Advanced Machine Learning

Bogdan Alexe,

bogdan.alexe@fmi.unibuc.ro

University of Bucharest, 2nd semester, 2019-2020

Administrative

- first seminar class today, 10-12, room3. I’ve uploaded the exercises for the first seminar (taken from the course book).

- Tuesday, 8-10, room 3 the same seminar. Each week?

- the first seminar in week 2 + 3

Today’s lecture: Overview

- A Formal Model – The Statistical learning framework

- Empirical Risk Minimization

- Probably Approximately Correct learning

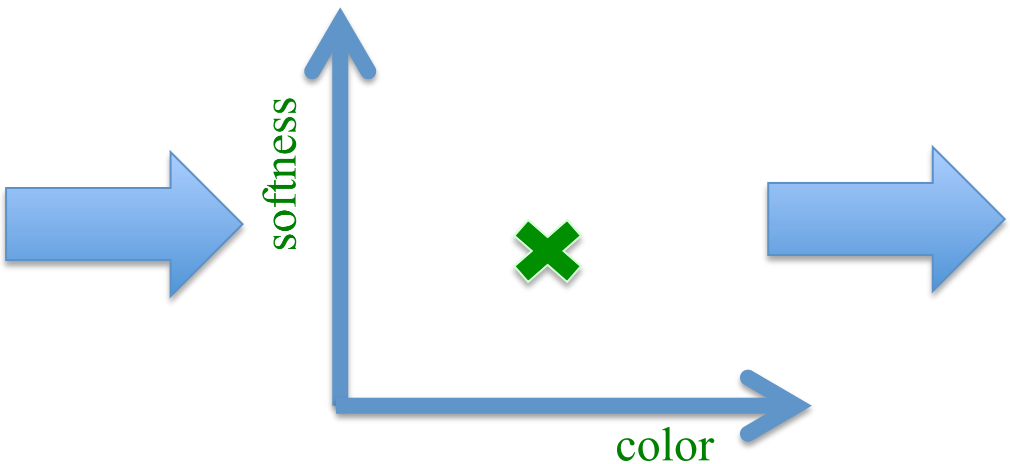
- The general Probably Approximately Correct learning model

Particular learning scenario – papaya tasting

You've just arrived in some small Pacific island

You soon find that papayas are a significant ingredient in the local diet

Based on previous experience with other fruits, you decide to use two features



{Tasty/Not tasty}

Formal model for statistical learning

- domain set, $\mathcal{X}$: this is the set of objects that we may wish to label.
  - instance space
  - elements of $\mathcal{X}$ are called instances, represented by *features*
- label set, $\mathcal{Y}$: the set of possible labels.
  - usually {0,1} or {-1, +1}
- training data S
  - the learner's input
  - supervised batch learning scenario
  - finite sequence of pairs S of labeled domain points: $S = ((x_1, y_1) \ldots (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$

Example

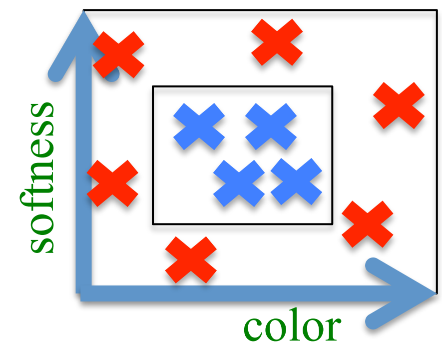
- $\mathcal{X} = \mathbf{R}^2$ representing color and shape of papays.
- $\mathcal{Y} = \{ 0, 1\}$ representing "tasty" (1) or "non-tasty"(0)
- S = set of already tasted papayas and their color, softness and tastiness (label)

Formal model for statistical learning

- a prediction rule, $h : \mathcal{X} \rightarrow \mathcal{Y}$
  - the learner's output;
  - used to label future examples;
  - called a *predictor*, a *hypothesis*, or a *classifier;*
  - $h = A(S)$: the hypothesis h is returned by a learning algorithm A based on the training sequence S
  - *goal of the learner: h* should be *correct on future examples*

Example

- prediction rule for tastiness
- $h(x) = 1$ if x = [color, shape] within the inner rectangle,
- $h(x) = 0$ if x = [color, shape] outside the inner rectangle.



“Correct on future examples”

- let f be the correct classifier, then we should find h such that $h \approx f$

- one way: define the error of h w.r.t. f to be the probability that it does not predict the correct label on a random data point x generated by the underlying (unknown) probability distribution $\mathcal{D}$ over $\mathcal{X}$:

$$L_{\mathcal{D},f}(h) = \underset{x \sim \mathcal{D}}{\mathbb{P}}[h(x) \neq f(x)]$$

- more formally, $\mathcal{D}$ is a probability distribution over $\mathcal{X}$, that is, for a given $A \subset \mathcal{X}$, the value of $\mathcal{D}(A)$ is the probability to observe some $x \in A$. Then:

$$L_{\mathcal{D},f}(h) \stackrel{\text{def}}{=} \underset{x \sim \mathcal{D}}{\mathbb{P}}[h(x) \neq f(x)] \stackrel{\text{def}}{=} \mathcal{D}(\{x \in \mathcal{X} : h(x) \neq f(x)\})$$

- can we find h s.t. $L_{\mathcal{D},f}(h)$ is small ?
  - $L_{\mathcal{D},f}(h)$ is called ***generalization error***, the ***true error*** of h, the ***real risk*** of h
  - L - loss of the learner

Data-generation model

- we must assume some relation between the training data S and $\mathcal{D}$, f

- simple data generation model:
  - Independently Identically Distributed (i.i.d.): Each x_i is sampled independently according to $\mathcal{D}$.
    - we do not assume that that the learner knows anything about $\mathcal{D}$.
  - Realizability: assume that there is a "correct" labeling function, $f : \mathcal{X} \to \mathcal{Y}$ and that for every $i \in 1, 2, \ldots, m$, $y_i = f(x_i)$
    - f is a deterministic labeling – strong assumption
    - relax this assumption later (make f probabilistic labeling)
    - we do not assume that the learner knows anything about f.

- each pair in the training data S is generated by first sampling i.i.d. a point x_i according to $\mathcal{D}$ and then labeling it by f.

Empirical Risk Minimization

The ERM learning paradigm

- $h_S = A(S)$: the hypothesis h_S is returned by a learning algorithm A based on the training sequence S, sampled i.i.d from an unknown distribution $\mathcal{D}$ and labeled by some target function f

- the true error $L_{\mathcal{D},f}(h)$ is unknown to the learner as he doesn't know $\mathcal{D}$ and f

- the learner has acces to the *training error* = *empirical error* = *empirical risk*:

$$L_S(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m},$$

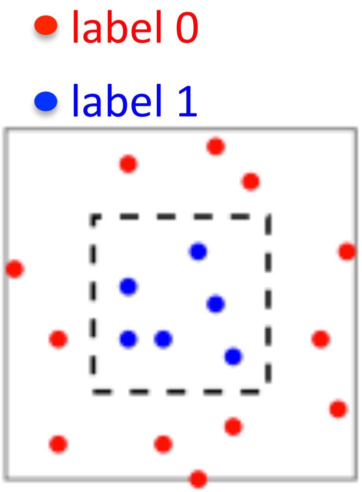
- search for a solution that works well on the training data – minimize $L_S(h)$
- *Empirical Risk Minimization* (ERM) = learning paradigm that returns a predictor h that minimizes $L_S(h)$

ERM might overfit

- simple rule for finding h with small empirical error:

$$h_S(x) = \begin{cases} y_i & \text{if } \exists i \in [m] \text{ s.t. } x_i = x \\ 0 & \text{otherwise.} \end{cases}$$

- consider:
  - $\mathcal{D}$ uniform over the gray rectangle
  - all instances in the inner rectangle are labeled 1 (**blue points**)
  - all other instances in the gray rectangle but outside the inner one are labeled 0 (**red points**)
  - area of the gray square is 2, area of the inner rectangle is 1



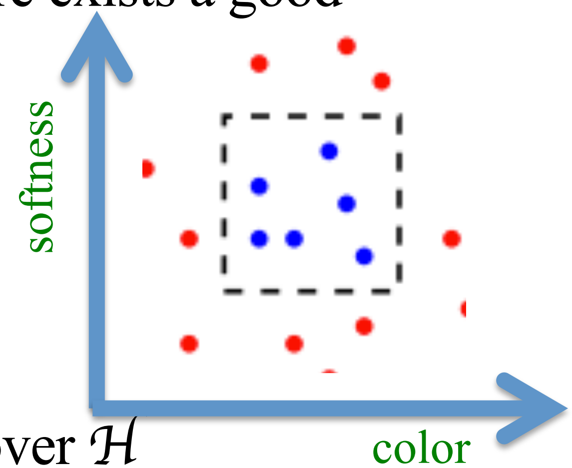
- we obtain that $L_S(h_S) = 0$, but $L_{\mathcal{D},f}(h_S) = ½$ (h predicts the label 1 on a finite number of instances)
- *overfitting*: small training error, large true error

- while the above predictor h_S seems to be very unnatural, it can be described as a thresholded polynomial (seminar exercise)

ERM with Inductive Bias

- to guard against overfitting we introduce some prior knowledge (inductive bias)

- example for the papaya prediction tasting problem: there exists a good prediction rule that is some axis aligned rectangle



- restricted search space: set of all rectangles

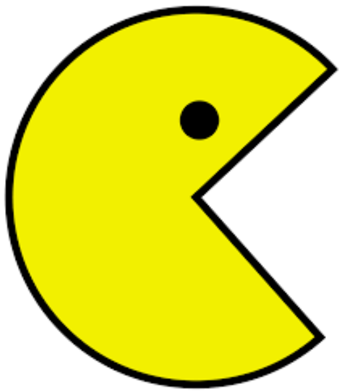
- hypothesis class = $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}} = \{g : \mathcal{X} \rightarrow \mathcal{Y}\}$

- revised ERM rule: apply the ERM learning paradigm over $\mathcal{H}$
  - for the training sample S, the $\mathrm{ERM}_{\mathcal{H}}$ learner chooses a predictor $h \in \mathcal{H}$ with the lowest possible error over S:

$$\mathrm{ERM}_{\mathcal{H}}(S) \in \underset{h \in \mathcal{H}}{\mathrm{argmin}}\, L_S(h),$$

(Regression, SGD, etc)

- over which hypothesis classes $\mathrm{ERM}_{\mathcal{H}}$ will not result in overfitting?
  - what characterizes a good hypothesis class?



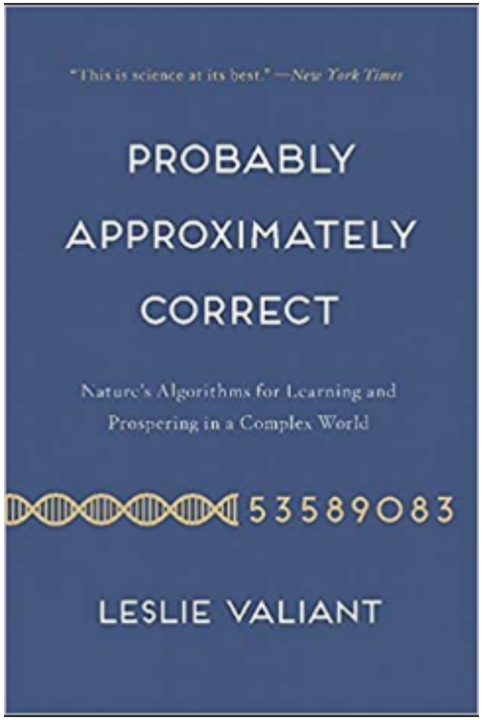
PAC learning

(Probably Approximately Correct learning)

PAC learning

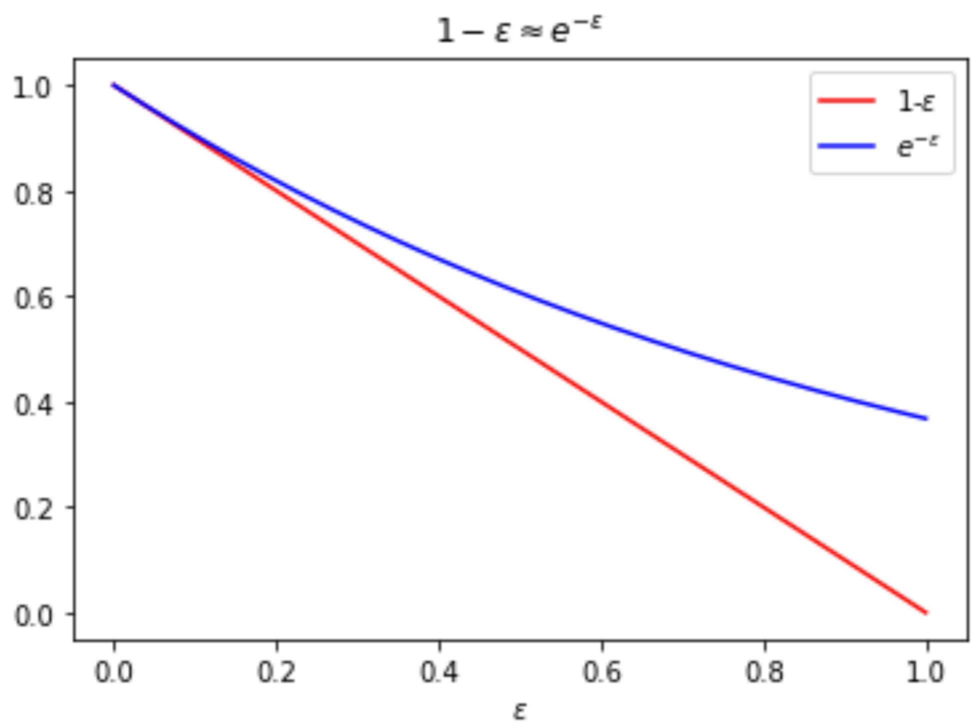
Leslie Valiant, Turing award 2010

For transformative contributions to the theory of computation, including the theory of probably approximately correct (PAC) learning, the complexity of enumeration and of algebraic computation, and the theory of parallel and distributed computing.



Can only be *Approximately* correct

- claim: we can't hope to find h_S s.t. $L_{(\mathcal{D},f)}(h_S) = 0$
- proof:
  - for every $\varepsilon \in (0,1)$ take $\mathcal{X} = \{x_1, x_2\}$ and $\mathcal{D}(\{x_1\}) = 1-\varepsilon$, $\mathcal{D}(\{x_2\}) = \varepsilon$
  - the probability not to see x_2 at all among m i.i.d. examples from S is $(1-\varepsilon)^m \approx e^{-\varepsilon m}$



Can only be *Approximately* correct

- claim: we can't hope to find h_S s.t. $L_{(\mathcal{D},f)}(h_S) = 0$
- proof:
  - for every $\varepsilon \in (0,1)$ take $\mathcal{X} = \{x_1, x_2\}$ and $\mathcal{D}(\{x_1\}) = 1 - \varepsilon$, $\mathcal{D}(\{x_2\}) = \varepsilon$
  - the probability not to see x_2 at all among m i.i.d. examples from S is $(1-\varepsilon)^m \approx e^{-\varepsilon m}$

- so, if $\varepsilon \ll 1/m$ we're likely not to see x_2 at all, but then we can't know its label

- relaxation: we'd be happy with $L_{(\mathcal{D},f)}(h_S) \leq \varepsilon$, where ε is a parameter user-specified
- ε is the accuracy parameter
- ε measures the quality of our prediction

Can only be *Probably* correct

- recall that the input to the learner is randomly generated
- there's always a (very small) chance to see the same example again and again
  - all the papayas we have happened to taste were not tasty,
  - we will come up with the classifier that assigns label 0 = Not Tasty to every future sample
  - will lead to large true error

- claim: no algorithm can guarantee $L_{(\mathcal{D},f)}(h_S) \leq \varepsilon$ for sure
  - address the probability to sample a training set S for which $L_{(\mathcal{D},f)}(h_S) \leq \varepsilon$

- relaxation: we'd allow the algorithm to fail with probability δ, where $\delta \in (0, 1)$

is user-specified

- here, the probability δ is over the random choice of examples
- $1 - \delta$ is the confidence parameter
- δ measures the probability of getting a nonrepresentative sample

*P*robably **A**pproximately **C**orrect (PAC) learning

- the learner doesn't know $\mathcal{D}$ and f
- the learner receives accuracy parameter ε and confidence parameter δ

- the learner can ask for training data, S, containing $m(\varepsilon, \delta)$ examples (that is, the number of examples can depend on the value of ε and δ, but it can't depend on $\mathcal{D}$ or f).

- learner should output a hypothesis h_S s.t. with probability of at least $1-\delta$ it holds that $L_{\mathrm{D},f}(h_S) \leq \varepsilon$

- that is, the learner should be **P**robably (with probability at least $1-\delta$) **A**pproximately (up to accuracy ε) **C**orrect

Learning finite classes

- give more knowledge to the learner: the target f comes from some hypothesis class, $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$
- the learner knows $\mathcal{H}$

- assume that H is a finite hypothesis class
  - example: $\mathcal{H}$ is all the functions from $\mathcal{X}$ to $\mathcal{Y}$ that can be implemented using a Python program of length at most b

- use the consistent learning rule:
  - input: $\mathcal{H}$ and $S = (x_1, y_1), \ldots, (x_m, y_m)$
  - output: $h \in \mathcal{H}$, h is an ERM hypothesis

- Empirical Risk Minimization (ERM)
  - input: training set $S = (x_1, y_1), \ldots, (x_m, y_m)$
  - define the empirical risk $L_S(h)$: $L_S(h) = \dfrac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m}$,
  - output: any $h \in \mathcal{H}$ that minimizes $L_S(h)$

The Realizability Assumption

- assume that the exists $h^* \in \mathcal{H}$ such that $L_{\mathcal{D},f}(h^*) = 0.$
  - implies that with probability 1 over random samples, S, where the instances of S are sampled according to $\mathcal{D}$ and are labeled by f, we have $L_S(h^*) = 0.$
  - implies that for every ERM hypothesis h_S we have that $L_S(h_S) = 0$
    - h_S has minimum empirical risk, but we already know that h^* has empirical risk equal to 0
- also know as the consistency assumption
- strong assumption
  - relax the assumption later

Learning finite classes

Theorem:
Let $\mathcal{H}$ be a finite hypothesis class. Let $\delta \in (0,1)$ and $\varepsilon > 0$ and let m be an integer that satisfies:

$$m \geq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$$

Then, for any labeling function f, and for any distribution $\mathcal{D}$, for which the realizability assumption holds (that is, for some $h^* \in \mathcal{H}$, $L_{\mathcal{D},f}(h^*) = 0$) with probability of least 1- δ over the choice of an i.i.d, sample S of size m, we have that for every ERM hypothesis h_S it holds that:

$$L_{(\mathcal{D},f)}(h_S) \leq \epsilon.$$

The theorem basically says that for a sufficient large number of training examples m the ERM$_{\mathcal{H}}$ rule over a finite hypothesis class is Probably Approximately Correct

Learning finite classes

The theorem basically says that for a sufficient large number of training examples m the ERM$_{\mathcal{H}}$ rule over a finite hypothesis class is Probably Approximately Correct

$$\underset{S\sim D^m}{P}(L_{f,D}(h_S) \leq \varepsilon) \geq 1-\delta$$

Idea of the proof

- a bad predictor h_b has $L_{D,f}(h_b) > \varepsilon$
- h_b can be output by the $ERM_{\mathcal{H}}$ learning paradigm if has zero empirical error: $L_S(h_b) = 0$
- this can happen if h_b labels correctly all the m training examples from S i.i.d from $\mathcal{D}$
- given a random example from $\mathcal{D}$, h_b has < 1-ε probability to label it correctly
- h_b labels correctly all the m training examples from S with probability $< (1-\varepsilon)^m \leq e^{-\varepsilon m}$
- there are at most $|\mathcal{H}|$ bad hypthotesis, so consider $|H| \times e^{-\varepsilon m} \leq \delta$, so take $m \geq \dfrac{\log(|\mathcal{H}|/\delta)}{\epsilon}$

PAC learnability of a class $\mathcal{H}$

A hypothesis class $\mathcal{H}$ is called ***PAC learnable*** if there exists a function $m_{\mathcal{H}}: (0,1)^2 \rightarrow N$ and a learning algorithm A with the following property:

- for every $\varepsilon > 0$ (*accuracy* → *"approximately correct"*)
- for every $\delta > 0$ (*confidence* → *"probably"*)
- for every labeling $f \in \mathcal{H}$ (*realizability case*)
- for every distribution $\mathcal{D}$ over $\mathcal{X}$

when we run the learning algorithm A on a training set S, consisting of $m \geq m_{\mathcal{H}}(\varepsilon, \delta)$ examples sampled i.i.d. from $\mathcal{D}$ and labeled by f the algorithm A returns a hypothesis $h_S \in \mathcal{H}$ such that, with probability at least $1-\delta$ (over the choice of examples), $L_{D,f}(h_S) \leq \varepsilon$.

L. G. Valiant, *A theory of the Learnable*, Communications ACM, 27(11):1134-1142, 1984

PAC learnability of a class $\mathcal{H}$

A hypothesis class $\mathcal{H}$ is called ***PAC learnable*** if there exists a function $m_{\mathcal{H}}: (0,1)^2 \rightarrow N$ and a learning algorithm A with the following property:

- for every $\varepsilon > 0$ (*accuracy* → "*approximately correct*")
- for every $\delta > 0$ (*confidence* → "*probably*")
- for every labeling $f \in \mathcal{H}$ (*realizability case*)
- for every distribution $\mathcal{D}$ over $\mathcal{X}$

when we run the learning algorithm A on a training set S, consisting of $m \geq m_{\mathcal{H}}(\varepsilon, \delta)$ examples sampled i.i.d. from $\mathcal{D}$ and labeled by f the algorithm A returns a hypothesis $h_S \in \mathcal{H}$ such that, with probability at least $1-\delta$ (over the choice of examples), $L_{\mathrm{D},f}(h_S) \leq \varepsilon$.

$$\underset{S \sim D^m}{P}(L_{f,D}(h_S) \leq \varepsilon) \geq 1-\delta$$

- $h_S = A(S)$
- the function $m_{\mathcal{H}}: (0,1)^2 \rightarrow N$ is called sample complexity of learning $\mathcal{H}$
- $m_{\mathcal{H}}(\varepsilon, \delta)$ – the minimum number of examples required to guarantee a PAC solution

L. G. Valiant, *A theory of the Learnable*, Communications ACM, 27(11):1134-1142, 1984

PAC learnability of a class $\mathcal{H}$

A hypothesis class $\mathcal{H}$ is called ***PAC learnable*** if there exists a function $m_{\mathcal{H}}: (0,1)^2 \rightarrow N$ and a learning algorithm A with the following property:

- for every $\varepsilon > 0$ (*accuracy* → *"approximately correct"*)
- for every $\delta > 0$ (*confidence* → *"probably"*)
- for every labeling $f \in \mathcal{H}$ (*realizability case*)
- for every distribution $\mathcal{D}$ over $\mathcal{X}$

when we run the learning algorithm A on a training set S, consisting of $m \geq m_{\mathcal{H}}(\varepsilon, \delta)$ examples sampled i.i.d. from $\mathcal{D}$ and labeled by f the algorithm A returns a hypothesis $h_S \in \mathcal{H}$ such that, with probability at least $1-\delta$ (over the choice of examples), $L_{\mathrm{D},f}(h_S) \leq \varepsilon$.

$$\underset{S \sim D^m}{P}(L_{f,D}(h_S) \leq \varepsilon) \geq 1-\delta \Leftrightarrow \underset{S \sim D^m}{P}(L_{f,D}(h_S) > \varepsilon) < \delta$$

L. G. Valiant, *A theory of the Learnable*, Communications ACM, 27(11):1134-1142, 1984

Sample complexity

- the function $m_{\mathcal{H}}: (0,1)^2 \rightarrow N$ is called sampled complexity of learning $\mathcal{H}$
- $m_{\mathcal{H}}(\varepsilon, \delta)$ – the minimum number of examples required to guarantee a PAC solution
- depends on:
  - accuracy ε
  - confidence δ
  - properties of $\mathcal{H}$
- different than *time complexity* (discuss it in the following lectures)

Every finite hypothesis class H is PAC learnable with sample complexity

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{\log(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil$$

Concept class

- h: $\mathcal{X} \rightarrow \{0,1\}$ the *target concept* to learn
  - can be identified with its support $\{x \in \mathcal{X} \mid h(x) = 1\}$
  - set of points inside a rectangle
    - h = indicator function of these points
    - the concept to learn is a rectangle

- $\mathcal{H}$ can be interpreted as the concept class, a set of target concepts h
  - set of all rectangles in the plane
  - conjunction of Boolean literals

Conjunctions of Boolean literals

- C_n = concept class of conjunctions of at most n Boolean literals $x_1, \ldots, x_n$
  - a Boolean literal is either x_i or its negation $\overline{x_i}$
  - can interpret x_i as feature i
  - example: $h = x_1 \wedge \overline{x_2} \wedge x_4$ where $\overline{x_2}$ denotes the negation of the Boolean literal x_2

Conjunctions of Boolean literals

- C_n = concept class of conjunctions of at most n Boolean literals $x_1, \ldots, x_n$
  - a Boolean literal is either x_i or its negation $\overline{x_i}$
  - can interpret x_i as feature i
  - example: $h = x_1 \wedge \overline{x_2} \wedge x_4$ where $\overline{x_2}$ denotes the negation of the Boolean literal x_2

- observe that for n = 4:
  - a positive example such as (1, 0, 0, 1) implies that the target concept cannot contain the literals $\overline{x_1}$, x_2, x_3 and $\overline{x_4}$
    - for example if x_2 was present in the conjunction then for the current positive example (where x_2 has value 0) the label should have been 0
  - cannot say anything about literals x_1, $\overline{x_2}$, $\overline{x_3}$ and x_4. They might be present or absent in the conjunction (target concept) that we are searching for
  - the first positive example eliminates half of the literals

Conjunctions of Boolean literals

- C_n = concept class of conjunctions of at most n Boolean literals $x_1, \ldots, x_n$
  - a Boolean literal is either x_i or its negation $\overline{x_i}$
  - can interpret x_i as feature i
  - example: $h = x_1 \wedge \overline{x_2} \wedge x_4$ where $\overline{x_2}$ denotes the negation of the Boolean literal x_2

- observe that for n = 4:
  - a positive example such as (1, 0, 0, 1) implies that the target concept cannot contain the literals $\overline{x_1}$, x_2, x_3 and $\overline{x_4}$
    - for example if x_2 was present in the conjunction then for the current positive example (where x_2 has value 0) the label should have been 0
  - cannot say anything about literals x_1, $\overline{x_2}$, $\overline{x_3}$ and x_4. They might be present or absent in the conjunction (target concept) that we are searching for
  - the first positive example eliminates half of the literals

  - in contrast, a negative example such as (1, 0, 0, 0) is not as informative since it is not known which of its n bits are incorrect.

Conjunctions of Boolean literals

- C_n = concept class of conjunctions of at most n Boolean literals $x_1, \ldots, x_n$
- a simple algorithm for finding a consistent hypothesis is thus based on positive examples and consists of the following:
  - for each positive example $(b_1, \ldots b_n)$,
    - if $b_i = 1$ then $\overline{x_i}$ is ruled out as a possible literal in the concept class
    - if $b_i = 0$ then x_i is ruled out.
  - the conjunction of all the literals not ruled out is thus a hypothesis consistent with the target

0	1	1	0	1	1	+
0	1	1	1	1	1	+
0	0	1	1	0	1	-
0	1	1	1	1	1	+
1	0	0	1	1	0	-
0	1	0	0	1	1	+

$x_1 \quad \overline{x_1} \quad x_2 \quad \overline{x_2} \quad x_3 \quad \overline{x_3} \quad x_4 \quad \overline{x_4} \quad x_5 \quad \overline{x_5} \quad x_6 \quad \overline{x_6}$

$$\rightarrow \overline{x}_1 \wedge x_2 \wedge x_5 \wedge x_6$$

Conjunctions of Boolean literals

- C_n = concept class of conjunctions of at most n Boolean literals $x_1, \ldots, x_n$
- $|C_n| = 3^n$ – finite, so is PAC learnable with sample complexity $m_{\mathcal{H}}(\varepsilon, \delta) \leq m$:

$$m \geq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$$

$$m \geq \left\lceil \frac{1}{\varepsilon}\left(n \log(3) + \log(\frac{1}{\delta}) \right) \right\rceil$$

$$m \geq \left\lceil \frac{1}{\varepsilon}\left(n \log(3) - \log(\delta) \right) \right\rceil$$

- for $\varepsilon = 0.01$, $\delta = 0.02$, n = 10, m ≥ 149, no matter how $\mathcal{D}$ looks like, all possible examples are $2^{10} = 1024$
- we need at least 149 examples; the bound guarantees (at least) 99% accuracy with (at least) 98% confidence

Universal concept class $\mathcal{U}_n$

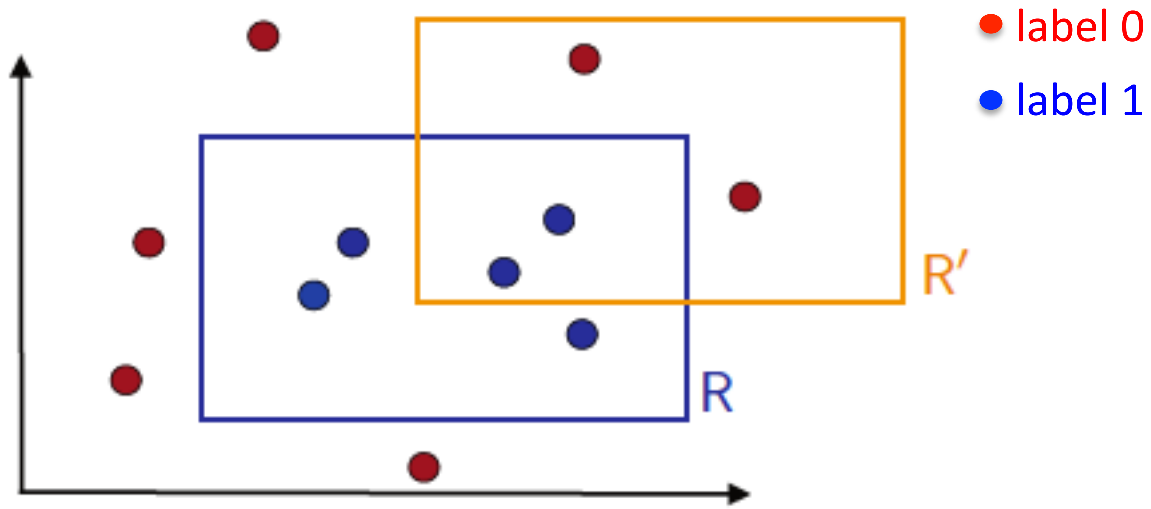
- B^n = set of boolean n-tuples, $|B| = 2^n$
- want to learn arbitrary subsets of B^n
- $\mathcal{U}_n = \{h: B^n \to \{0,1\}\}$ - the concept class formed by all subsets of B^n
- $\mathcal{U}_n$ – universal class
- is this concept class PAC-learnable?
- $|\mathcal{U}_n| = 2^{2^n}$ – finite, so is PAC learnable with $m_{\mathcal{H}}(\varepsilon, \delta)$ in the order of m:

$$m \geq \left\lceil \frac{1}{\varepsilon}\left(2^n \log(2) + \log(\frac{1}{\delta})\right)\right\rceil$$

- sample complexity exponential in n, number of variables
- $\mathcal{U}_n$ is finite and hence PAC-learnable, but we will need exponential time (to inspect exponentially many examples)
- for $\varepsilon = 0.01$, $\delta = 0.02$, n = 10, $m \geq 71370$, no matter how $\mathcal{D}$ looks like, all possible examples are $2^{10} = 1024$
- it is not PAC-learnable in any practical sense (need polynomial time complexity = later require $m_{\mathcal{H}}$ be polynomial in $1/\varepsilon, 1/\delta, n, |\mathcal{H}|$)

Axis-aligned rectangles

- $\mathcal{X} = \mathrm{R}^2$ points in the plane
- $\mathcal{H}$ = set of all axis-aligned rectangle lying in R^2
- each concept $h \in \mathcal{H}$ is an indicator function of a rectangle
- the learning problem consists of determining with small error a target axis-aligned rectangle using the labeled training sample



Target concept R and possible hypothesis R′. Circles represent training instances. A blue circle is a point labeled with 1, since it falls within the rectangle R. Others are red and labeled with 0.

Axis-aligned rectangles

- $\mathcal{X} = \mathrm{R}^2$ points in the plane
- $\mathcal{H}$ = set of all axis-aligned rectangle lying in R^2
- $|\mathcal{H}| = \infty$
- still $\mathcal{H}$ is PAC-learnable with sample complexity in the order of:

$$m \geq \left\lceil \frac{4}{\varepsilon} \log(\frac{1}{\delta}) \right\rceil$$

- simple algorithm: take the tightest rectangle enclosing all the positive examples (or take the largest rectangle not including negative samples)

- discuss this example in seminar