Aalto University
School of Science

CS-E4070 — Computational learning theory

Slide set 01 : introduction to PAC learning

Cigdem Aslay and Aris Gionis

Aalto University

spring 2019

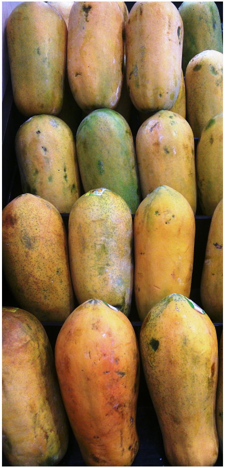
reading material

- SS&BD, chapters 2 and 3
- K&V, chapter 1

stranded in a tropical island

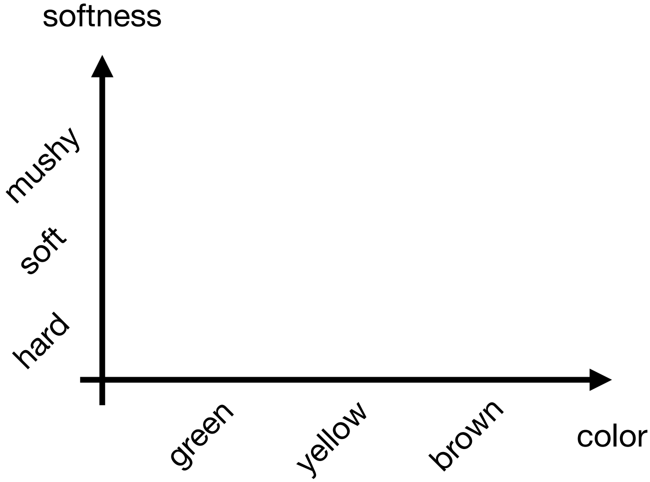
need to buy papayas from the local market

- want to learn to recognize tasty fruits
- judge based on color and softness
- start learning after tasting few samples

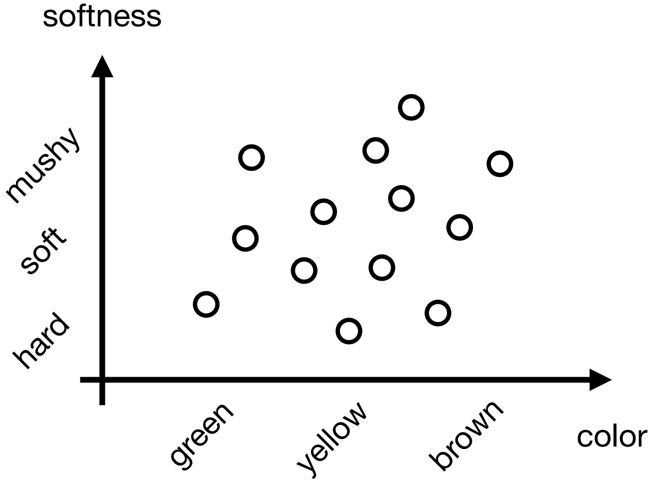


example from SS&BD

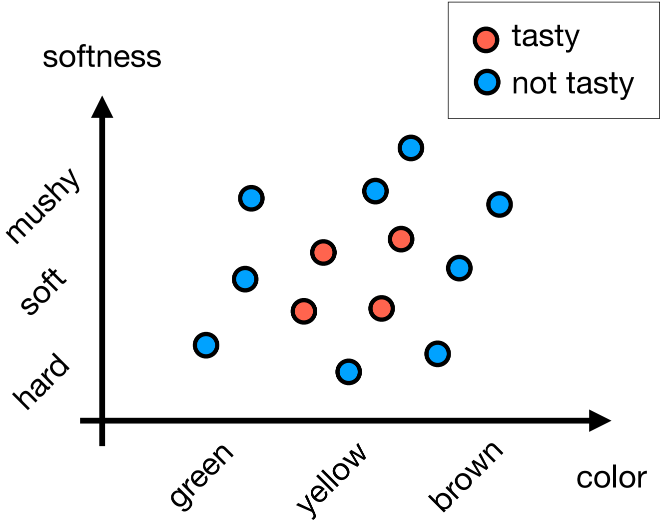
papayas tasting data



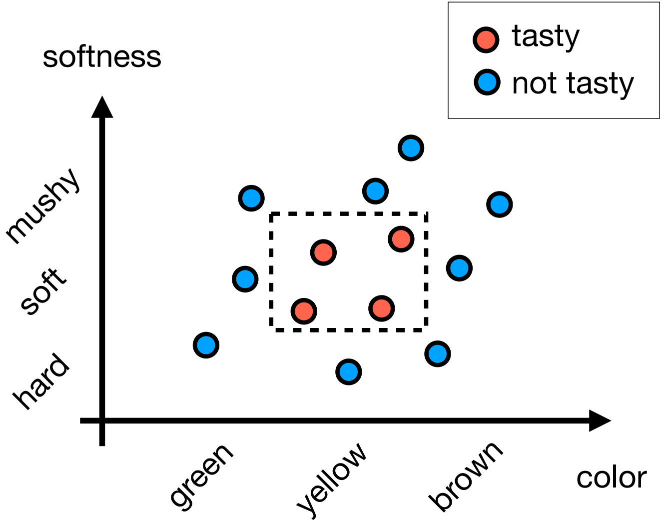
papayas tasting data



papayas tasting data



papayas tasting data



formalization

- X : instance space, or input space

  the space in which we represent our input data

- Y : label space, e.g., $Y = \{0, 1\}$ or $Y = \{-1, 1\}$

  the set of available labels

- $c : X \to Y$: target concept

  the mapping we want to learn

- $\mathcal{C}$: concept class, i.e., $c \in \mathcal{C}$

  a collection of concepts over X

formalization

- $\mathcal{D}$: a probability distribution over X
- $EX(\mathcal{D}, c)$: example (sample) generator

  returns an example (sample) $(\mathbf{x}, y)$, where $\mathbf{x}$ is sampled from $\mathcal{D}$, and $y = c(\mathbf{x})$
- $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$: sample set, or training set

  each $(\mathbf{x}, y) \in S$ is generated by $EX(\mathcal{D}, c)$

the learner

- the learner observes sample set S and outputs

  $h : X \to Y$: hypothesis, or predictor

  also denoted h_S to emphasize dependence on S

- hypothesis h can be used to predict the label of future data points $\mathbf{x}$

- particularly interested in quantifying the performance of the learner for predicting data drawn from $\mathcal{D}$

measures of success

- the error of the learner is defined as the probability that the learner does not predict the correct label on a random data point sampled from $\mathcal{D}$

$$error_{\mathcal{D}}(h) = \mathbf{Pr}_{\mathbf{x} \sim \mathcal{D}}[h(\mathbf{x}) \neq c(\mathbf{x})]$$

other considerations

- the size m of the sample set S
- the running time of the learner
- the class required to represent the hypothesis h

empirical risk

- define empirical risk the error on the training set

$$error_S(h) = \frac{1}{m} |\{i \in [m] \mid h(\mathbf{x}_i) \neq y_i\}| = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}[h(\mathbf{x}_i) \neq y_i]$$

  where $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ a sample set of size m, $[n] = \{1, \ldots, n\}$, and $\mathbb{I}$ the indicator function

we want to minimize empirical risk

- what may go wrong ?

overfitting

- the hypothesis

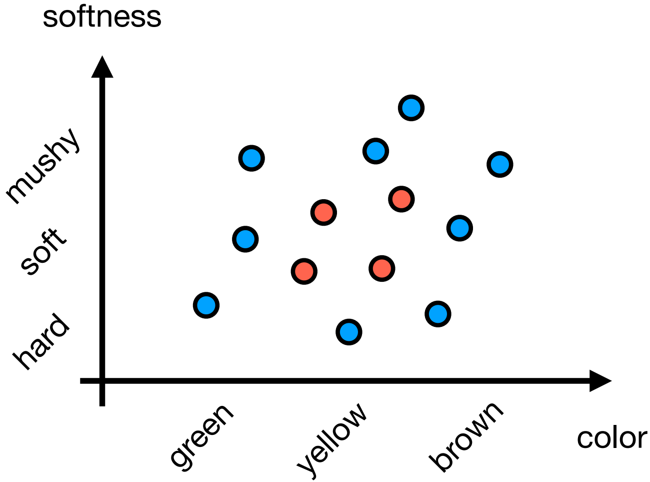
$$h(\mathbf{x}) = \begin{cases} y_i & \text{if } \mathbf{x} = \mathbf{x}_i \text{ for some } i \\ 0 & \text{otherwise} \end{cases}$$

achieves $error_S(h) = 0$ but has no generalization power

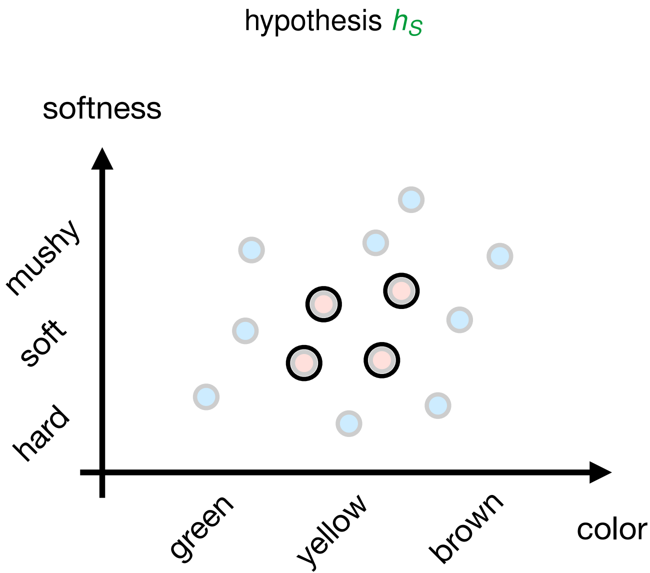
- such hypothesis may seem artificial, but could be achieved by a “natural” polynomial of sufficiently high degree

overfitting

sample $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$



overfitting



how to deal with overfitting

- do not consider arbritrarily complex hypotheses
- restrict search over a “natural” family of hypotheses
- $\mathcal{H}$: hypothesis class
  - e.g., $\mathcal{H} =$ set of axis-aligned rectangles
- such rectification is known as inductive bias
- bias is decided in advance; prior knowledge is needed
- empirical risk minimization rule becomes

$$EX_{\mathcal{H}}(S) = \arg\min_{h \in \mathcal{H}} error_S(h)$$

the case of finite hypothesis class $\mathcal{H}$

- let's assume that $\mathcal{H}$ is finite
  - not an unreasonable assumption;
    we can always discretize
- the empirical risk minimization rule does not overfit

what do we want show ?

- the empirical risk minimization rule gives hypothesis

$$h_S = EX_{\mathcal{H}}(S) = \arg\min_{h \in \mathcal{H}} error_S(h)$$

- we want to show that $error_{\mathcal{D}}(h_S)$ is small
- recall that S has been drawn from $\mathcal{D}$
- we assume independent samples, denoted by $S \sim \mathcal{D}^m$
- realizability assumption : there exists a hypothesis $h^* \in \mathcal{H}$ such that $error_{\mathcal{D}}(h^*) = 0$
- the realizability assumption implies that $error_S(h^*) = 0$, and thus, also $error_S(h_S) = 0$

what can we hope to show ?

- we want to show that $error_{\mathcal{D}}(h_S)$ is small
- we want to show that $error_{\mathcal{D}}(h_S) \leq \epsilon$

  where $\epsilon > 0$ is an accuracy parameter
- in addition, we may get “unlucky” and draw a “bad” sample
- thus, we want $error_{\mathcal{D}}(h_S) \leq \epsilon$ with high probability
- we introduce a confidence parameter $\delta \in (0, 1)$
- we require $error_{\mathcal{D}}(h_S) \leq \epsilon$ with probability at least $1 - \delta$

what else do we want show ?

- we also want to show that our learning scheme is efficient
- not “too many” samples are sufficient

finite hypothesis class and realizability

- assuming a finite hypothesis class and realizability the empirical risk minimization rule does not overfit

- **theorem** (FINITE) : consider a finite hypothesis class $\mathcal{H}$ and assume realizability. Consider accuracy $\epsilon > 0$, confidence $\delta \in (0, 1)$, and sample size

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

  let h_S the hypothesis selected by the empirical risk minimization rule over a sample $S \sim \mathcal{D}^m$. Then

  $$\mathit{error}_{\mathcal{D}}(h_S) \leq \epsilon$$

  with probability at least $1 - \delta$.

proof of FINITE theorem (sketch)

- **lemma**: the probability that any hypothesis with error more than ϵ is consistent with a sample S of size m is less than $(1-\epsilon)^m|\mathcal{H}|$

- thus, the probability that all consistent hypotheses have error at most ϵ is at least $1-(1-\epsilon)^m|\mathcal{H}|$

- we want to select m so that

$$(1-\epsilon)^m|\mathcal{H}| \leq \delta$$

  which gives

$$m \geq \frac{1}{-\ln(1-\epsilon)}\left(\ln|\mathcal{H}| + \ln\left(\frac{1}{\delta}\right)\right) \geq \frac{1}{\epsilon}\left(\ln|\mathcal{H}| + \ln\left(\frac{1}{\delta}\right)\right)$$

PAC learning

- previous statement has the form

$$\underbrace{\text{the error is at most } \epsilon}_{\text{approximate}} \underbrace{\text{with probability at least } 1 - \delta}_{\text{probable}}$$

- probably approximate correct (PAC) learning
  - note that ϵ and δ can be arbitrarily close to 0

definition of PAC learning

- (preliminary) **definition** (PAC learning) :
  a concept class $\mathcal{C}$ is PAC learnable if there is a learning algorithm A with the following property:
  for every concept $c \in \mathcal{C}$, every distribution $\mathcal{D}$, and every $\epsilon > 0$ and $\delta \in (0, 1)$, there is a number m so that if A is given a sample $S \sim \mathcal{D}^m$, it outputs a hypothesis $h \in \mathcal{C}$ that satisfies

  $$error_{\mathcal{D}}(h) \leq \epsilon$$

  with probability at least $1 - \delta$.

notes on PAC learning definition

- the sample data are drawn from $\mathcal{D}$ and labeled according to a taget concept $c \in \mathcal{C}$
- realizability assumption holds because we require $h \in \mathcal{C}$

- the definition can be modified so that we can consider learning a target concept $c \in \mathcal{C}$ using a hypothesis h from a different class $\mathcal{H}$
- this is useful when we are agnostic about concept class $\mathcal{C}$

efficient PAC learning

- if the learning algorithm runs in time polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ we say that the $\mathcal{C}$ is efficiently PAC learnable
- this implies that m is polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$

applications

- **theorem** (FINITE) can be rephrased as

  every finite hypothesis class is PAC learnable with sample complexity

$$m_{\mathcal{H}} \leq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$$

application : no-free-lunch theorem

SS&BD, chapter 5

- we can show that there is no universal learner
  - some form of prior knowledge is necessary
  - we should know something about $\mathcal{D}$ and/or $\mathcal{C}$

- **theorem** (no-free-lunch) : let A be a learner over X.
  Then there exists a distribution $\mathcal{D}$ over $X \times \{0, 1\}$ such that
  1. there exists concept $c : X \to \{0, 1\}$ with $error_{\mathcal{D}}(c) = 0$
  2. with probability at least $1/7$ over $S \sim \mathcal{D}^m$ we have that $error_{\mathcal{D}}(A(S)) \geq 1/8$

- **corollary** : let $\mathcal{C}$ be the set of all mappings from an infinite domain X to $\{0, 1\}$. Then, $\mathcal{C}$ is not PAC learnable.

representation size

- efficient PAC learning $=$ polynomial learning algorithm
- we have ignored representation issues
- however, the representation of the target concept matters
  - different representations of the same concept may differ exponentially

examples

- boolean functions represented in DNF or not
- convex polytope represented by its vertices or by linear constraints of its faces

representation size

- for running-time considerations the hypothesis representation size is important
- hypothesis representation size is a lower bound on time complexity
- notice that we have no information about the representation of the target concept
  - we only observe labeled data

representation scheme

- a representation scheme specifies how to represent a concept class with strings of a finite vocabulary
  - e.g., a decision tree can be represented by a C program that implements the tree
- $size(h)$ is the encoding in bits of a concept h
- for a target concept c (that we do not know how it is actually represented) we define

$$size(c) = \min_{\mathcal{R}(z)=c} \{size(z)\}$$

i.e., the minimum possible encoding

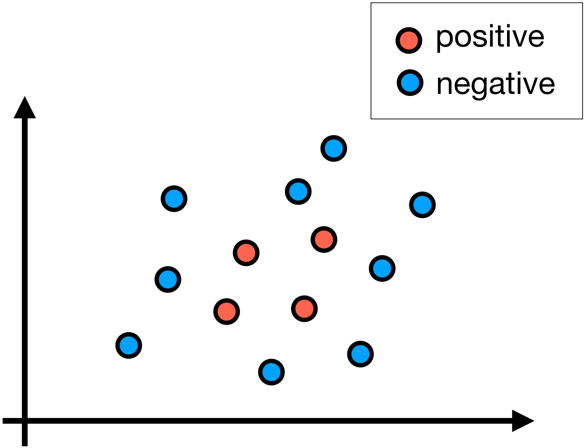
instance dimension

- we often parameterize an instance space and an associated concept class by a notion of dimension
- for example
  - $X_n = \{0, 1\}^n$: the set of n boolean variables
  - $\mathcal{C}_n$: boolean formulas in 3-CNF over n variables
  - $X = \bigcup_{n \geq 1} X_n$
  - $\mathcal{C} = \bigcup_{n \geq 1} \mathcal{C}_n$

modified definition of PAC learning

- (modified) **definition** (PAC learning) :
  a concept class $\mathcal{C}_n$ over an instance space X_n is PAC learnable if there is a learning algorithm that satisfies the properties of the previous (preliminary) definition, and in addition the algorithm runs in polynomial time with respect to n, $size(c)$, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$, when learning a target concept $c \in \mathcal{C}_n$.

learning axis-aligned rectangles



K&V, section 1.1

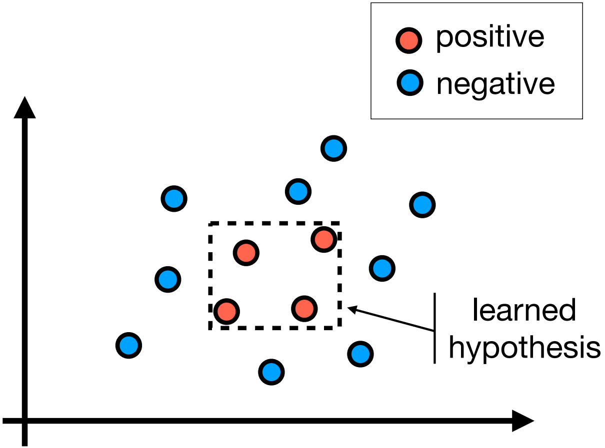
learning axis-aligned rectangles

learning algorithm

1. observe sample $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ drawn from distribution $\mathcal{D}^m$
2. return the tightest-fit axis-aligned rectangle that contains all positive examples

   (by realizability assumption the returned rectangle does not contain any negative example)

learning axis-aligned rectangles



K&V, section 1.1

learning axis-aligned rectangles

K&V, section 1.1

- **theorem**

  the class of axis-aligned rectangles is efficiently PAC learnable with sample complexity

$$m_{\mathcal{R}} \leq \frac{4}{\epsilon} \ln \left(\frac{4}{\delta} \right)$$

learning boolean conjunctions

K&V, section 1.3

- consider n boolean variables $x_1, \ldots, x_n$
- instance space $X_n = \{0, 1\}^n$ is the set of all truth assignments of the boolean variables $x_1, \ldots, x_n$
- we use a_i to denote the value of x_i in a truth assignment
- concept class $\mathcal{C}_n$ is the set of all boolean conjunctions over X_n, e.g.,

$$c(x_1, x_2, x_3, x_4) = x_1 \wedge \overline{x}_2 \wedge x_4$$

- $size(c) \leq 2n$, and encoding requires $\mathcal{O}(n \log n)$ bits
- examples $(\mathbf{a}, y)$ drawn from $EX(\mathcal{D}, c)$ consist of truth assignments $\mathbf{a}$ and their evaluation $y = c(\mathbf{a}) \in \{0, 1\}$

learning boolean conjunctions

K&V, section 1.3

learning algorithm

- initial hypothesis

$$h(x_1, \ldots, x_n) = x_1 \wedge \overline{x}_1 \wedge x_2 \wedge \overline{x}_2 \wedge \ldots \wedge x_n \wedge \overline{x}_n$$

  (initially not satisfiable)

- negative examples drawn from $EX(\mathcal{D}, c)$ are ignored
- for positive examples
  - if $a_i = 0$ we delete literal x_i from h
  - if $a_i = 1$ we delete literal $\overline{x}_i$ from h

learning boolean conjunctions

K&V, section 1.3

analysis of the learning algorithm

- a literal is deleted from h if it is 0 in a positive example
- clearly, such a literal cannot be in the concept target c
- the literals of h include those of c
  i.e., h is a more specific than c
- h will never err in a negative example
- h will only err in a positive example due to some literal
  that was not deleted in the training
- high-level idea : if such a literal is not likely to appear in the training set, then it is also not likely to appear in the test set

proof sketch

K&V, section 1.3

- consider literal z that is in h but not in c
- z causes h to err in positive examples in which $z = 0$
- define $p(z) = \mathbf{Pr}_{\mathbf{a} \in \mathcal{D}} [c(\mathbf{a}) = 1 \wedge z \text{ is } 0 \text{ in } \mathbf{a}]$
- every error of h can be "blamed" to at least one literal z of h
- by union bound: $error(h) \leq \sum_{z \in h} p(z)$
- we call literal z "bad" if $p(z) \geq \epsilon/(2n)$
- if h contains no bad literals then $error(h) \leq (2n)\epsilon/(2n) = \epsilon$
- the probability that a bad literal is not removed from h (after seeing m examples) is at most $(1 - \epsilon/2n)^m$
- the probability that some bad literal is not removed is at most $2n(1 - \epsilon/2n)^m$
- again, select m so that $2n(1 - \epsilon/2n)^m \leq \delta$

learning boolean conjunctions

K&V, section 1.3

- **theorem**

  the class of conjunctions of boolean literals is efficiently PAC learnable with sample complexity

$$m_{\mathcal{C}} \leq \frac{2n}{\epsilon}\left(\ln(2n) + \ln\left(\frac{1}{\delta}\right)\right)$$

intractability 3-term DNF formulas

K&V, section 1.4

- concept class $\mathcal{C}_n$ of 3-term DNF formulas is the set of all disjunctions

$$T_1 \vee T_2 \vee T_3$$

  where T_1, T_2, and T_3 are conjunctions of literals over boolean variables $x_1, \ldots, x_n$

- **theorem**

  the class of 3-term DNF formulas is not efficiently PAC learnable, unless $\mathbf{RP} = \mathbf{NP}$

  - reduction from graph 3-coloring problem (!)

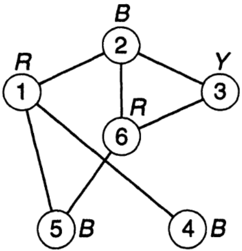
intractability proof sketch

- we want to show that $\mathcal{C}$ is not PAC learnable
- obtain reduction from an **NP**-hard language A
- given a we want to answer whether $a \in A$
- we want to : map a to a sample set S_a so that

  $a \in A$ if and only if $\exists$ concept $c \in \mathcal{C}$ consistent with S_a

- we can use a PAC learning algorithm L to decide $a \in A$
- trick : set $\epsilon = 1/(2|S_a|)$ and $\mathcal{D}$ uniform over S_a
- any h found by L would be consistent with S_a
  because even for one mistake, error would be $1/|S_a| > \epsilon$

reduction from graph 3-coloring problem



Graph G

S_G^+	S_G^-
$\langle 011111, 1 \rangle$	$\langle 001111, 0 \rangle$
$\langle 101111, 1 \rangle$	$\langle 011011, 0 \rangle$
$\langle 110111, 1 \rangle$	$\langle 011101, 0 \rangle$
$\langle 111011, 1 \rangle$	$\langle 100111, 0 \rangle$
$\langle 111101, 1 \rangle$	$\langle 101110, 0 \rangle$
$\langle 111110, 1 \rangle$	$\langle 110110, 0 \rangle$
	$\langle 111100, 0 \rangle$

$$T_R = x_2 \wedge x_3 \wedge x_4 \wedge x_5$$

$$T_B = x_1 \wedge x_3 \wedge x_6$$

$$T_Y = x_1 \wedge x_2 \wedge x_4 \wedge x_5 \wedge x_6$$

Figure 1.5: *A graph G with a legal 3-coloring, the associated sample, and the terms defined by the coloring.*

avoiding intractability by using 3-CNF formulas

- the class of 3-CNF formulas is the set of conjunctions of clauses, where each clause is a disjunction of at most 3 literals over boolean variables $x_1, \ldots, x_n$

- 3-CNF formulas are more expressive than 3-term DNF formulas, as

$$T_1 \vee T_2 \vee T_3 = \bigwedge_{u \in T_1, v \in T_2, w \in T_3} (u \vee v \vee w)$$

- **theorem** K&V, section 1.5

  the class of 3-term DNF formulas is efficiently PAC learnable using 3-CNF formulas

remark

- 3-CNF formulas are more expressive than 3-term DNF
- 3-term DNF formulas are not efficiently PAC learnable in their own representation class, but they are efficiently PAC learnable using 3-CNF formulas
- the choice of hypothesis representation is very important

final definition of PAC learning

- (final) **definition** (PAC learning) :
  let $\mathcal{C}$ be a concept class over an instance space X and $\mathcal{H}$ be a representation class over X. We say that $\mathcal{C}$ is efficiently PAC learnable using $\mathcal{H}$ if the previous (modified) definition of PAC learning is satisfied by a learning algorithm that is allowed to output a hypothesis from $\mathcal{H}$.

  $\mathcal{H}$ needs to be at least as expressive as $\mathcal{C}$

  We refer to $\mathcal{H}$ as the hypothesis class of the PAC learning algorithm.

summary of previous results

K&V, section 1.5

- the representation class of 1-term DNF formulas (conjunctions) is efficiently PAC learnable using 1-term DNF formulas

- for $k \geq 2$, the representation class of k-term DNF formulas is not efficiently PAC learnable using k-term DNF formulas, but it is efficiently PAC learnable using k-CNF formulas

reading assignment

study in detail the proofs of the theorems we discussed

- SS&BD, chapters 2 and 3
- K&V, chapter 1