Machine Learning
Is Learning Feasible?

Prof.Shuyuan Yang, Zhixi Feng

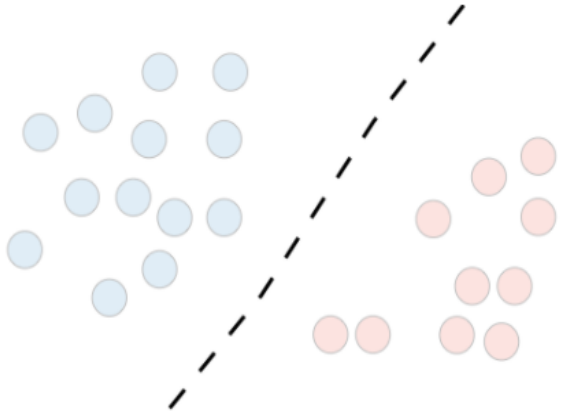
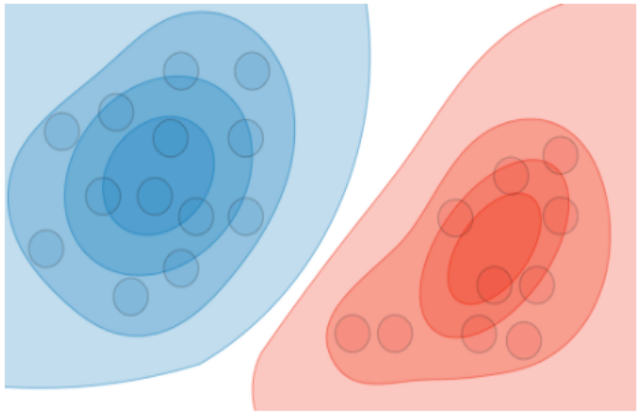
E-mail： syyang@xidianedu.cn

zxfeng@xidian.edu.cn

Contents

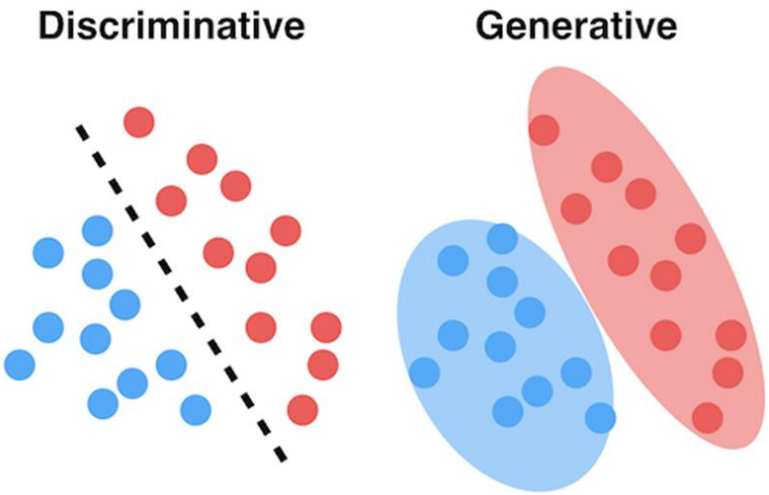
- Supervised-learning -- Generative VS. Generative Learning
- What constitutes a learning problem?
- Is learning feasible?
- Is learning feasible?

Type of model

	Discriminative model	Generative model
Goal	Directly estimate $P(y\|x)$	Estimate $P(x\|y)$ to then deduce $P(y\|x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes

Generative VS. Generative Learning

- Generative:
  - probabilistic “model” of each class
  - decision boundary:
    - where one model becomes more likely
  - natural use of unlabeled data



- Discriminative:
  - focus on the decision boundary
  - more powerful with lots of examples
  - not designed to use unlabeled data
  - only supervised tasks

Gaussian Discriminant Analysis

❒ **Setting** — The Gaussian Discriminant Analysis assumes that y and $x|y=0$ and $x|y=1$ are such that:

(1) $\boxed{y \sim \text{Bernoulli}(\phi)}$ (2) $\boxed{x|y=0 \sim \mathcal{N}(\mu_0, \Sigma)}$ (3) $\boxed{x|y=1 \sim \mathcal{N}(\mu_1, \Sigma)}$

❒ **Estimation** — The following table sums up the estimates that we find when maximizing the likelihood:

$\widehat{\phi}$	$\widehat{\mu_j} \quad (j=0,1)$	$\widehat{\Sigma}$
$\frac{1}{m}\sum_{i=1}^{m} 1_{\{y^{(i)}=1\}}$	$\frac{\sum_{i=1}^{m} 1_{\{y^{(i)}=j\}} x^{(i)}}{\sum_{i=1}^{m} 1_{\{y^{(i)}=j\}}}$	$\frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu_{y^{(i)}})(x^{(i)}-\mu_{y^{(i)}})^T$

Naive Bayes

❒ **Assumption** — The Naive Bayes model supposes that the features of each data point are all independent:

$$\boxed{P(x|y) = P(x_1, x_2, ...|y) = P(x_1|y)P(x_2|y)... = \prod_{i=1}^{n} P(x_i|y)}$$

❒ **Solutions** — Maximizing the log-likelihood gives the following solutions:

$$\boxed{P(y=k) = \frac{1}{m} \times \#\{j|y^{(j)} = k\}} \quad \text{and} \quad \boxed{P(x_i = l|y=k) = \frac{\#\{j|y^{(j)} = k \text{ and } x_i^{(j)} = l\}}{\#\{j|y^{(j)} = k\}}}$$

with $k \in \{0, 1\}$ and $l \in [\![1, L]\!]$

Remark: Naive Bayes is widely used for text classification and spam detection.

Supervised Learning cheatsheet

Given a set of data points $\{x^{(1)}, ..., x^{(m)}\}$ associated to a set of outcomes $\{y^{(1)}, ..., y^{(m)}\}$, we want to build a classifier that learns how to predict y from x.

❒ **Type of prediction** — The different types of predictive models are summed up in the table below:

	Regression	Classification
Outcome	Continuous	Class
Examples	Linear regression	Logistic regression, SVM, Naive Bayes

Supervised Learning

Linear regression

We assume here that $y|x;\theta \sim \mathcal{N}(\mu, \sigma^2)$

❒ **Normal equations** — By noting X the design matrix, the value of θ that minimizes the cost function is a closed-form solution such that:

$$\boxed{\theta = (X^T X)^{-1} X^T y}$$

❒ **LMS algorithm** — By noting α the learning rate, the update rule of the Least Mean Squares (LMS) algorithm for a training set of m data points, which is also known as the Widrow-Hoff learning rule, is as follows:

$$\boxed{\forall j, \quad \theta_j \leftarrow \theta_j + \alpha \sum_{i=1}^{m} \left[y^{(i)} - h_\theta(x^{(i)}) \right] x_j^{(i)}}$$

Remark: the update rule is a particular case of the gradient ascent.

Remark: Stochastic gradient descent (SGD) is updating the parameter based on each training example, and batch gradient descent is on a batch of training examples.

Supervised Learning

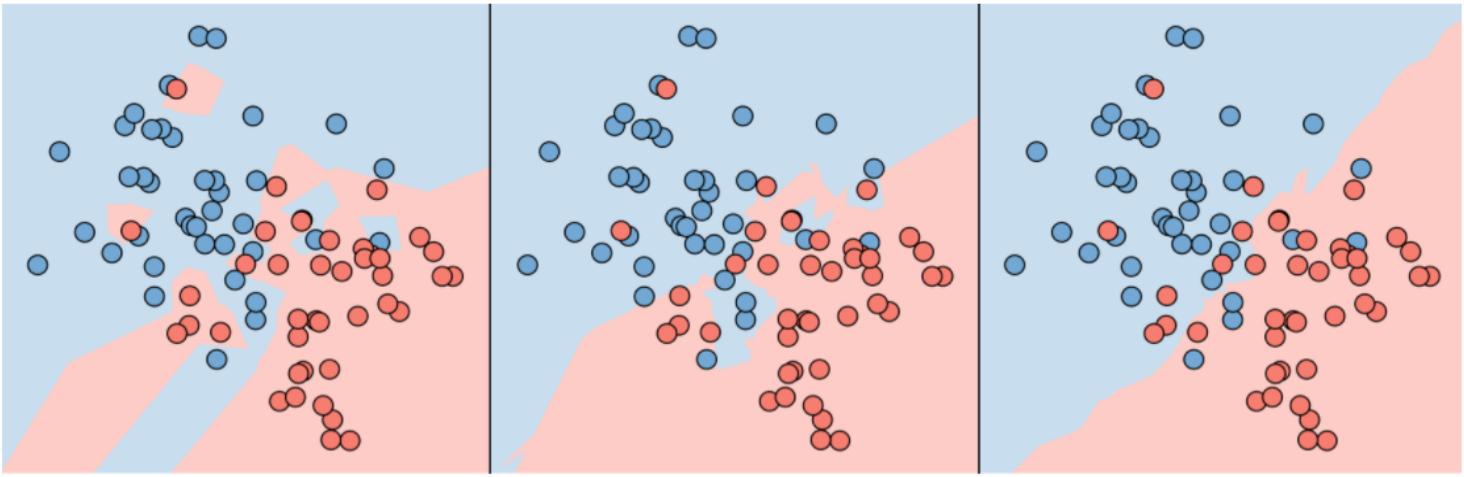
❒ **LWR** — Locally Weighted Regression, also known as LWR, is a variant of linear regression that weights each training example in its cost function by $w^{(i)}(x)$, which is defined with parameter $\tau \in \mathbb{R}$ as:

$$\boxed{w^{(i)}(x) = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)}$$

Non-parametric approaches

❒ **k-nearest neighbors** — The k-nearest neighbors algorithm, commonly known as k-NN, is a non-parametric approach where the response of a data point is determined by the nature of its k neighbors from the training set. It can be used in both classification and regression settings.

Remark: the higher the parameter k, the higher the bias, and the lower the parameter k, the higher the variance.



$k = 1$ $k = 3$ $k = 11$

Classification and logistic regression

❒ **Sigmoid function** — The sigmoid function g, also known as the logistic function, is defined as follows:

$$\forall z \in \mathbb{R}, \quad \boxed{g(z) = \frac{1}{1+e^{-z}} \in]0,1[}$$

❒ **Logistic regression** — We assume here that $y|x;\theta \sim \text{Bernoulli}(\phi)$. We have the following form:

$$\boxed{\phi = p(y=1|x;\theta) = \frac{1}{1+\exp(-\theta^T x)} = g(\theta^T x)}$$

Remark: logistic regressions do not have closed form solutions.

Classification and logistic regression

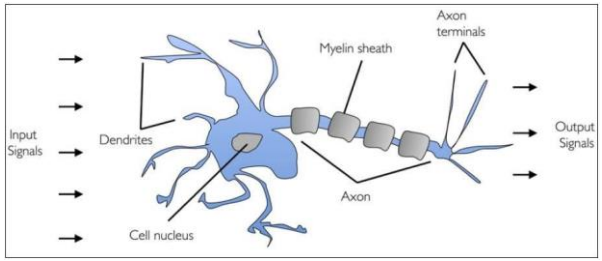
❐ **Softmax regression** — A softmax regression, also called a multiclass logistic regression, is used to generalize logistic regression when there are more than 2 outcome classes. By convention, we set $\theta_K = 0$, which makes the Bernoulli parameter ϕ_i of each class i be such that:

$$\boxed{\phi_i = \frac{\exp(\theta_i^T x)}{\sum_{j=1}^{K} \exp(\theta_j^T x)}}$$

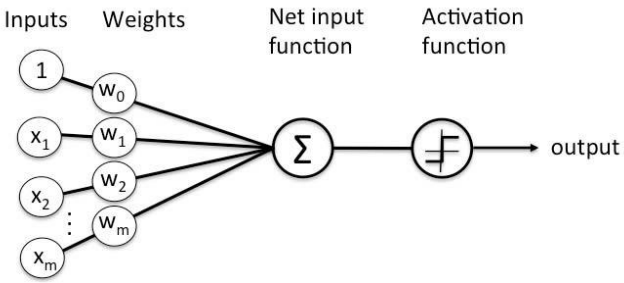
Classification and logistic regression

Perceptrons: Single layer and Multilayer.

- Single layer - Single layer perceptrons can learn only linearly separable patterns
- Multilayer - Multilayer perceptrons or feedforward neural networks with two or more layers have the greater processing power



- Perceptron Learning Rule/ Algorithm
- Pockets Algorithm
- Convergence of PLA



Linearly fractionable data set

Classification metrics

❒ **Confusion matrix** — The confusion matrix is used to have a more complete picture when assessing the performance of a model. It is defined as follows:

		Predicted class	
		+	-
Actual class	+	**TP** True Positives	**FN** False Negatives Type II error
	-	**FP** False Positives Type I error	**TN** True Negatives

Classification metrics

❒ **Main metrics** — The following metrics are commonly used to assess the performance of classification models:

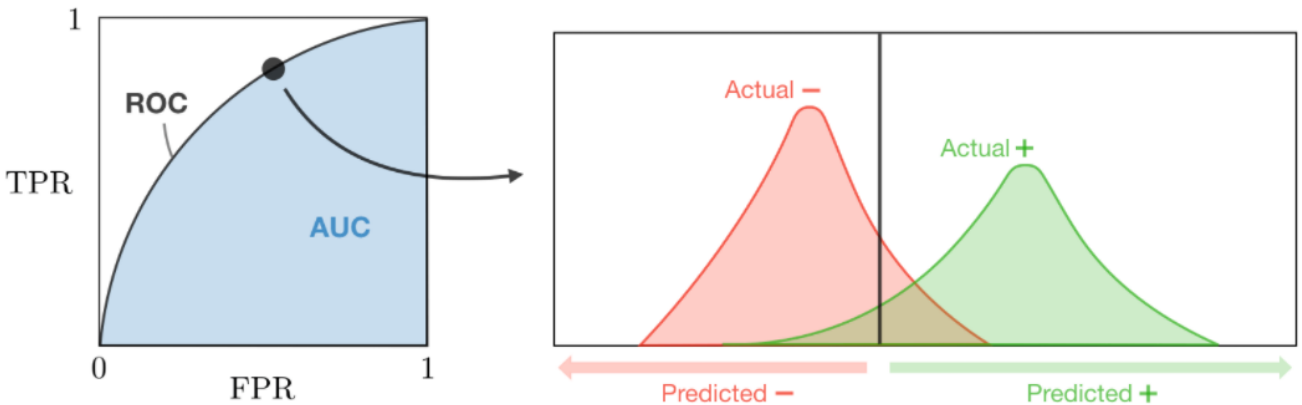
Metric	Formula	Interpretation
Accuracy	$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$	Overall performance of model
Precision	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	Coverage of actual positive sample
Specificity	$\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$	Coverage of actual negative sample
F1 score	$\frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$	Hybrid metric useful for unbalanced classes

Classification metrics

❒ **ROC** — The receiver operating curve, also noted ROC, is the plot of TPR versus FPR by varying the threshold. These metrics are are summed up in the table below:

Metric	Formula	Equivalent
True Positive Rate TPR	$\frac{\text{TP}}{\text{TP}+\text{FN}}$	Recall, sensitivity
False Positive Rate FPR	$\frac{\text{FP}}{\text{TN}+\text{FP}}$	1-specificity

❒ **AUC** — The area under the receiving operating curve, also noted AUC or AUROC, is the area below the ROC as shown in the following figure:



Regression metrics

❒ **Basic metrics** — Given a regression model f, the following metrics are commonly used to assess the performance of the model:

Total sum of squares	Explained sum of squares	Residual sum of squares
$\mathrm{SS}_{\mathrm{tot}} = \sum_{i=1}^{m} (y_i - \overline{y})^2$	$\mathrm{SS}_{\mathrm{reg}} = \sum_{i=1}^{m} (f(x_i) - \overline{y})^2$	$\mathrm{SS}_{\mathrm{res}} = \sum_{i=1}^{m} (y_i - f(x_i))^2$

❒ **Coefficient of determination** — The coefficient of determination, often noted R^2 or r^2, provides a measure of how well the observed outcomes are replicated by the model and is defined as follows:

$$\boxed{R^2 = 1 - \frac{\mathrm{SS}_{\mathrm{res}}}{\mathrm{SS}_{\mathrm{tot}}}}$$

Classification metrics

❒ **Main metrics** — The following metrics are commonly used to assess the performance of regression models, by taking into account the number of variables n that they take into consideration:

Mallow's Cp	AIC	BIC	Adjusted R^2
$\frac{\mathrm{SS}_{\mathrm{res}} + 2(n+1)\widehat{\sigma}^2}{m}$	$2\Big[(n+2) - \log(L)\Big]$	$\log(m)(n+2) - 2\log(L)$	$1 - \frac{(1-R^2)(m-1)}{m-n-1}$

where L is the likelihood and $\widehat{\sigma}^2$ is an estimate of the variance associated with each response.

Contents

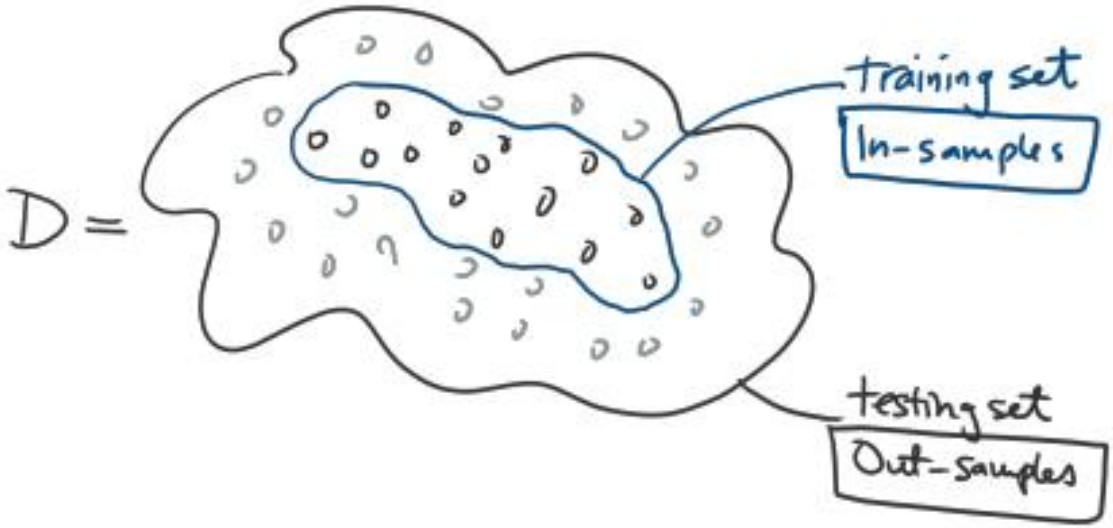
- Generative VS. Generative Learning
- What constitutes a learning problem?
- Is learning feasible?
- Is learning feasible?

Dataset

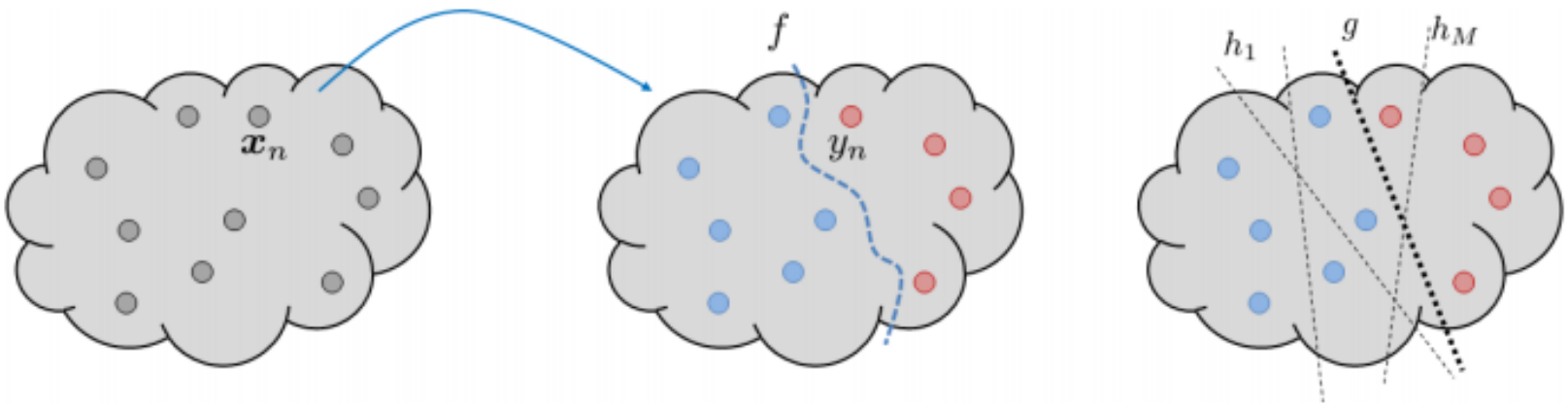
- **Dataset**
  - Input vectors: $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$
  - Labels: $y_1, \ldots, y_N$
  - Training set: $\mathcal{D}$
  - Target function f: Maps $\boldsymbol{x}_n$ to y_n
  - Target function is always **unknown** to you

Training and Testing Set

- **Training and Testing Set**
  - **In-sample:** Samples that are inside the training set
  - **Out-sample:** Samples that are outside the training set

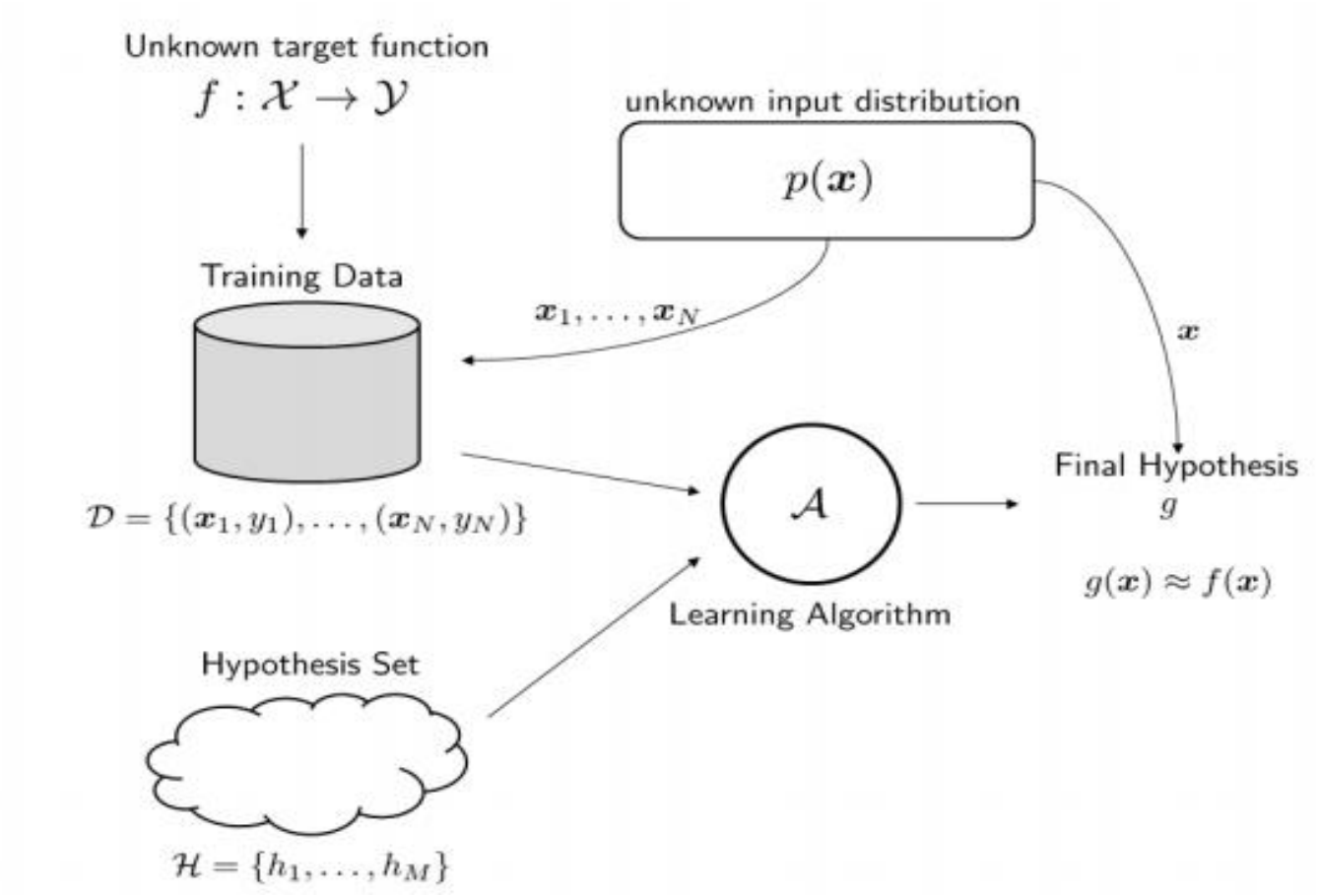


HypothesisFucntion



- Hypothesis set: $\mathcal{H} = \{h_1, \ldots, h_M\}$: Possible decision boundaries
- Algorithm: Picks h_m from $\mathcal{H}$
- Final hypothesis: g: The one you found

Learning Model



Is Learning Feasible?

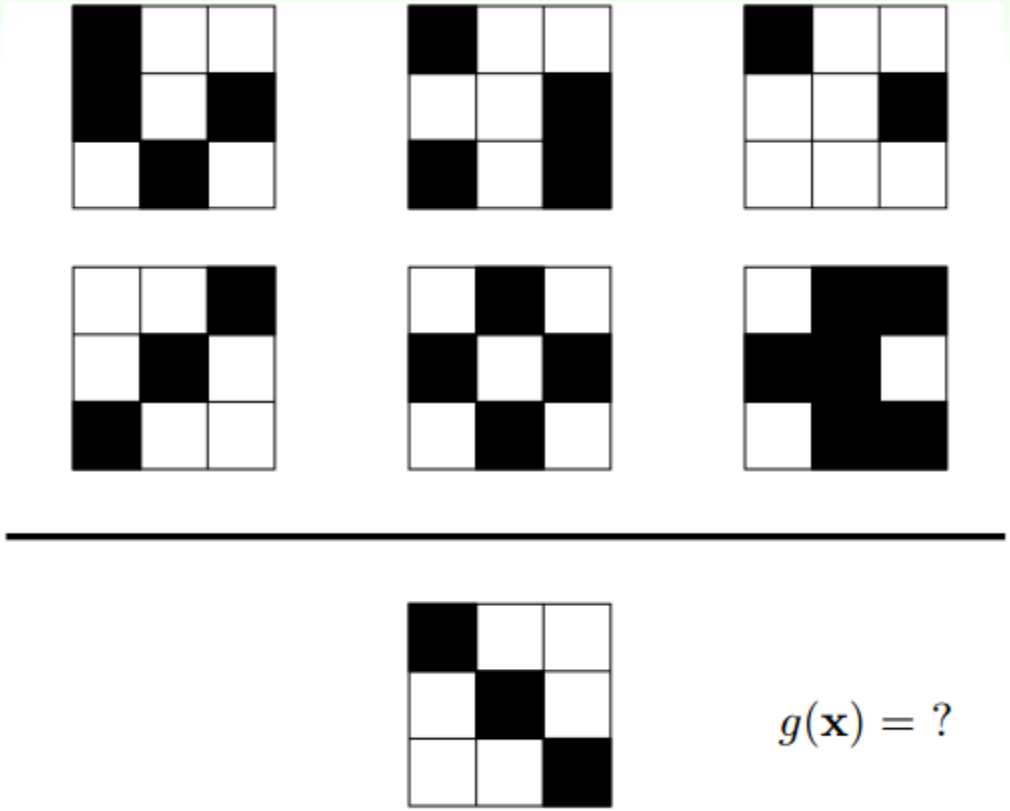
- **In-sample** and **Out-sample**:
  - **In-sample**: Training Data
  - **Out-sample**: Testing Data

When can we claim "learning is feasible"?

Suppose we have a training set $\mathcal{D}$, can we learn the target function f?

- "Learn" means: I use the data you give me to come up with an f
- "Successful" means: All in-samples are correctly predicted
- And all out-samples are also correctly predicted
- If YES, then we are in business.
- Learning is feasible!
- If NO, then we can go home and sleep.
- There is just no way to learn f from $\mathcal{D}$.

Is Learning Feasible?



$y_n = -1$

$y_n = +1$

$g(\mathbf{x}) = ?$

A ‘Simple’ Binary Classification Problem

$\mathbf{x}_n$	$y_n = f(\mathbf{x}_n)$
0 0 0	$\circ$
0 0 1	$\times$
0 1 0	$\times$
0 1 1	$\circ$
1 0 0	$\times$

- $\mathcal{X} = \{0, 1\}^3$, $\mathcal{Y} = \{\circ, \times\}$, can enumerate all candidate f as $\mathcal{H}$

- How many possible vectors are there?
- How many possible f’s ?

A ‘Simple’ Binary Classification Problem

- We have 8 input vectors:
- We have 256 hypotheses:
- **Is learning feasible?**

$\mathcal{D}$

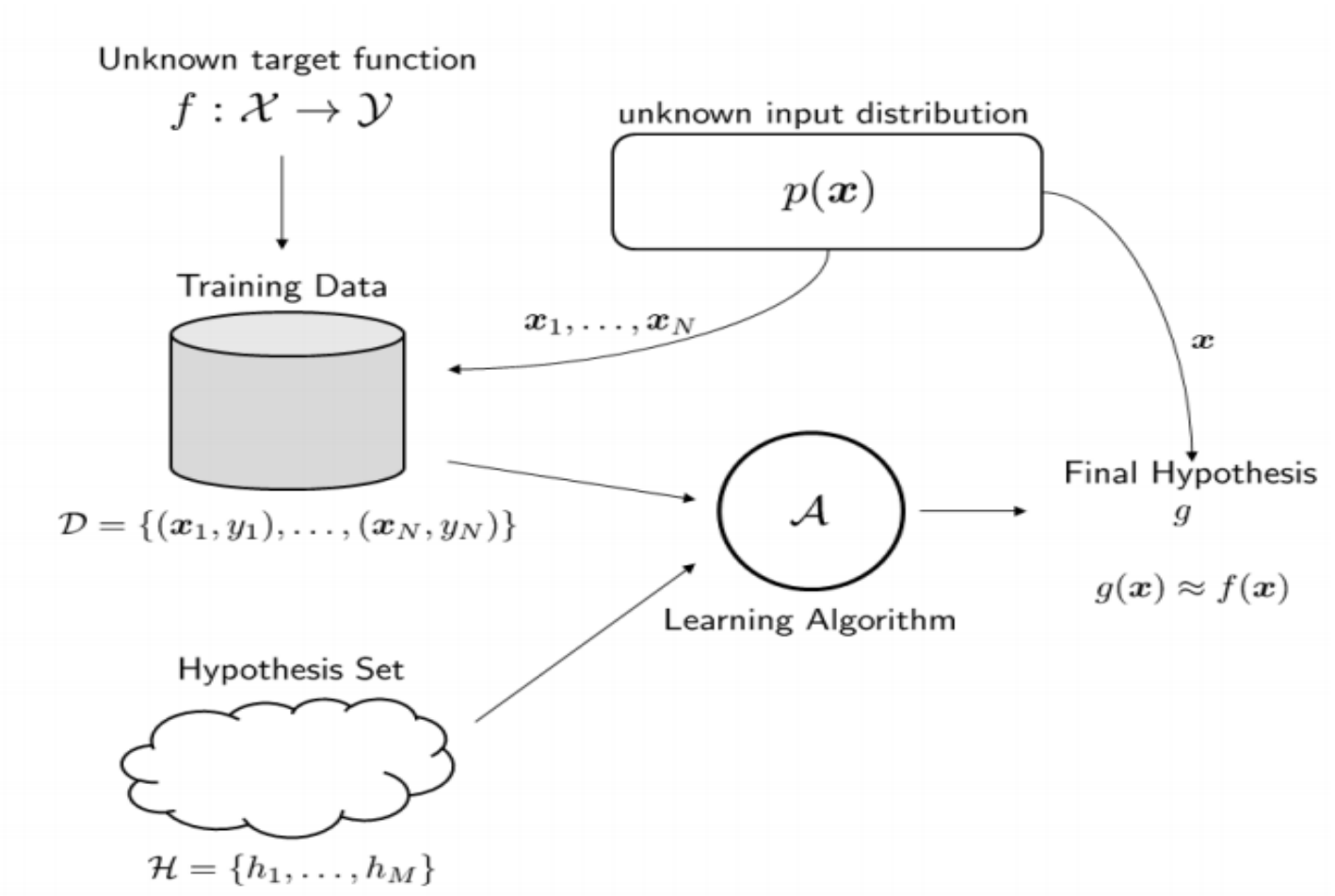
$\mathbf{x}$	y	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0 0 0	○	○	○	○	○	○	○	○	○	○
0 0 1	×	×	×	×	×	×	×	×	×	×
0 1 0	×	×	×	×	×	×	×	×	×	×
0 1 1	○	○	○	○	○	○	○	○	○	○
1 0 0	×	×	×	×	×	×	×	×	×	×
1 0 1		?	○	○	○	○	×	×	×	×
1 1 0		?	○	○	×	×	○	○	×	×
1 1 1		?	○	×	○	×	○	×	○	×

A ‘Simple’ Binary Classification Problem

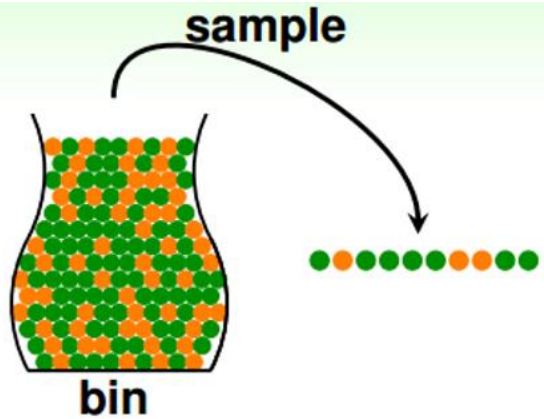
$\mathcal{D}$

$\mathbf{x}$	y	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0 0 0	○	○	○	○	○	○	○	○	○	○
0 0 1	×	×	×	×	×	×	×	×	×	×
0 1 0	×	×	×	×	×	×	×	×	×	×
0 1 1	○	○	○	○	○	○	○	○	○	○
1 0 0	×	×	×	×	×	×	×	×	×	×
1 0 1		?	○	○	○	○	×	×	×	×
1 1 0		?	○	○	×	×	○	○	×	×
1 1 1		?	○	×	○	×	○	×	○	×

The Power of Probability



The Power of Probability



bin

assume

orange probability = μ,
green probability = $1 - \mu$,

with μ **unknown**

sample

N marbles sampled independently, with

orange fraction = ν,
green fraction = $1 - \nu$,

now ν **known**

does **in-sample** ν say anything about out-of-sample μ?

The Power of Probability

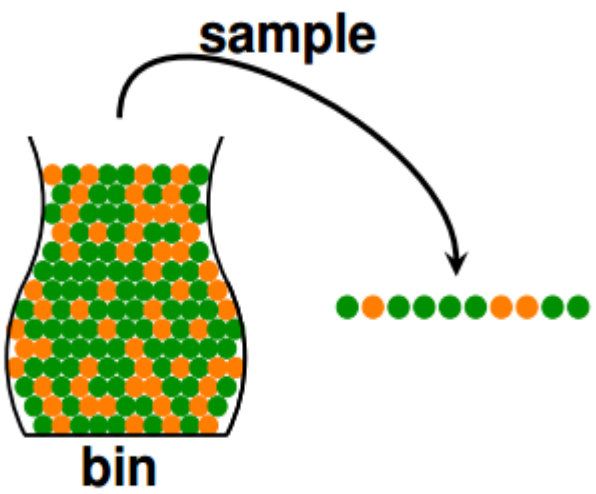
does **in-sample** ν say anything about out-of-sample μ?

No!

possibly not: sample can be mostly green while bin is mostly orange

Yes!

probably yes: in-sample ν likely **close to** unknown μ



formally, **what does ν say about μ**?

In-Sample Error

- Let $\boldsymbol{x}_n$ be a *training* sample
- h: Your hypothesis
- f: The unknown target function
- If $h(\boldsymbol{x}_n) = f(\boldsymbol{x}_n)$, then say training sample $\boldsymbol{x}_n$ is correctly classified.
- This will give you the **in-sample error**

Definition (In-sample Error / Training Error)

Consider a training set $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, and a target function f. The **in-sample error** (or the training error) of a hypothesis function $h \in \mathcal{H}$ is the empirical average of $\{h(\boldsymbol{x}_n) \neq f(\boldsymbol{x}_n)\}$:

$$E_{\text{in}}(h) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{n=1}^{N} [\![h(\boldsymbol{x}_n) \neq f(\boldsymbol{x}_n)]\!], \tag{1}$$

where $[\![\cdot]\!] = 1$ if the statement inside the bracket is true, and $= 0$ if the statement is false.

In-Sample Error

- Let $\boldsymbol{x}$ be a *testing* sample drawn from $p(\boldsymbol{x})$
- h: Your hypothesis
- f: The unknown target function
- If $h(\boldsymbol{x}) = f(\boldsymbol{x})$, then say testing sample $\boldsymbol{x}$ is correctly classified.
- Since $\boldsymbol{x} \sim p(\boldsymbol{x})$, you need to compute the probability of error, called the **out-sample error**

Definition (Out-sample Error / Testing Error)

Consider an input space $\mathcal{X}$ containing elements $\boldsymbol{x}$ drawn from a distribution $p_{\boldsymbol{X}}(\boldsymbol{x})$, and a target function f. The **out-sample error** (or the testing error) of a hypothesis function $h \in \mathcal{H}$ is

$$E_{\text{out}}(h) \stackrel{\text{def}}{=} \mathbb{P}[h(\boldsymbol{x}) \neq f(\boldsymbol{x})], \tag{2}$$

where $\mathbb{P}[\cdot]$ measures the probability of the statement based on the distribution $p_{\boldsymbol{X}}(\boldsymbol{x})$.

Out-Sample Error

- Let $\boldsymbol{x}$ be a *testing* sample drawn from $p(\boldsymbol{x})$
- h: Your hypothesis
- f: The unknown target function
- If $h(\boldsymbol{x}) = f(\boldsymbol{x})$, then say testing sample $\boldsymbol{x}$ is correctly classified.
- Since $\boldsymbol{x} \sim p(\boldsymbol{x})$, you need to compute the probability of error, called the **out-sample error**

Definition (Out-sample Error / Testing Error)

Consider an input space $\mathcal{X}$ containing elements $\boldsymbol{x}$ drawn from a distribution $p_{\boldsymbol{X}}(\boldsymbol{x})$, and a target function f. The **out-sample error** (or the testing error) of a hypothesis function $h \in \mathcal{H}$ is

$$E_{\text{out}}(h) \stackrel{\text{def}}{=} \mathbb{P}[h(\boldsymbol{x}) \neq f(\boldsymbol{x})], \tag{2}$$

where $\mathbb{P}[\cdot]$ measures the probability of the statement based on the distribution $p_{\boldsymbol{X}}(\boldsymbol{x})$.

In-sample VS Out-sample

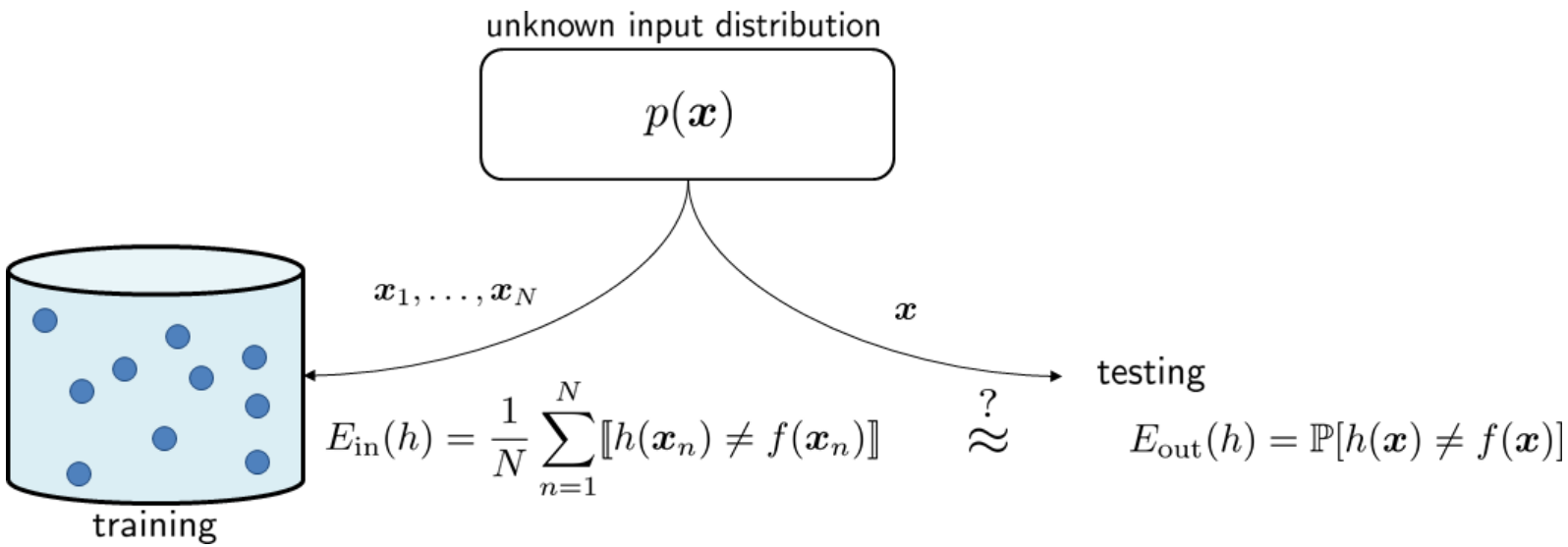
In-Sample Error

$$E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N} [\![h(\boldsymbol{x}_n) \neq f(\boldsymbol{x}_n)]\!]$$

Out-Sample Error

$$\begin{aligned}
E_{\text{out}}(h) &= \mathbb{P}[h(\boldsymbol{x}) \neq f(\boldsymbol{x})] \\
&= \underbrace{[\![h(\boldsymbol{x}_n) \neq f(\boldsymbol{x}_n)]\!]}_{=1} \mathbb{P}\Big\{ h(\boldsymbol{x}_n) \neq f(\boldsymbol{x}_n) \Big\} \\
&\quad + \underbrace{[\![h(\boldsymbol{x}_n) = f(\boldsymbol{x}_n)]\!]}_{=0} \Big(1 - \mathbb{P}\Big\{ h(\boldsymbol{x}_n) \neq f(\boldsymbol{x}_n) \Big\} \Big) \\
&= \mathbb{E}\Big\{ [\![h(\boldsymbol{x}_n) \neq f(\boldsymbol{x}_n)]\!] \Big\}
\end{aligned}$$

The Role of $p(x)$



- Learning is feasible if $\boldsymbol{x} \sim p(\boldsymbol{x})$
- $p(\boldsymbol{x})$ says: Training and testing are related
- If training and testing are unrelated, then hopeless – the deterministic example shown previously
- If you draw training and testing samples with different bias, then you will suffer

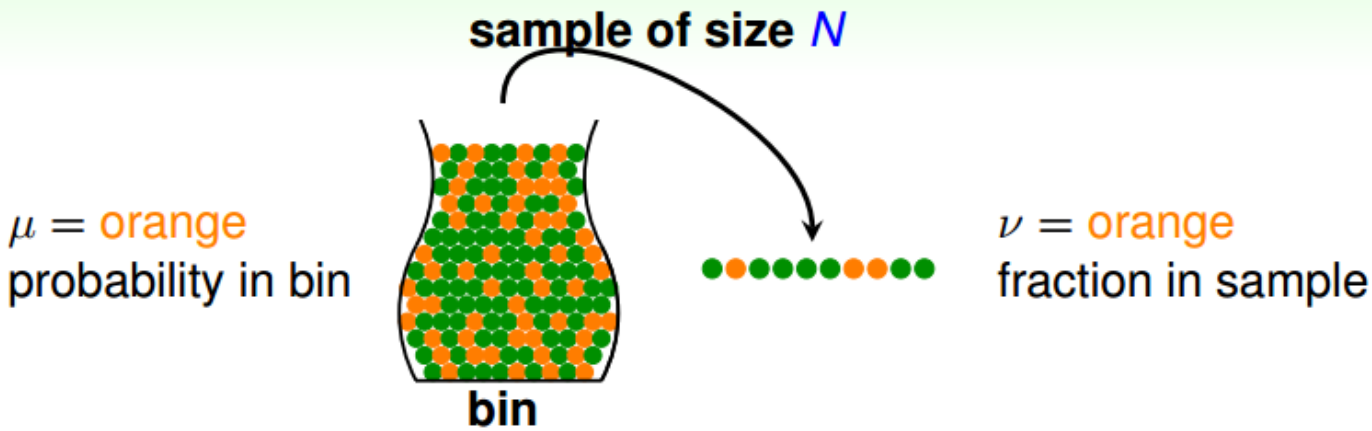
Theorem (Hoeffding Inequality)

Theorem (Hoeffding Inequality)

Let $X_1, \ldots, X_N$ be a sequence of i.i.d. random variables such that $0 \leq X_n \leq 1$ and $\mathbb{E}[X_n] = \mu$. Then, for any $\epsilon > 0$,

$$\mathbb{P}\left[\left|\frac{1}{N}\sum_{n=1}^{N} X_n - \mu\right| > \epsilon\right] \leq 2e^{-2\epsilon^2 N}. \tag{3}$$

When Will *E*in = *E*out?



- in big sample (N large), ν is probably close to μ (within ϵ)

$$\mathbb{P}\left[\left|\nu - \mu\right| > \epsilon\right] \leq 2\exp\left(-2\epsilon^2 N\right)$$

- called **Hoeffding's Inequality**, for marbles, coin, polling, ...

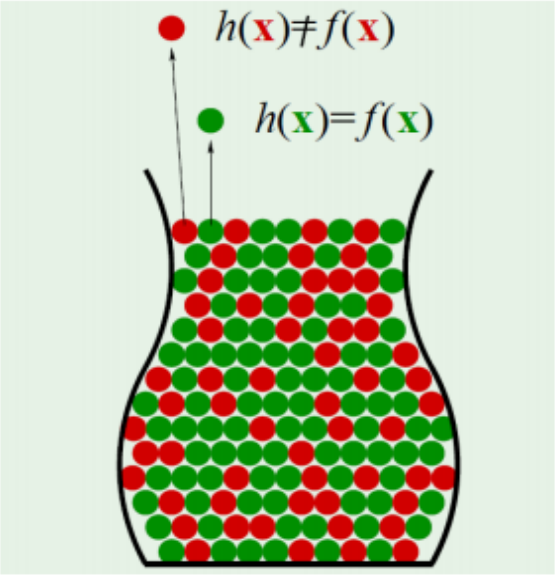
the statement '$\nu = \mu$' is
probably approximately correct (PAC)

When Will *E*in = *E*out?

- To us, the inequality can be stated as

$$\mathbb{P}\left[|E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon\right] \leq 2e^{-2\epsilon^2 N}.$$

- N = number of training samples
- ϵ = tolerance level
- Hoeffding is applicable because $[\![h(\boldsymbol{x}) \neq f(\boldsymbol{x})]\!]$ is either 1 or 0.



Connection to Learning

bin

- unknown orange prob. μ
- marble $\bullet \in$ bin
- orange $\bullet$
- green $\bullet$
- size-N sample from bin

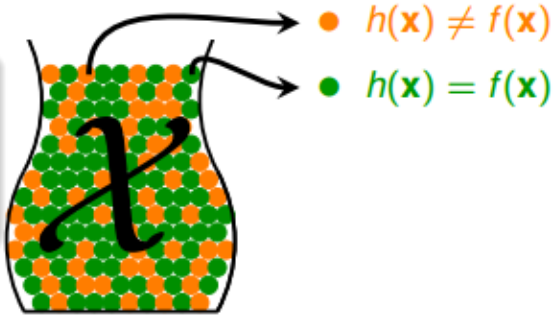
  of i.i.d. marbles

learning

- fixed hypothesis $h(\mathbf{x}) \stackrel{?}{=}$ target $f(\mathbf{x})$
- $\mathbf{x} \in \mathcal{X}$
- h is wrong $\Leftrightarrow h(\mathbf{x}) \neq f(\mathbf{x})$
- h is right $\Leftrightarrow h(\mathbf{x}) = f(\mathbf{x})$
- check h on $\mathcal{D} = \{(\mathbf{x}_n, \underbrace{y_n}_{f(\mathbf{x}_n)})\}$

  with i.i.d. $\mathbf{x}_n$

if **large** N **&** **i.i.d.** $\mathbf{x}_n$, can **probably** infer
unknown $[\![h(\mathbf{x}) \neq f(\mathbf{x})]\!]$ probability
by known $[\![h(\mathbf{x}_n) \neq y_n]\!]$ fraction



The Formal Guarantee

for any fixed h, in 'big' data (N large),

$\boxed{\text{in-sample error } E_{\text{in}}(h)}$ is probably close to

$\boxed{\text{out-of-sample error } E_{\text{out}}(h)}$ (within ϵ)

$$\mathbb{P}\left[\left|E_{\text{in}}(h) - E_{\text{out}}(h)\right| > \epsilon\right] \leq 2\exp\left(-2\epsilon^2 N\right)$$

same as the 'bin' analogy . . .

- valid for all N and ϵ
- does not depend on $E_{\text{out}}(h)$, **no need to 'know'** $E_{\text{out}}(h)$ —f and P can stay unknown
- '$E_{\text{in}}(h) = E_{\text{out}}(h)$' is **probably approximately correct (PAC)**

if '$E_{\text{in}}(h) \approx E_{\text{out}}(h)$' and '$E_{\text{in}}(h)$ **small**'
$\Longrightarrow E_{\text{out}}(h)$ small $\Longrightarrow h \approx f$ with respect to P

Verification of One h

for any fixed h, when data large enough,

$$E_{\text{in}}(h) \approx E_{\text{out}}(h)$$

Can we claim 'good learning' ($g \approx f$)?

Yes!

if $E_{\text{in}}(h)$ **small for the fixed** h
and $\mathcal{A}$ **pick the** h **as** g
$\Longrightarrow$ '$g = f$' PAC

No!

if $\mathcal{A}$ **forced to pick THE** h **as** g
$\Longrightarrow$ $E_{\text{in}}(h)$ **almost always not small**
$\Longrightarrow$ '$g \neq f$' PAC!

real learning:
$\mathcal{A}$ shall **make choices** $\in \mathcal{H}$ (like PLA)
rather than **being forced to pick one** h. **:-(**