbrace{\mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{x}_{t+k})}_{\substack{\text{transition} \\ \text{model}}} \cdot \underbrace{\mathbf{P}(\mathbf{x}_{t+k} \mid \mathbf{e}_{1:t})}_{\substack{\text{recursive} \\ \text{call}}}\end{aligned}$$

Definition 58 (Smoothing or hindsight)**.** $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ for $0 \leq k < t$, better estimate of past states.

$$\begin{aligned}&\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) \\ &= \alpha \cdot \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}) \cdot \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k)\end{aligned}$$

with backward message $\mathbf{b}_{k+1:t}$:

$$\begin{aligned}&\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) \\ &= \sum_{\mathbf{x}_{k+1}} \underbrace{P(\mathbf{e}_{k+1} \mid \mathbf{x}_{k+1})}_{\substack{\text{sensor} \\ \text{model}}} \cdot \underbrace{P(\mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})}_{\substack{\text{recursive} \\ \text{call}}} \cdot \underbrace{\mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k)}_{\substack{\text{transition} \\ \text{model}}}\end{aligned}$$

Definition 59 (Forward-backward algorithm)**.** Cache forward messages along the way:

Function FORWARD-BACKWARD(ev, $prior$)

Inputs:
ev, a vector of evidence values for steps $1, \ldots, t$
$prior$, the prior distribution on the initial state

Returns:
a vector of pabability distributions

Locals:
fv, a vector of forward messages for steps $1, \ldots, t$
b, a representation of the backward message
sv, a vector of smoothed estimates for steps $1, \ldots, t$

1: $fv[0] := prior$
2: **for** $i = 1$ **to** t **do**
3: $\quad fv[i] :=$ FORWARD$(fv[i-1], ev[i])$
4: **for** $i = t$ **downto** 1 **do**
5: $\quad sv[i] :=$ NORMALIZE$(fv[i], b)$
6: $\quad b :=$ BACKWARD$(b, ev[i])$
7: **return** sv

Remark 5. Time linear in t (polytree inference), space $\mathcal{O}(t \cdot \#(\mathbf{f}))$.

Definition 60 (Most Likely Explanation)**.** *tbd.*

Definition 61 (Transition Matrix)**.**

$$\mathbf{T}_{ij} = P(X_t = j \mid X_{t-1} = i)$$

Definition 62 (Sensor Matrix)**.** $\mathbf{O}_t$ for each time step, diagonal elements $P(e_t \mid X_t = i)$.

Definition 63 (Hidden Markov Models)**.** Forward and backward messages as column vectors:

HMM filtering : $\mathbf{f}_{1:t+1} = \alpha \cdot (\mathbf{O}_{t+1}\ \mathbf{T}^t\ \mathbf{f}_{1:t})$

HMM smoothing : $\mathbf{b}_{k+1:t} = \mathbf{T}\ \mathbf{O}_{k+1}\ \mathbf{b}_{k+2:t}$

Definition 64 (Dynamic Bayesian Network)**.** A Bayesian network D is called dynamic (a DBN), if its random variables are indexed by a time structure. We assume that its structure is

- time sliced, i.e. that the time slices D_t – the subgraphs of t-indexed random variables and the edges between them – are isomorophic
- a first-order Markov process, i. e. that variables X_t can only have parents in D_t and D_{t-1}

Definition 65 (Naive method)**.** Unroll the network and run any exact algorithm. Problem: inference cost for each update grows with t.

Definition 66 (Rollup filtering)**.** Add slice $t+1$, sum out slice t using variable elimination.

5 Making Complex decisions

Definition 67 (Markov Process)**.** A sequential decision problem in a fully observable, stochastic environment with a Markovian transition model and an additive reward function is called a Markov decision process. It consist of

- a set of S of states (with initial state $s_0 \in S$)
- sets Actions(s) of actions for each state s
- a transition model $P(s' \mid s, a)$
- a reward function $R : S \to \mathbb{R}$

Definition 68 (stationary)**.** We call preferences on reward sequences stationary, iff

$$[r, r_0, r_1, r_2, \ldots] \prec [r, r'_0, r'_1, r'_2, \ldots] \prec$$
$$\Leftrightarrow [r_0, r_1, r_2, \ldots] \prec [r'_0, r'_1, r'_2, \ldots]$$

Theorem 6. For stationary preferences, there are only two ways to combine rewards over time.

- An additive rewards:
$$U([s_0, s_1, \ldots]) = R(s_0) + R(s_1) + \ldots$$
- A discounted rewards:
$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_1) + \ldots$$
where γ is the discount factor.

Definition 69 (Expected Utility)**.** Given a policy π, let S_t be the state the agent reaches at time t starting at state s_0. Then the expected utility obtained by executing π starting in s is given by

$$U^\pi(s) = E(\sum_{t=0}^{\infty} \gamma^t\ R(s_t))$$

we define the $\pi^*_s := \operatorname{argmax}_\pi(U^\pi(s))$.

Definition 70 (optimal policy)**.** We call $\pi^* := \pi^*_s$ for some s the optimal policy. This does not hold for finite-horizon policies.

Definition 71. The utility $U(s)$ of a state s is $U^{\pi^*}(s)$.

Theorem 7 (Bellman equation, 1957)**.**

$$U(s) = R(s) + \gamma \cdot \max_a \sum_{s'} U(s') \cdot T(s, a, s')$$

Definition 72 (Value Iteration Algorithm)**.** The value iteration algorithm for utility functions is given by

Function VALUE-ITERATION(mdp, ε)

Inputs:
mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$, rewards $R(s)$, and discount γ.
ε, the max. error allowed in the utility of any state

Returns:
a utility function

Locals:
U, U', vectors of utilities for states in S, initially zero
δ, max. change in utility of any state in an iteration

1: **repeat**
2: $\quad U := U'$, $\delta := 0$
3: $\quad$ **for each** State s in S **do**
4: $\quad\quad U'[s] := R(s) + \gamma \cdot \max_a \sum_{s'} U[s'] \cdot P(s' \mid a, s)$
5:
6: $\quad\quad$ **if** $|U'[s] - U[s]| > \delta$ **then** $\delta := |U'[s] - U[s]|$
7: $\quad$ **end for**
8: **until** $\delta < \varepsilon \frac{(1-\gamma)}{\gamma}$
9: **return** U

Theorem 8. For any two approximations U^t and V^t

$$||U^{t+1} - V^{t+1}|| \leq \gamma ||U^t - V^t||$$

i. e. any distinct approximations must get closer to each other so, in particular, any approximation must get closer to the true U and value iteration converges to a unique, stable, optimal solution.

Theorem 9. If $||U^{t+1} - U^t|| < \varepsilon$ then $||U^{t+1} - U|| < 2\varepsilon \frac{\gamma}{(1-\gamma)}$. I. e. once the change in U^t becomes small, we are almost done.

Definition 73. The policy iteration algorithm is given by the following pseudocode:

Function POLICY-ITERATION(mdp)

Inputs:
mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$, rewards $R(s)$, and discount γ.

Returns:
a policy

Locals:
U, a vector of utilities for states in S, initially zero
π, a policy indexed by state, initially random

1: **repeat**
2: $U :=$ POLICY-EVALUATION(π, U, mdp)
3: unchanged := TRUE
4: **for each** State s in X **do**
5: **if** $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) \cdot U(s') > \sum_{s'} P(s' \mid s, \pi[s']) \cdot U(s')$ **then**
6: $\pi[s] := \operatorname{argmax}_{b \in A(s)} (\sum_{s'} P(s' \mid s, a) \cdot U(s'))$
7: unchanged := FALSE
8: **end if**
9: **end for**
10: **until** unchanged
11: **return** π

Definition 74 (partially observable MDP)**.** A partially observable MDP (a POMDP for short) is a MDP together with an observation model O that is stationary and has the sensor Markov property: $O(s, e) = P(e \mid s)$.

Theorem 10 (Astrom, 1965)**.** The optimal policy in a POMDP is a function $\pi(b)$ where b is the belief state (probability distribution over states).

Definition 75 (Dynamic Decision Networks)**.**

- Transition and Sensor Networks are represented as a DBN (a dynamic Bayesian network)
- action nodes and utility nodes are added to create a dynamic decision network (DDN)
- a filtering algo is used to incorporate each new percept and action and to update the belief state representation
- decisions are made by projecting forward possible action sequences and choosing the best one

6 Learning from Observations

Definition 76 (Learning agent)**.** This is an agent that augments the performance element (which chooses actions from percept sequences) with a

learning element that makes improvements to the agent's performance element

critic which gives feedback to the learning element based on an external performance standard

problem generator which suggests actions that can lead to new, informative experiences

Definition 77 (Target Function)**.** f is the target function we want to learn. An example is a pair $(x, y) \in f$.

Definition 78 (Inductive Learning)**.** The inductive learning problem $\langle \mathcal{H}, f \rangle$ consists in finding a hypothesis $h \in \mathcal{H}$ such that $h|_{\text{dom}(f)} \approx f$ given a training set f of examples and a hypothesis space $\mathcal{H}$.

Definition 79. We call h consistent with f, if it agrees with f on all examples in T.

Definition 80 (Ockham's razor)**.** Maximize a combination of consistency and simplicity.

Definition 81. We say an inductive learning problem $\langle \mathcal{H}, f \rangle$ is realizable, if there is a $h \in \mathcal{H}$ consistent with f.

Definition 82 (attribute-based)**.** In attribute-based representations, examples are described by attributes and their values (Boolean, discrete, continuous, etc.).

Definition 83 (Classification)**.** Classification of examples is positive (T, $\top$) or negative (F, $\bot$).

Definition 84 (Decision Tree Learning)**.** The following algorithm performs decision tree learning:

Algorithm DTL(*examples*, *attributes*, *default*)

1: **if** *examples* is empty **then return** *default*
2: **else if** all *examples* have the same classification **then**
3: **return** the classification
4: **else if** *attributes* is empty **then**
5: **return** MODE(*examples*)
6: **else**
7: *best* := CHOOSE-ATTRIBUTE(*attributes*, *examples*)
8: *tree* := a new decision tree with root test *best*
9: $m :=$ MODE(*examples*)
10: **for each** value v_i of *best* **do**
11: *examples* := {elements of *examples* with $best = v_i$}
12: *subtree* := DTL($examples_i$, $attributes \setminus best, m$)
13: add a branch to *tree* with label v_i and subtree *subtree*
14: **end for**
15: **return** *tree*

Remark 6. MODE(*examples*) := most frequent value in *examples*.

Definition 85 (Information)**.** If the prior is $\langle P_1, \ldots, P_n \rangle$, then the information in an answer is

$$I(\langle P_1, \ldots, P_n \rangle) = \sum_{i=1}^{n} -P_i \cdot \log_2(P_i)$$

(also called entropy of the prior).

Definition 86 (Information Gain)**.** The information gain from an attribute test A is

$$\textsc{Gain}(A) := I(\langle \frac{p}{p+n}, \frac{n}{p+n} \rangle) - \sum_i \frac{p_i + n_i}{p+n} \cdot I(\langle \frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i} \rangle)$$

Remark 7. The learning curve depends on

- realizable (can express target function) vs. non-realizable (can be due to missing attributes or restricted hypothesis class)
- redundant expressiveness (e.g., loads of irrelevant attributes)

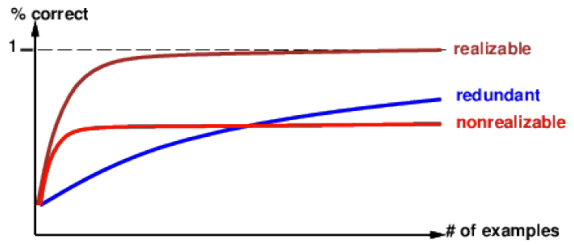

Figure 2: different learning curves

Definition 87 (Overfitting)**.** We speak of overfitting, if a hypothesis h describes random error rather than the underlying relationship. Underfitting occurs when h cannot capture the underlying trend of the data. Overfitting increases with the size of hypothesis space and the number of attributes, but decreases with number of examples.

Definition 88 (decision tree pruning)**.** For decision tree pruning repeat the following on a learned decision tree:

- Find a terminal test node n (only result leaves as descendent)
- if test is irrelevant, i.e. has low information gain, prune it by replacing n by with a leaf node.

Definition 89 (Significance)**.** A result has statistical significance, if the probability they could arise from the null hypothesis (the assumption that there is no underlying pattern) is very low (usually 5%).

Remark 8 (Sum of squared errors)**.** A convenient measure of the total deviation is

$$\Delta = \sum_{k=1}^{d} \frac{(p_k - \hat{p}_k)^2}{\hat{p}_k} + \frac{(n_k - \hat{n}_k)^2}{\hat{n}_k}$$

Definition 90 (χ^2 pruning)**.** Decision tree pruning with Pearson's χ^2 with $d-1$ degrees of freedom for Δ is called χ^2 pruning.

Definition 91 (IID)**.** A sequence of E_j of random variables is independent and identically distributed (short IID), iff they are

- independent,
  i. e. $\mathbf{P}(E_j \mid E_{j-1}, E_{j-2}, \ldots) = \mathbf{P}(E_j)$ and
- independently distributed,
  i. e. $\mathbf{P}(E_i) = \mathbf{P}(E_j)$ for all i and j.

Definition 92 (Error Rate)**.** Given an inductive learning problem $\langle \mathcal{H}, f \rangle$, we define the error rate of a hypothesis $h \in \mathcal{H}$ as

$$\frac{\#(\{x \in \mathbf{dom}(f) \mid h(x) \neq f(x)\})}{\#(\mathbf{dom}(f))}$$

Definition 93 (holdout cross-validation)**.** Splitting the data available for learning into a

- training set from which the learning algorithm produces a hypothesis h
- and a test set, which is used for evaluating h

is called holdout cross-validation (no peeking at test set allowed).

Definition 94 (k-fold cross-validation)**.** In k-fold cross-validation, we perform k rounds of learning, each with $\frac{1}{k}$ of the data as test set and average ove the k test scores.

Definition 95 (LOOCV)**.** If $k = \#(\mathbf{dom}(f))$, then k-fold cross-validation is called leave-one-out cross-validation (LOOCV).

Definition 96 (Model Selection)**.** The model selection problem is to determine (given data) a good hypothesis space.

Remark 9. We can solve the problem of "learning from observations f" in a two-part process:

- model selection determines a hypotheses space $\mathcal{H}$,
- optimization solves the induced inductive learning problem $\{\mathcal{H}, f\}$.

Definition 97 (Model Selection Algorithm)**.**

Function
CROSS-VALIDATION-WRAPPER(*Learner*, k, *examples*)
Returns:
a hypothesis
Locals:
errT, an array, storing training-set error rates
errV, an array, storing validation-set error rates
for $size = 1$ **to** ∞ **do**
  errT[*size*], *errV*[*size*] :=
    CROSS-VALIDATION(*Learner*, *size*, k, *examples*)
  if *errT* hast converged **then**
    best_size :=
      the value of *size* with minimum *errV*[*size*]
    return LEARNER(*best_size*, *examples*)
  end if
end for

Function
CROSS-VALIDATION(*Learner*, *size*, k, *examples*)

Returns:
average training set error rate,
average validation set error rate

1: $fold\_errT := 0$, $fold\_errV := 0$
2: **for** $fold = 1$ **to** k **do**
3: $training\_set, validation\_set :=$ PARTITION(*examples*, *fold*, k)
4: $h :=$ LEARNER(*size*, *training_set*)
5: $fold\_errT :=$ $fold\_errT\_set +$ ERROR-RATE(h, *training_set*)
6: $fold\_errV :=$ $fold\_errV\_set +$ ERROR-RATE(h, *validation_set*)
7: **end for**
8: **return** $\frac{fold\_errT}{k}, \frac{fold\_errV}{k}$

Remark 10. PARTITION(*examples*, *fold*, k) returns two sets: a validation set of size $\frac{|examples|}{k}$ and the rest (the split is different for each *fold* value).

Definition 98 (Loss function)**.** The loss function L is defined by setting $L(x, y, \hat{y})$ to be the amount of utility lost by prediction $h(x) = \hat{y}$ instead of $f(x) = y$. If L is independent of x, we often use $L(y, \hat{y})$.

Definition 99 (Popular general loss functions)**.**

- absolute value loss: $L_1(y, \hat{y}) := |y - \hat{y}|$
- squared value loss: $L_2(y, \hat{y}) := (y - \hat{y})^2$
- 0/1 loss: $L_{0/1}(y, \hat{y}) := 0$, if $y = \hat{y}$, else 1

Definition 100 (Generalization)**.** Let ε be the set of all possible example and $\mathbf{P}(X, Y)$ the prior probability distribution over its components, then the expected generalization loss for a hypothesis h with respect to a loss function L is

$$\text{GENLOSS}_L(h) := \sum_{(x,y)\in\mathcal{E}} L(y, h(x)) \cdot P(x, y)$$

and the best hypothesis $h^* := \operatorname{argmin}_{h\in\mathcal{H}} \text{GENLOSS}_L(h)$.

Definition 101 (Emperical Loss)**.** $\mathbf{P}(X, Y)$ is not know so the learner can only estimation generalization loss: let L be a loss function and E a set of examples with $\#(E) = N$, then

$$\text{EMPLOSS}_L(h) := \frac{1}{N} \sum_{(x,y)\in E} L(y, h(x))$$

and the estimated hypothesis $\hat{h}^* := \operatorname{argmin}_{h\in\mathcal{H}} \text{EMPLOSS}_L(h)$.

Remark 11. There are four reasons why $\hat{h}^*$ may differ from f:

- realizablility: then we have to settle for an approximation h^* of f
- variance: different subsets of f give differents $h^* \to$ more examples
- noise: if f is non-deterministic, then we cannot expect perfect results
- computational complexity: if $\mathcal{H}$ is too large to systematically explore, we make due with subset and get an approximation.

Definition 102 (PAC)**.** Any learning algorithm that returns hypotheses that are probably approximately correct is called a PAC learning algorithm.

Definition 103. The error rate error(h) of a hypothesis h is the probability that h misclassifies a new example.

$$\text{error}(h) := \text{GENLOSS}_{L_{0/1}}(h) = \sum_{(x,y)\in\mathcal{E}} L_{0/1}(y, h(x)) \cdot P(x, y)$$

Definition 104. A hypothesis h is called approximatively correct, iff $\text{error}(h) \leq \varepsilon$ for some small $\varepsilon > 0$. We write $\mathcal{H}_b := \{h \in \mathcal{H} \mid \text{error}(h) > \varepsilon\}$.

Definition 105. The number of required examples as a function of ε and δ is called the sample complexity of $\mathcal{H}$.

Definition 106 (Classification, Regression)**.** We call an inductive learning problem $\{\mathcal{H}, f\}$ a classification problem, iff $\mathbf{codom}(f)$ is discrete, and a regression problem if $\mathbf{codom}(f)$ is continuous, i. e. non-discrete (usually real-valued).

Definition 107. A univariate or unary function is a function with one argument.

Remark 12. A univariate, linear function $f : \mathbb{R} \to \mathbb{R}$ is of the form $f(x) = w_1 x + w_0$ for some $w_i \in \mathbb{R}$.

Definition 108. Given a vector $\mathbf{w} := (w_0, w_1)$, we define $h_{\mathbf{w}}(x) := w_1 x + w_0$.

Definition 109. Given a set of examples $E \subseteq \mathbb{R} \times \mathbb{R}$, the task of finding $h_{\mathbf{w}}$ that best fits E is called linear regression.

Definition 110 (Weight Space)**.** The weight space is the space of all possible combinations of weights. The weight space of univariate linear regression is $\mathbb{R}^2$ (convex).

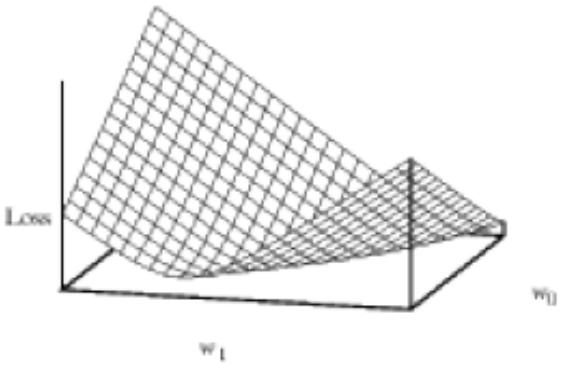

Figure 3: graph of the loss function over $\mathbb{R}^2$

Remark 13. The squared error loss function is convex for any linear regression problem. There are no local minima.

Definition 111 (Gradient descent algorithm)**.** The gradient descent algorithm for finding a minimum of a continuous function f is hill-climbing in the direction of the steepest descent, which can be computed by the partial derivatives of f.

Function GRADIENT-DESCENT$(f, \mathbf{w}, \alpha)$

Inputs:
a differentiable function f and
initial weights $\mathbf{w} = (w_0, w_1)$
Returns:
a local minimum of f
repeat
  for each w_i **do**
    $w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i}(f(\mathbf{w}))$
  end for
until **w** converges

The parameter α is called the learning rate. It can be a fixed constant or it can decay as learning proceeds.

Definition 112 (Decision Boundary)**.** A decision boundary is a line (or a surface, in higher dimensions) that separates two classes of points. A linear decision boundary is called a linear separator and data that admits one are called linearly separable.

Definition 113. For n training examples (x_j, y_j) we have:

$$w_0 \leftarrow w_0 - \alpha \sum_j -2(y_j - h_{\mathrm{w}}(x_j))$$

$$w_1 \leftarrow w_1 - \alpha \sum_j -2(y_j - h_{\mathrm{w}}(x_j)) \cdot x_n$$

These updates constitute the batch gradient-descent learning rule for univariate linear regression.

Definition 114. A multivariate or n-ary function is a function with one or more arguments.

Definition 115. Given an example $(\mathbf{x}, y)$, the perceptron learning rule is

$$w_i \leftarrow w_i + \alpha \cdot (y - h_{\mathbf{w}}(\mathbf{x})) \cdot x_i$$

Definition 116 (Logistic regression)**.** The process of weight-fitting in

$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w} \cdot x)}}$$

is called logistic regression. There is no easy closed form solution, but gradient descent is straightforward.

Definition 117. Mental activity consists primarily of electrochemical activity in networks of brain cells called neurons.

Definition 118. The animal brain is a biological neural network

- with 10^{11} neurons of > 20 types, 10^{14} synapses and 1 ms to 10 ms cycle time.
- Signals are noisy "spike trains" of electrical potential

Definition 119. The AI sub-field of neural networks (also called connectionism, parallel distributed processing, and neural computation) studies computing systems inspired by the biological neural networks that constitute brains.

Definition 120 (Neuronal Network)**.** An artificial neural network is a directed graph of units and links. A link from unit i to unit j propagates the activation $a_i k$ from i to j, it has a weight $w_{i,j}$ associated with it.

Definition 121 (McCulloch-Pitts)**.** A McCulloch-Pitts unit first computes a weighted sum of all inputs and then applies an activation function g to it.

$$\mathrm{in}_i = \sum_j w_{j,i} a_j$$

$$a_i \leftarrow g(\mathrm{in}_i)) = g(\sum_j w_{j,i} a_j)$$

If g is a threshold function, we call the unit a perceptron unit, if g is a logistic function a sigmoid perceptron unit. A McCullorch-Pitts network is a neural network with McCullorch-Pitts units.

Theorem 11 (McCulloch and Pitts)**.** Every Boolean function can be implemented as McCulloch-Pitts units.

Definition 122 (Feed-Forward)**.** A neural network is called a feed-forward network, if it is acyclic. We will look at single-layer and multi-layer perceptrons.

Definition 123 (Recurrence)**.** A neural network is called recurrent, iff it has cycles.

- Hopfield networks have symmetric weights $(w_{i,j} = w_{j_i})$ $g(x) = \mathrm{sign}(x)$, $a_i = \pm 1$; holographic associative memory
- Boltzmann machines use stochastic activation functions, $\approx$ MCMC in Bayes nets

Definition 124 (Perception network)**.** A perceptron network is a feed-forward network of perceptron units. A single-layer perceptron network is called perceptron.

Definition 125 (Hidden units)**.** Layers are usually fully connected. Numbers of hidden units typically chosen by hand.

Definition 126 (Back-propagation)**.** The back-propagation rule for hidden nodes of a multilayer perceptron is

$$\Delta_j \leftarrow g'(\mathrm{in}_i) \cdot \sum_i w_{j,i} \Delta_i$$

Update rule for weights in hidden layer:

$$w_{k,j} \leftarrow w_{k,j} + \alpha \cdot a_k \cdot \Delta_j$$

Remark 14. Most neuroscientists deny that back-propagation occurs in the brain.

Definition 127 (Back-propagation learning algorithm)**.**

Function BACK-PROP-LEARNING(*examples*, *network*)
Inputs:
examples, a set of examples, each with input vector $\mathbf{x}$ and output vector $\mathbf{y}$
network, a multilayer network with L layers, weights $w_{i,j}$, activation function g
Returns:
a neuronal network
Locals:
Δ, a vector of errors, indexed by network node

repeat
  for each weight $w_{i,j}$ in *network* **do**
    $w_{i,j}$:= a small random number
    for each example $(\mathbf{x}, \mathbf{y})$ in *examples* **do**
      /* Propagate the inputs forward to compute the outputs */
      for each node i in the input layer **do**
        $a_i := x_i$
      for $I = 2$ to L **do**
        for each node j in layer I **do**
          $\text{in}_i := \sum_i w_{i,j} \cdot a_i$
          $a_j := g(\text{in}_i)$
        end for
        /* Propagate deltas backward from output layer to input layer */
        for each node j in output layer **do**
          $\Delta[j] := g'(\text{in}_i) \cdot (y_j - a_j)$
        for $I = L - 1$ to 1 **do**
          for each node i in layer I **do**
            $\Delta[i] := g'(\text{in}_i) \cdot \sum_j w_{i,j} \cdot \Delta[j]$
          /* Update every weight in network using deltas */
          for each $w_{i,j}$ in network **do**
            $w_{i,j} := w_{i,j} + \alpha \cdot a_i \cdot \Delta[j]$
        end for
      end for
    end for
until some stopping criterion is satisfied
return *network*

Definition 128 (Support Vector Machine)**.** Given a linearly separable data set E the maximal margin separator is the linear separator s that maximizes the margin, i. e. the distance of the E from s. Find the optimal solution by solving the SVM equation

$$\underset{\alpha}{\text{argmax}}(\sum_j \alpha_j - \frac{1}{2}\sum_{j,k} \alpha_j\ \alpha_k\ y_j\ y_k\ (\mathbf{x_j} \cdot \mathbf{x_j}))$$

under the constraints $\alpha_j \geq 0$ and $\sum_j \alpha_j\ y_j = 0$.

Definition 129 (Kernel Function)**.** We call a function $(\mathbf{x_j} \cdot \mathbf{x_j})^2$ a Kernel function.

Definition 130 (Polynomial Kernel)**.** The function $K(\mathbf{x}_j, \mathbf{x}_k) = (1 + (\mathbf{x}_j \cdot \mathbf{x_j}))^d$ a Kernel function corresponding to a feature space whose dimension is exponential in d. It is called the polynomial kernel.

Theorem 12 (Mercer's Theorem)**.** Every kernel function where $K(\mathbf{x}_j, \mathbf{x}_k)$ is positive definite corresponds to some feature space.