Classification

Classification is a special case of the agent-environment interaction defined by two assumptions

1. Y is finite
2. rewards are instant (??)

Elements of Y are called *classes*.

Similarly, for 'regression' we would replace the first condition with $Y = \mathbb{R}$. We do not elaborate regression in this course.

(exercise problem)

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Example: Classification of Handwritten Numbers

$X = \mathbb{R}^{16\times 16}$ (x are pixel vectors)
$Y = \{\, 0, \ldots, 9 \,\}$

When training a classification agent, if we know the 'true' classes $\overline{y}(x)$ of observations, we may use them to prescribe rewards by function $r : Y \times Y \to R$, such as

$$r_{k+1} = r(y_{k+1}, \overline{y}(x_k)) = -|y_{k+1} - \overline{y}(x_k)|$$

$$r_{k+1} = r(y_{k+1}, y^*(x_k)) = \begin{cases} 0 \text{ if } y_{k+1} = \overline{y}(x_k) \\ -1 \text{ otherwise} \end{cases} \quad \text{(Unit reward)} \qquad (1)$$

The negative reward function $-r(.,.)$ is called a **loss function**.

Concept Classification

We will now focus on the simplest interesting form of classification: only two classes and no "noise".

Formally, any subset $C \subseteq X$ is called a **concept on X** and we define **concept classification** as a special case of classification where $Y = \{0, 1\}$ there is a **target concept** C on X instantiating the rewards (??) as ($k \in \mathbb{N}$)

$$r_1 = 0$$

$$r_{k+1} = \begin{cases} y_{k+1} - 1 \text{ if } x_k \in C \\ -y_{k+1} \text{ otherwise} \end{cases} \tag{2}$$

Concept Classification (cont'd)

In other words, the agent decides by $y_{k+1} = 1$ ($y_{k+1} = 0$) that $x_k \in C$ ($x_k \notin C$, respectively) and gets reward $r_{k+1} = 0$ ($r_{k+1} = -1$) if the decision was right (wrong, respectively).

(Exercise problem)

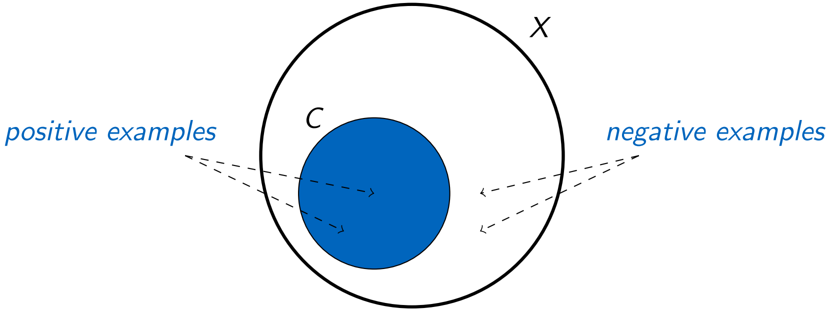
Rewards r_{k+1} here depend deterministically on x_k and y_{k+1}, hence no "noise".

Note that arbitrary classification can be done by a finite number of concept classification agents. Indeed, since Y is finite, each $y \in Y$ can be represented by a binary number with $n \approx \lg |Y|$ digits. So we just let the target concept for the i's agent contain all $x \in X$ for which the i's digit of optimal action $\overline{y}$ is 1. This agent will learn to predict the i's digit of the optimal action for x.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Positive and Negative Examples

Observations $x \in C$ ($x \notin C$, respectively) received by the agent are called **positive (negative) examples** of C.



Example: $X \sim$ descriptons of animals. $C \sim$ same for mammals. Positive example: description of a cat, negative example: same for a chicken.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Hypothesis

From the examples, the agent learns a **hypothesis** h, which is a *finite-size* description of a binary policy. The hypothesis also induces a concept

$$C(h) = \{ x \in X \mid h(x) = 1 \} \tag{3}$$

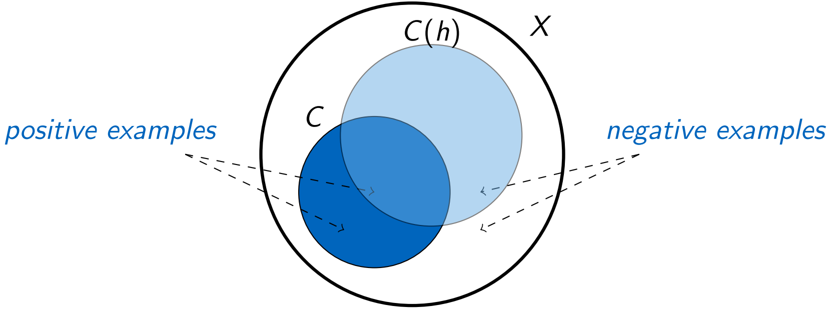
Note that we overload the h symbol to denote both the description of the hypothesis and the policy function it defines.

(3) depends on the way the function $h(x)$ is determined from the description h. We do not make this dependence explicit as we will only be interested in hypotheses with an obvious functional interpretation.

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Hypothesis vs. Concept

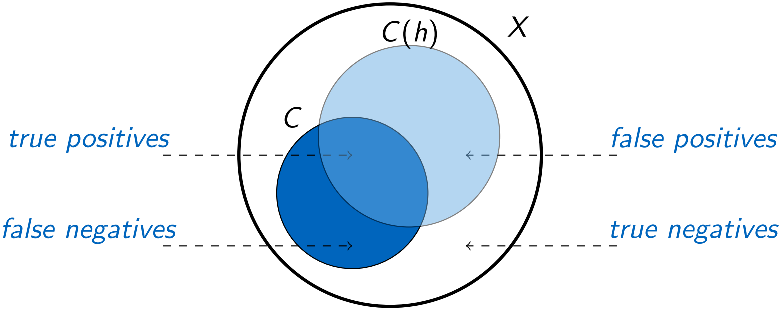
The goal of learning is to find h such that $C(h) = C$.



Example (logical): $h = \mathrm{milk} \wedge \neg\mathrm{feathers}$, $h(x) = 1$ iff $x \models h$.

Error Types

Until $C(h) = C$, there are four kinds of observations



False positives and false negatives form the *error region*.

Concept Class

With an unlimited supply of non-repeated examples, can we always learn the target concept, i.e. find a h such that

$$C(h) = C \tag{4}$$

In general, *no*. There are $2^{|X|}$ possible concepts on X. If X is infinite (e.g. $X = \mathbb{N}$), there is an uncountable ($|\mathbb{R}|$) number of such concepts. A hypothesis is a finite description so there is a countable number ($|\mathbb{N}|$) of hypotheses. Thus there are more concepts than hypotheses.

To allow any learnability results, we will always have to assume that C is not arbitrary ($C \in 2^X$) but belongs to a smaller **concept class on X**

$$\mathcal{C} \subset 2^X \tag{5}$$

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Hypothesis Class

A **hypothesis class** $\mathcal{H}$ is a set of hypotheses. For example, a set of *propositional-logic conjunctions*.

$\mathcal{C}(\mathcal{H})$ denotes the concept class on X induced by hypotheses in $\mathcal{H}$, i.e.

$$\mathcal{C}(\mathcal{H}) = \{ \, C(h) \mid h \in \mathcal{H} \, \} \tag{6}$$

So if $\mathcal{H}$ = propositional-logic conjunctions then $\mathcal{C}(\mathcal{H})$ means the set of all concepts on X that can be described by such conjunctions.

Terminology: when there is no risk of confusion, we will call $\mathcal{C}(\mathcal{H})$ the same as $\mathcal{H}$, e.g. "conjunctions " rather than "concept class on X induced by conjuctions". If $C(\overline{h})$ is the target concept, we will call $\overline{h}$ the **target hypothesis**.

Mistake-Bound Learning Model

Given a concept class $\mathcal{C}$ we want to study whether a learning agent can learn concepts from $\mathcal{C}$. What does "can learn concepts from $\mathcal{C}$" exactly mean? One definition is provided by the **mistake-bound** learning model also known as the **online learning** model.

Due to (2), maximizing the utilities (??) or (??) means minimizing the number of mistakes, i.e., actions followed immediately by reward $r = -1$. But what utility value is considered a success?

In the mistake-bound model, we request that if the target concept $C \in \mathcal{C}$, the number of mistakes is *finite* even for an infinite time horizon m, and this is true for *any distribution of observations* (??).

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Mistake-Bound Learning Model (cont'd)

Given (2), the total number of mistakes in one possible history $xr_{\leq k}$ is $\sum_{k=1}^{\infty} |r_k|$. Since the latter must be finite, $\sum_{k=1}^{\infty} r\gamma^k$ converges even with $\gamma = 1$ (we will keep $\gamma = 1$ unless stated otherwise).

As this must be true for any distribution of observations (??), the expectation in the infinite utility (??) also converges and $|U^{y_1, y_2, \cdots}|$ is the expected total number of mistakes.

Recall that the sequence of observations determined by distribution (??) is the only source of randomness in the agent-environment interaction in concept classification as (??) is set deterministically by (2).

Mistake-Bound Learning Model (cont'd)

Moreover, the model requests that the number of mistakes is not just finite, but reasonably small. In particular, it should grow at most *polynomially* with the *size (descriptive complexity)* of observations $x \in X$, denoted n_X. When observations are feature tuples of dimension n, we will always set $n_X = n$.

The model also defines when concept learning is time-efficient.

Mistake-bound model (or Online learning model)

In the concept classification protocol, an agent **learns $\mathcal{C}$ from X online** if for any target concept from $\mathcal{C}$ on X and an arbitrary distribution (??), it makes a sequence of decisions $y_1, y_2, \ldots$ such that $\sum_{k=1}^{\infty} |r_k| \leq poly(n_X)$. It learns $\mathcal{C}$ online from X **efficiently**, if in addition, the time taken to compute an action from an observation is also at most polynomial in n_X.

(*Exercise problem*)

The Winnow Algorithm

Winnow assumes Boolean-tuple observations, i.e. $X = \{0, 1\}^n, n \in \mathbb{N}$ and tries to identify the target concept $C \subseteq X$ by a hyperplane in X.

The agent's hypothesis at time k is an n-tuple of integers $h_k = (h_k^1, h_k^2, \ldots, h_k^n)$ specifying the hyperplane, initially set to

$$h_1 = (1, 1, \ldots, 1) \tag{7}$$

Its policy $y_{k+1} = h_k(x_k)$ $(k \in \mathbb{N})$ is given by

$$h_k(x_k) = 1 \text{ iff } \sum_{i=1}^{n} h_k^i x_k^i > \frac{n}{2} \tag{8}$$

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

The Winnow Algorithm (cont'd)

On each mistake, h_k is updated to h_{k+1} with a simple learning rule:

- On a false negative x_k ($y_{k+1} = 0$, $r_{k+1} = -1$), *promote* each component h_k^i where $x_k^i = 1$ by doubling its value:

$$h_{k+1}^i = 2h_k^i \tag{9}$$

- On a false positive ($y_{k+1} = 1$, $r_{k+1} = -1$), *eliminate* each component h_k^i where $x_k^i = 1$ by zeroing it:

$$h_{k+1}^i = 0 \tag{10}$$

Winnow Learning Monotone Disjunctions

Using hyperplane separation, Winnow can learn only classes of linearly separable concepts. One example is the class $\mathcal{C}(\mathcal{H})$ of *monotone disjunctions* made out of up to n propositional variables $\mathrm{p}_1, \mathrm{p}_2, \ldots \mathrm{p}_n$, i.e.,

$$\mathcal{H} = \{ \mathrm{p}_{i_1} \vee \mathrm{p}_{i_2} \vee \ldots \mathrm{p}_{i_s} \mid 1 \leq i_j \leq n \}$$

So, for example, when $n = 4$ and the target hypothesis is $\mathrm{p}_1 \vee \mathrm{p}_3$, the agent's hypothesis $h = [2, 0, 1, 0]$ will not make any mistakes because the target disjunction is true iff

$$2x^1 + x^3 > 2$$

We are putting component indexes to the superscript of x and h_k, reserving the subscript for a time index.

Conjunctive Observations

For Winnow, we assumed $X = \{0, 1\}^n$, $n \in \mathbb{N}$, so an example $x \in X$ specifies *each* of the n Boolean values. We want to allow the case that x does not specify some of them. This could be done by leting $X = \{0, 1, ?\}^n$ where ? stands for *value uknown*.

Another way is to let X be the set of **contingent** (not tautologically true or false) *conjunctions* of propositional literals made of atoms selected from $\mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p}_n$. For example, with $n = 3$ the observation

$$\mathrm{p}_1 \wedge \neg \mathrm{p}_3$$

represents the same information as

$$(1, ?, 0)$$

We define the complexity n_X of such observations to be n.

(*Exercise problem*)

Conjunctive Hypotheses

Unless stated otherwise, the term conjunction (disjunction) will mean a conjunction (disjunction) of *propositional literals*, excluding e.g. a conjunction of disjunctions (or the reverse). Non-propositional cases will be marked explicitly.

We will be interested in hypotheses which are conjunctions, providing the following decision policy $y = h(x)$ for conjunctive examples

$$h(x) = 1 \text{ iff } x \models h \tag{11}$$

where $\models$ is *tautological consequence*. So e.g. the observation

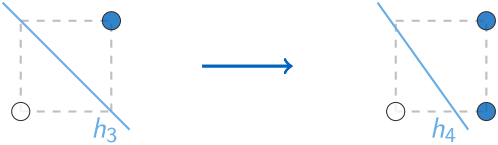
$$x = \mathrm{milk} \wedge \neg \mathrm{feathers} \wedge \neg \mathrm{flies}$$

is decided positively ($h(x) = 1$) by $h = \mathrm{milk} \wedge \neg \mathrm{feathers}$.

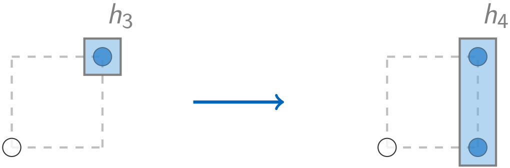
FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Separation vs. Generalization

Winnow uses the popular learning technique of *separation*



An alternative is the "covering" approach seeking the smallest joint *generalization* of positive examples



FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Generality and Subsumption Order

Let π, π' be two policies $X \to \{0, 1\}^n$. We say that y is **at least as general as** y' if $\pi(x) = 1$ for any $x \in X$ such that $\pi'(x) = 1$.

Let h, h' be conjunctions that prescribe policies by (11). If $h' \models h$ then h is at least as general as h'. (*exercise problem*)

Let h, h' be two conjunctions or two disjunctions. We say that h **subsumes** h' (written $h \subseteq h'$) if $\mathrm{Lits}(h) \subseteq \mathrm{Lits}(h')$ where $\mathrm{Lits}(c)$ denotes the set of literals in c. We say that h **strictly subsumes** (written $h \subset h'$) h' if $\mathrm{Lits}(h) \subset \mathrm{Lits}(h')$.

Theorem 1

Let h, h' be conjunctions. If $h \subseteq h'$ then $h' \models h$. Let furthermore h' not be tautologically false. Then $h \subseteq h'$ if and only if $h' \models h$.

(*exercise problem*)

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Least General Generalization

Let h, h' be two conjunctions (two disjunctions, respectively). We say that g is a **least general generalization** of h and h' if $g \subseteq h$, $g \subseteq h'$, and there is no conjunction (disjunction) g' such that $g \subset g'$, $g' \subseteq h$, $g' \subseteq h'$.

Let h, h' be two conjunctions (two disjunctions, respectively) and let us define:

$$\text{lgg}(\mathbf{h}, \mathbf{h'}) = \text{Lits}(h) \cap \text{Lits}(h') \tag{12}$$

Easy to verify: $\text{lgg}(h, h')$ is a least general generalization of h and h'.

The proof is an exercise problem.

(*a further exercise problem*)

LGG Properties and Set-LGG

The subsumption order $\subseteq$ induces a *lattice* where lgg is the *least upper bound* (lup). As any lup, lgg has these properties:

$$\text{lgg}(a, b) = a \text{ if } a \subseteq b \tag{13}$$

$$\text{lgg}(a, b) = \text{lgg}(b, a) \text{ (commutativity)} \tag{14}$$

$$\text{lgg}(a, \text{lgg}(b, c)) = \text{lgg}(\text{lgg}(a, b), c) \text{ (associativity)} \tag{15}$$

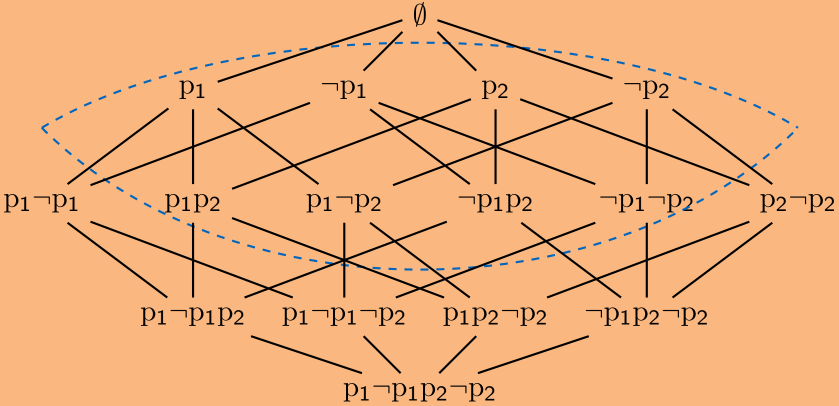
Properties 14 and 15 let us extend lgg naturally to *sets* of conjunctions or disjunctions:

$$\text{lgg}(\{ x_1, x_2, \ldots, x_m \}) = \text{lgg}(\ldots \text{lgg}(\text{lgg}(x_1, x_2), \ldots), x_m) \tag{16}$$

where the order of the x_k on the RHS is irrelevant. For conjunctions, obviously $\text{lgg}(\{ x_1, x_2, \ldots, x_m \}) = \bigcap_{k=1}^{m} x_k$.

Example: Subsumption Lattice on Conjunctions

Contingent conjunctions are enclosed by the dashed curve.



The Generalization Algorithm

The generalization algorithm assumes X to consist of contingent conjunctions on variables $\mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p}_n$. It uses the policy (11), which can be written as

$$h(x) = 1 \text{ if } h \subseteq x \tag{17}$$

because $x \in X$ are contingent. It has a simple learning rule:

- Wait for the first positive example. That is, emit actions $y_k = 0$ until $r_k = -1$, then x_{k-1} is a positive example. Set $h_k = x_{k-1}$.
- Continue receiving percepts and on each mistake ($r_{k+1} = -1$), set

$$h_{k+1} = \mathrm{lgg}(h_k, x_k) \tag{18}$$

FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE