Generalizing Agent: Properties

Assume a perfect conjunction h^* exists: $C(h^*) = C$, i.e. the target concept can be expressed through a conjunction. Observe:

1. All literals of h^* are consistent with any positive example (otherwise h^* would not be true for a positive example).
2. If a hypothesis misclassifies a negative example, it has all literals consistent with the example.
3. From 1 and 2: no literal that is in h^* is removed from the agent's hypothesis. Since h_1 contains *all* literals, we have $h_k \supseteq h^*$, $\forall k \in N$.
4. From 2 and 3: if h_k misclassifies a neg. example, then h^* ($\subseteq h_k$) is consistent with the example $\rightarrow$ contradiction $\rightarrow$ *h_k never ($\forall k$) misclassifies negative examples* (so yes, they are useless to this agent).

$h_k \supseteq h^*$ means that all literals of h^* are in h_k. We will use set relations for conjunctions and disjunctions this way without further warnings.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Generalizing Agent: Mistake Bound

1. Since h_k makes mistakes only on positive examples, whenever it misclassifies some o_k, it contains some literals inconsistent with o_k.
2. From 1 and the fact that all inconsistent literals are deleted after each mistake: at least one literal is deleted on each mistake.
3. From 2 and the fact that the initial hypothesis h_1 has $2n$ literals: *no more than $2n$ mistakes* are made before $h_k = h^*$.

So the cumulative reward for an arbitrary horizon $m \in N$ is

$$\sum_{k=1}^{m} r_k \geq -2n$$

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Mistake-Bound Learning Model

Let $\mathcal{H}$ be a *hypothesis class*, i.e. any set of hypotheses. $\mathcal{H}$ induces the *concept class* $\mathcal{C}(\mathcal{H}) = \{ C(h) \mid h \in \mathcal{H} \}$. Let n be the *size* (complexity) of observations (usually their arity).

On-Line (Mistake-Bound) Learning

Agent *learns class $\mathcal{H}$ on-line (in the mistake-bound model)* if with any target concept $C \in \mathcal{C}(\mathcal{H})$ it will make no more than a polynomial (in n) number of mistakes. Furthermore, it learns $\mathcal{H}$ *efficiently* if it spends at most poly-time btw. each percept and the next action. $\mathcal{H}$ is *(efficiently) learnable on-line)* if there is an agent that learns it (efficiently) on-line.

The definition only assumes $C \in \mathcal{C}(\mathcal{H})$ but makes no assumption about the choice and order of observations coming from the environment. The mistake bound must hold for any sequence of observations.

Verify that the generalization agent learns conjunctions *efficiently*.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Generality Relation

We call *concept C more general than concept C'* if $C \supseteq C'$. A *hypothesis h is more general than hypothesis h'* if

$$C(h) \supseteq C(h') \tag{1}$$

Assume h, h' are logic formulas and the policy is $\pi(h, o) = 1 \leftrightarrow o \models h$ as in the generalizing agent. Then a *sufficient* condition for (1) is that $o \models h'$ implies $o \models h$ for any model (valuation) o. In logic, this is written as

$$h' \models h \tag{2}$$

If furthermore h, h' are *conjunctions*, a *sufficient* condition for (2) is that

$$h \subseteq h'$$

because for any o, h: $o \models h$ iff *each literal* of h is consistent with o.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Generality vs. Subsumption

When $h \subseteq h'$ for conjunctions h, h', we say that h *subsumes* h'. Although $h \subseteq h'$ implies $h' \models h$, the reverse is not true if h' is *self-resolving*, i.e. it contains a literal as well as its negation. For example

$$\mathrm{p}_1 \wedge \neg \mathrm{p}_1 \models \mathrm{p}_2$$

but

$$\mathrm{p}_2 \not\subseteq \mathrm{p}_1 \wedge \neg \mathrm{p}_1$$

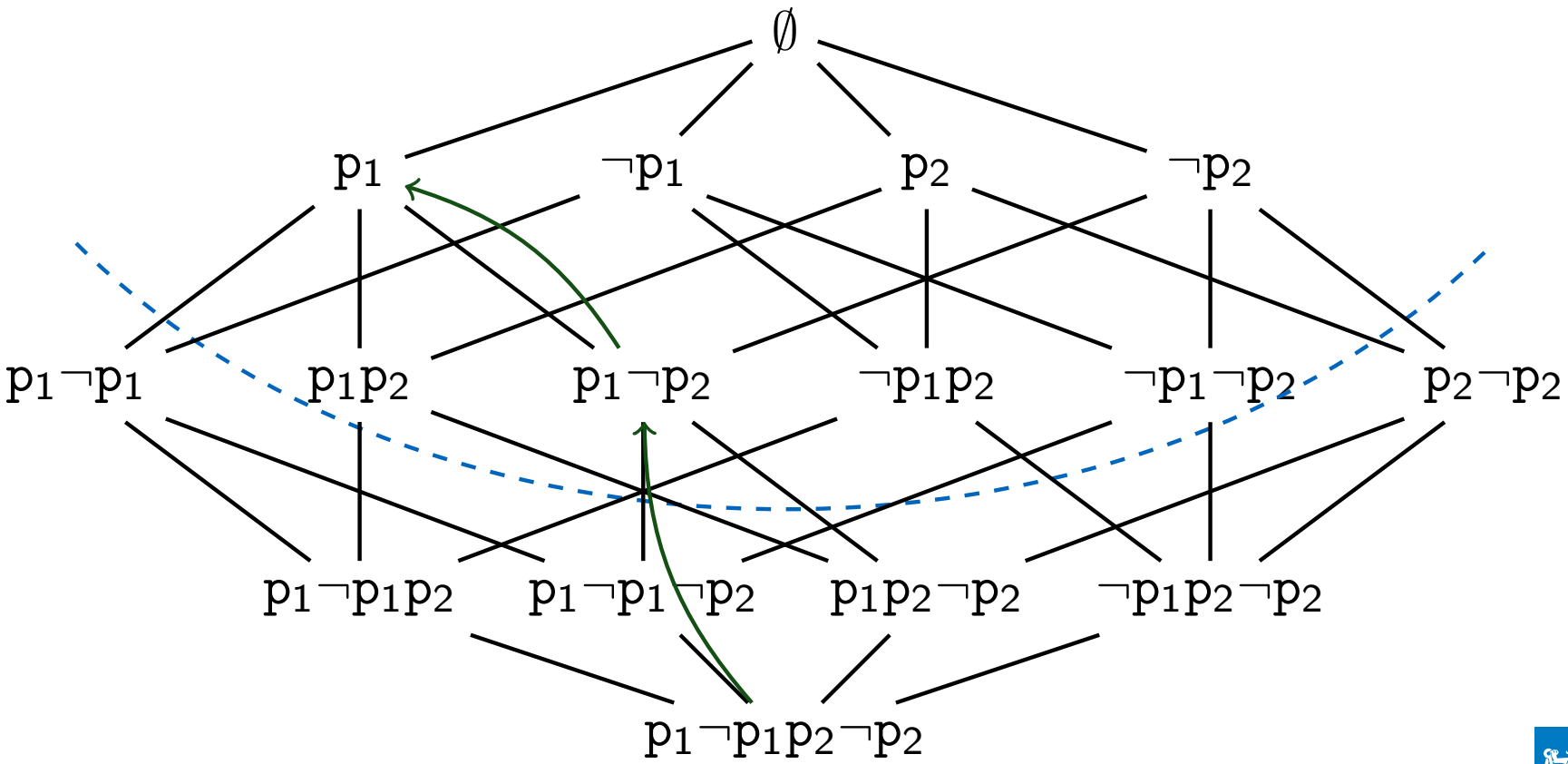
Self-resolving propositional conjunctions are trivial, they are all contradictions. (This will not be the case in in first-order logic.)

We call the last agent *generalizing* because it only *deletes* literals (never adds them), so $\forall k$, $h_{k+1} \subseteq h_k$, thus $h_k \models h_{k+1}$, thus $C(h_{k+1}) \supseteq C(h_k)$.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Subsumption Lattice

The subsumption relation forms a *lattice* (partially ordered set) on conjunctions and the agent traverses it from the bottom (e.g. along the green path). Note that all conjunctions below the dashed line are self-resolving (contradictions). On the first positive example, the agent deletes n literals from the initial hypothesis, making the next one satisfiable (non-contradictory).



Least Upper Bound

In a lattice induced by relation $\preceq$, every two elements a, b have exactly one *least upper bound* $c = \mathrm{lup}(a, b)$ such that

- $c \preceq a$ and $c \preceq b$
- there is no "lesser bound" d, i.e. no d such that $c \preceq d$, $d \not\preceq c$, $d \preceq a$, $d \preceq b$.

Properties of lup:

- $\mathrm{lup}(a, b) = a$ if $a \preceq b$
- $\mathrm{lup}(a, b) = \mathrm{lup}(b, a)$ (commutativity)
- $\mathrm{lup}(a, \mathrm{lup}(b, c)) = \mathrm{lup}(\mathrm{lup}(a, b), c)$ (associativity)

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Least General Generalization

A subsumption lattice is induced by the subset relation $\subseteq$. When the lattice members are conjunctions or disjunctions of literals, we call the lup the *least general generalization* denoted lgg and clearly

$$\mathrm{lgg}(a, b) = a \cap b \tag{3}$$

Due to commutativity and associativity of any lup, lgg is defined uniquely for any finite *set* $\mathcal{H}$ of lattice elements:

$$\mathrm{lgg}(\mathcal{H}) = \mathrm{lgg}\left(h_1, \mathrm{lgg}\left(h_2, \mathrm{lgg}\left(h_3, \ldots\right)\right)\ldots\right) \tag{4}$$

where $h_1, h_2, h_3, \ldots$ are all elements of $\mathcal{H}$ in arbitrary order.

Due to (3) and (4)

$$\mathrm{lgg}(\mathcal{H}) = \bigcap_{h \in \mathcal{H}} h$$

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Generalizing Agent: Design

Recall our Markovian agent model with the function $\mathcal{T}$ updating the agent's state given its previous state and the new percept:

$$t_{k+1} = \mathcal{T}(t_k, x_{k+1})$$

Where for each k, the state t_k contains the hypothesis h_k.

When h_k makes a mistake (i.e., $r_{k+1} = -1$), h_{k+1} should exclude all literals of h_k inconsistent with o_k. This is done at time $k+1$ so the agent needs to remember o_k as part of its state. So we will maintain the state in the form of a tuple

$$t_k = \langle h_k, o_k \rangle$$

Generalizing Agent: Design (cont'd)

The hypothesis will then be updated by

$$h_{k+1} = \begin{cases} h_k \text{ if } r_{k+1} = 0 \\ \mathrm{lgg}(h_k, \bar{o}_k) \text{ otherwise} \end{cases}$$

where $\bar{o}$ denotes a conjunction representing observation o:

$$\bar{o} = \bigwedge_{o^i=1} \mathrm{p}_i \bigwedge_{o^j=0} \neg \mathrm{p}_j$$

So e.g. for $n = 3$, $o_k = \langle 1, 0, 1 \rangle$, we have $\bar{o}_k = \mathrm{p}_1 \wedge \neg \mathrm{p}_2 \wedge \mathrm{p}_3$. Assume $h_k = \mathrm{p}_1 \wedge \mathrm{p}_2$ makes a mistake, i.e. $r_{k+1} = -1$. Then $h_{k+1} = \mathrm{lgg}(h_k, \bar{o}_k = \mathrm{p}_1$, so indeed the inconsistent literal was deleted.

Verify that generally, $lgg(h, \bar{o})$ deletes from h exactly those literals inconsistent with o.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Generalizing Agent in One Line

Since negative examples are irrelevant, assume without loss of generality that all of $o_1, \ldots, o_m$ are *positive*. Since h_1 contains all literals, we have $o_1 \subseteq h_1$ and thus

$$h_2 = \mathrm{lgg}(h_1, \bar{o}_1) = \bar{o}_1$$
$$h_3 = \mathrm{lgg}(h_2, \bar{o}_2) = \mathrm{lgg}(\mathrm{lgg}(h_1, \bar{o}_1), \bar{o}_2) = \mathrm{lgg}(\bar{o}_2, \bar{o}_1)$$

So the agent's output can be written as

$$h_m = \mathrm{lgg}(\bar{o}_{m-1}, \mathrm{lgg}(\bar{o}_{m-2}, \ldots \mathrm{lgg}(\bar{o}_2, \bar{o}_1) \ldots)) = \mathrm{lgg}(\{\, \bar{o}_1, \ldots, \bar{o}_m \,\})$$

The learned hypothesis is the least general generalization of all positive examples, which gives the intuition why negative examples are not needed: if the *least general* hypothesis was already *too general* (= covering a negative example), it means the target concept cannot be expressed through a conjunction.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Generalizing Agent for Disjunctions

Try the inverse strategy: ignore positive examples and take the lgg of the *negative examples*. Assume this time that $o_1, \ldots, o_m$ are all *negative*.

Say $h'_m = \mathrm{lgg}(\{\, \bar{o}_1, \ldots, \bar{o}_m \,\})$ covers no positives (otherwise there is no conjunction covering all negatives and no positives). Then clearly $\neg h'_m$ covers no negatives and all positives just like h_m from the previous page.

But h_m and $\neg h'_m$ are not the same: h_m is a conjunction while $\neg h'_m$ is the negation of a conjunction, i.e. a *disjunction*. So this inverse approach is suitable when the target concept can be expressed through a disjunction.

The agent can also compute both h_m and $\neg h'_m$ and output whichever one covers all positive and no negative examples. *Such an agent learns $\mathcal{H} =$ Conjunctions $\cup$ Disjunctions on n variables efficiently on-line*.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Sidenote: Non-Binary Observations

We have considered learning examples o to be binary tuples for simplicity. But what if observations are richer, e.g. tuples of rational numbers?

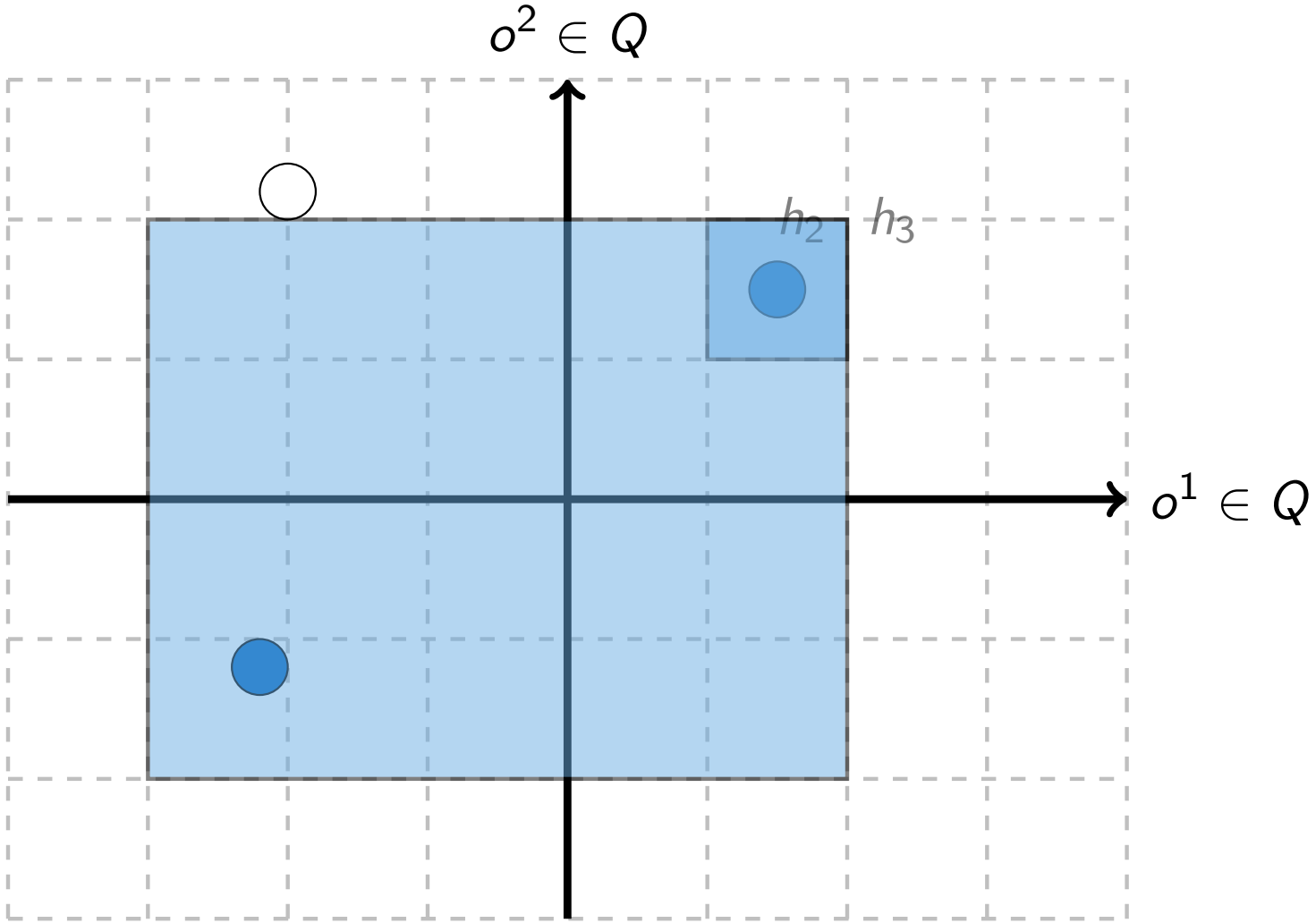
A practical way is to set *thresholds* $\theta^{i,j}$ for o^i along each axis i obtaining a (bigger) set of binary observations tuples o'.

$$o'^{1,1} = 1 \text{ iff } o^1 > \theta^{1,1} \qquad o'^{1,2} = 1 \text{ iff } o^1 > \theta^{1,2} \qquad \dots$$
$$o'^{2,1} = 1 \text{ iff } o^2 > \theta^{2,1} \qquad o'^{2,2} = 1 \text{ iff } o^2 > \theta^{2,2} \qquad \dots$$
$$\dots \qquad \dots \qquad \dots$$

Choosing good threshold is a task for *discretization* techniques which are out of our scope and out of the scope of learnability theory.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Visualizing Generalization (with Rational Features)



FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Generalizing Agent for s-CNF, s-DNF

An *s-CNF* is a conjunction of *s-clauses*, which is a disjunction of at most $s \in \mathbb{N}$ literals. s-CNF's can be learned using the generalization strategy. Given n propositional variables:

- Set h_1 to the conjunction of *all* s-clauses on these variables.
- One each positive example, remove from h_k all clauses false for the example.

Reasoning just like for the conjunction-learning agent, the number of mistakes will not be greater than the number of all s-clauses on n variables. This number is $\mathcal{O}\left[\binom{2n}{s}\right] = \mathcal{O}(n^s)$, i.e. *polynomial*. Therefore, *s-CNF's are learnable online*. Check that they are also learned *efficiently* .

Show the same for *s-DNF*, i.e. disjunctions of *s-terms*, which are conjunctions of at most s literals.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

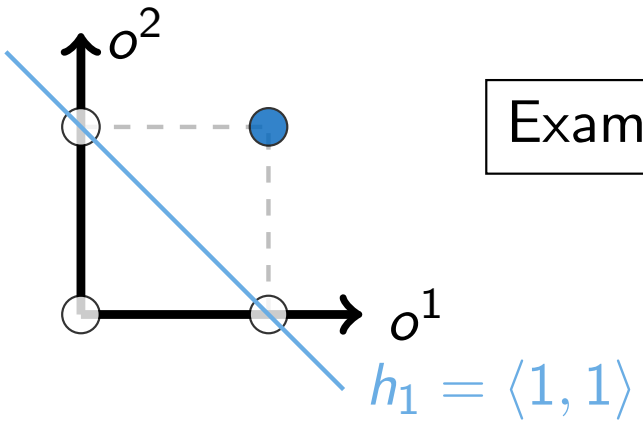
Separating agent

Besides generalization, another prominent learning technique is to *separate* C from its complement with a hyperplane in O. We continue with $O = \{ 0, 1 \}^n$. Agent's hypothesis is a tuple of integers ('weights'), i.e.

$$h_k = \left\langle h_k^1, h_k^2, \ldots, h_k^n \right\rangle$$

and $h_1 = \langle 1, 1, \ldots 1 \rangle$. Decisions are:

$$a_k = \pi(h_k, o_k) = \begin{cases} 1 \text{ if } h_k \cdot o_k > n/2 \text{ (dot product)} \\ 0 \text{ otherwise} \end{cases}$$



Example for $n = 2$

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Separating agent: Design

The agent (called Winnow in literature) has a simple learning rule:

- On a false negative o_k, *double* weights h^i for all i, $o_k^i = 1$
- On a false positive, *nullify* weights of these features.

Formally:

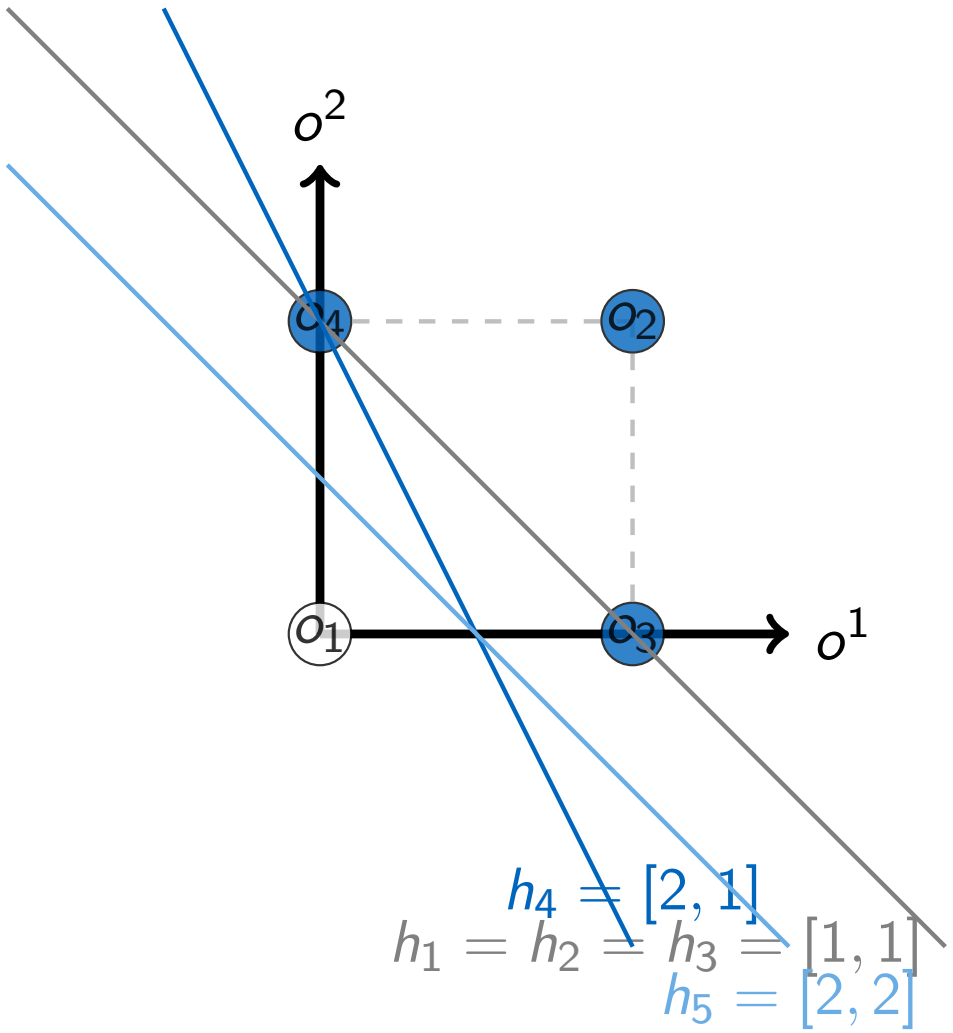
$$h_{k+1} = \begin{cases} h_k \text{ if } r_{k+1} = 0 \text{ (Keep } h_k \text{ if it did not make a mistake.)} \\ \mathrm{update}(2, h_k, o_k) \text{ if } r_{k+1} = -1 \text{ and } h_k \cdot o_k \leq n/2 \\ \mathrm{update}(0, h_k, o_k) \text{ if } r_{k+1} = -1 \text{ and } h_k \cdot o_k > n/2 \end{cases}$$

where $\mathrm{update}(\alpha, h_k, o_k)$ multiplies all features with value 1 by α, i.e.

$$h_{k+1}^i = \begin{cases} \alpha \cdot h_k^i \text{ if } o_k^i = 1 \\ h_k^i \text{ otherwise} \end{cases}$$

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Visualizing Separation



FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Separating agent: Properties

Agent's properties, omitting proofs:

- Learns efficiently any class $\mathcal{H}$ of linearly separable hypotheses. A linearly separable hypothesis h is such that $C(h)$ is separable from $O \setminus C(h)$ on O. This includes conjunctions and disjunctions.
- Let the target hypothesis h^* ($C(h^*) = C$) be an *s-conjunction* or an *s-disjunction*, i.e. a conjunction (disjunction) *with at most s literals*. Then the agent makes at most $2 + 2s \log n$ mistakes. For sufficiently small s, this is better than the generalizing agent which has a mistake bound linear in n.

Similarly to the generalizing agent, it can be extended to learning s-CNF's or s-DNF's by using the poly-size set of features corresponding to all possible s-clauses (s-terms).

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Sidenotes on the Separating agent

The perceptron algorithm is similar to the separating (Winnow) agent, using real-valued weights and a gradient-based learning rule, allowing non-binary observation vectors. Aritificial neural networks are popular machine-learning models and they are networks of perceptrons.

Numerous machine-learning algorithms including support vector machines follow the separation-by-hyperplane strategy. Non-linear separation (by a hypercurve) can be achieved by a suitable expansion of the observation vectors such as by adding to them cross-products $o^i o^j$ of all feature pairs.

These methods are our of our scope (find them in the Statistical ML class instead) as we are concerned mainly with symbolic, interpretable hypotheses.

General Learning Agent

Consider an agent *general* in the sense that it applies the same strategy using any given hypothesis class $\mathcal{H}$. It keeps *a set* $\mathcal{H}_k$ of hypotheses, rather than a single one starting with $\mathcal{H}_1 = \mathcal{H}$.

At each k, the agent picks an arbitrary h from $\mathcal{H}_k$ and makes a decision by it. If it makes an error, it deletes h from $\mathcal{H}_k$. Formally,

$$\mathcal{H}_{k+1} = \begin{cases} \mathcal{H}_k \text{ if } r_{k+1} = 0 \text{ (Keep } \mathcal{H}_k \text{ if it did not make a mistake.)} \\ \mathcal{H}_k \setminus \{ h \} \text{ if } r_{k+1} = -1 \end{cases}$$

If $C \in \mathcal{C}(\mathcal{H})$, then this agent makes at most $|\mathcal{H}| - 1$ mistakes. (If all hypotheses made a mistake, the last one must be the target.)

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

General Learning Agent

So the agent learns online (not necessarily efficiently) any $\mathcal{H}$ such that $|\mathcal{H}| \leq poly(n)$, including

- conjunctions: $|\mathcal{H}| = 2^{2n}$ (resp. $|\mathcal{H}| = 3^n$ leaving out self-resolving).
- s-conjunctions

$$|\mathcal{H}| = \binom{n}{s} + \binom{n}{s-1} + \ldots + \binom{n}{0} \leq poly(n)$$

- disjunctions and s-disjunctions for the same reasons.

but not e.g.

- s-clause CNF or s-term DNF:

$$|\mathcal{H}| = \mathcal{O}\left[(3^n)^s\right]$$

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Version Space Agent

The version space agent is similar to the general agent but decides by a majority vote among all hypotheses in $\mathcal{H}_k$ and deletes from $\mathcal{H}_k$ *all* hypotheses inconsistent with the last observation. The agent's state t_k consists of $\mathcal{H}_k$ and the memorized observation o_k.

Specifically, when $\mathcal{H}_k$ contains logical formulas and o_k are valuations:

$$a_k = \pi(\mathcal{H}_k, o_k) = \begin{cases} 1 \text{ if } |\{\, h \in \mathcal{H}_k \mid o_k \models h \,\}| > |\mathcal{H}_k|/2 \\ 0 \text{ otherwise} \end{cases}$$

$$\mathcal{H}_{k+1} = \begin{cases} \{\, h \in \mathcal{H}_k \mid o_k \models h \,\} & \text{if } s_k = 1 \\ \{\, h \in \mathcal{H}_k \mid o_k \not\models h \,\} & \text{if } s_k = 0 \end{cases}$$

Note: $s_{k+1} = |a_k + r_{k+1}|$ where $a_k = \pi(\mathcal{H}_k, o_k)$, so the above can indeed be evaluated at update time.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Version Space Agent

If a mistake is made, at least half of the hypotheses are $\mathcal{H}_k$ are deleted. In the worst case, the last remaining hypothesis is correct.

So the agent makes at most $\log |\mathcal{H}|$ mistakes, i.e. the cumulative reward is

$$\sum_{k=1}^{m} r_k \geq -\log |\mathcal{H}| \tag{5}$$

for any horizon $m \in N$.

Version space *not efficient* even for super-poly (not just super-exp) $\mathcal{H}$. Needs to verify each $h \in \mathcal{H}$ in the update step.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

On-Line Learning with Version Space

If $|\mathcal{H}|$ at most exponential then $\log |\mathcal{H}|$ polynomial and VS agent learns $\mathcal{H}$ online. This includes s-clause CNF's and s-term DNF's.

What about $\mathcal{H}$ covering all possible concepts on $O = \{ 0, 1 \}^n$, i.e.

$$\mathcal{C}(\mathcal{H}) = 2^O$$

We would not need to worry whether $C \in \mathcal{C}(\mathcal{H})$.

Since $|O| = 2^n$, we have $|\mathcal{H}| \geq 2^{|O|} = 2^{(2^n)}$, so $|\mathcal{H}|$ is super-exponential. So, nice try but no on-line learning.

Even some hypothesis classes which are more reasonable are super-exponential (we will see later).

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

VC Dimension

Concept class $\mathcal{C}$ *shatters* $O' \subseteq O$ if any subset of O' coincides with $C \cap O'$ where $C \in \mathcal{C}$. A hypothesis class $\mathcal{H}$ shatters O' if $\mathcal{C}(\mathcal{H})$ shatters O'.

So O' is shattered by $\mathcal{C}$ (resp. $\mathcal{H}$) if its elements can be classified in all $2^{O'}$ possible ways by concepts from $\mathcal{C}$ (hypotheses from $\mathcal{H}$).

Vapnik-Chervonenkis Dimension

The *VC-dimension* of $\mathcal{C}$ (on O) denoted $\mathrm{VC}(\mathcal{C})$ is the cardinality of the largest subset of O shattered by $\mathcal{C}$. The VC-dimension of hypothesis class $\mathcal{H}$ is defined as $\mathrm{VC}(\mathcal{C}(\mathcal{H}))$, also denoted $\mathrm{VC}(\mathcal{H})$.

Note: the definition does not assume $\mathcal{C}$ or $\mathcal{H}$ finite.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Lower Bounds on Mistake Bounds

No general lower bound on mistake *counts* as the agent may simply be lucky guessing right each time. But mistake *bounds* can be lower-bounded.

A mistake bound with no special assumptions cannot be lower than $|O|$ as each $o \in O$ may have an arbitrary class.

A mistake bound assuming only $C \in \mathcal{C} \subset 2^O$ for the target concept $\mathcal{C}$ cannot be smaller than $\mathrm{VC}(\mathcal{C})$ as there is a set $\left\{ o_1, o_2, \ldots, o_{\mathrm{VC}(\mathcal{C})} \right\} \subseteq O$ shattered by $\mathcal{C}$. So for the observation sequence $o_1, o_2, \ldots, o_{\mathrm{VC}(\mathcal{C})}$ and any sequence of agent's decisions $a_1, a_2, \ldots, a_{\mathrm{VC}(\mathcal{C})}$ there is a target concept $C \in \mathcal{C}$ by which all these decisions are wrong.

Corollary: a mistake bound assuming only a hypothesis-based agent with hyp. class $\mathcal{H}$ and $C \in \mathcal{C}(\mathcal{H})$ for the target concept C cannot be smaller than $\mathrm{VC}(\mathcal{H})$.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE