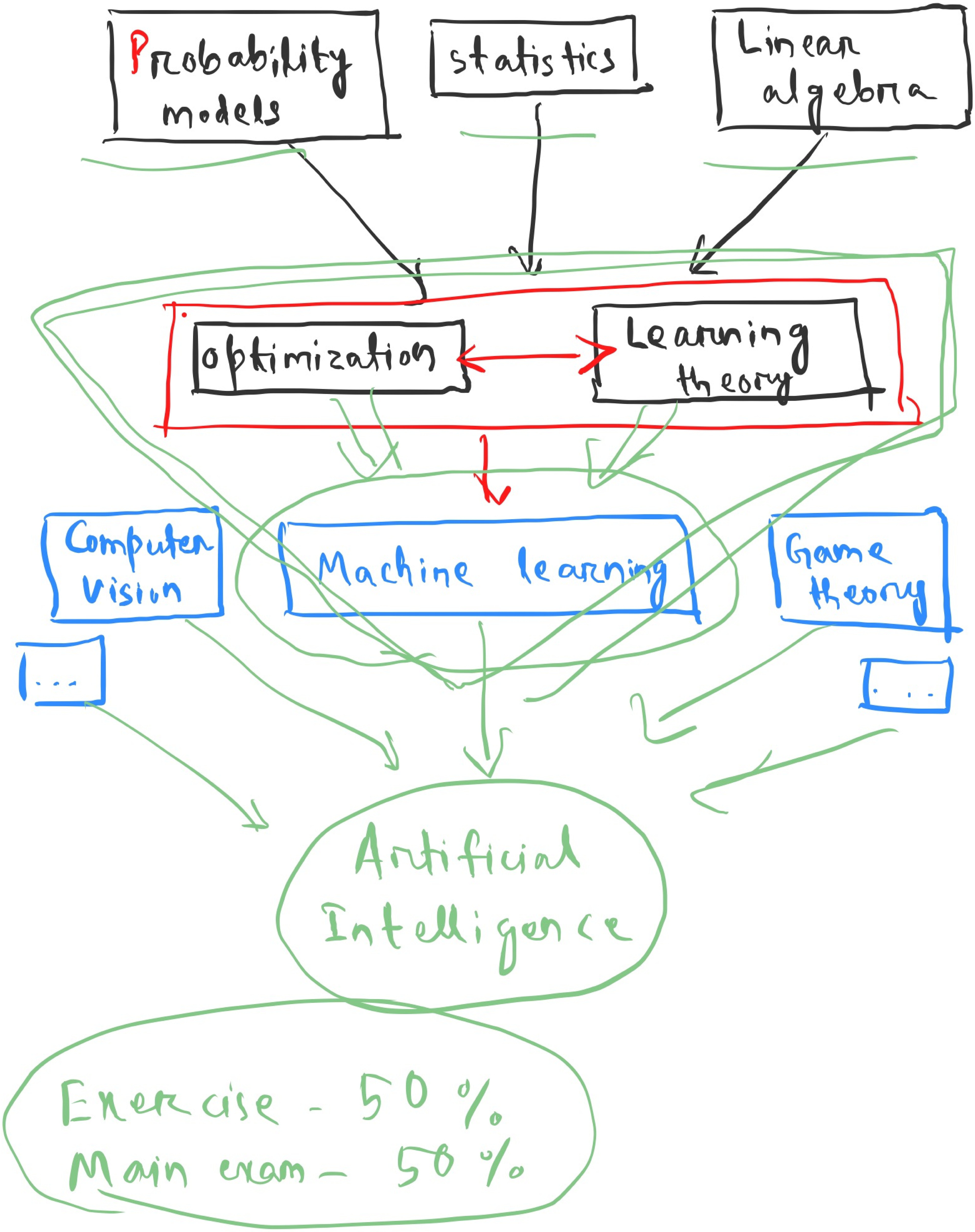
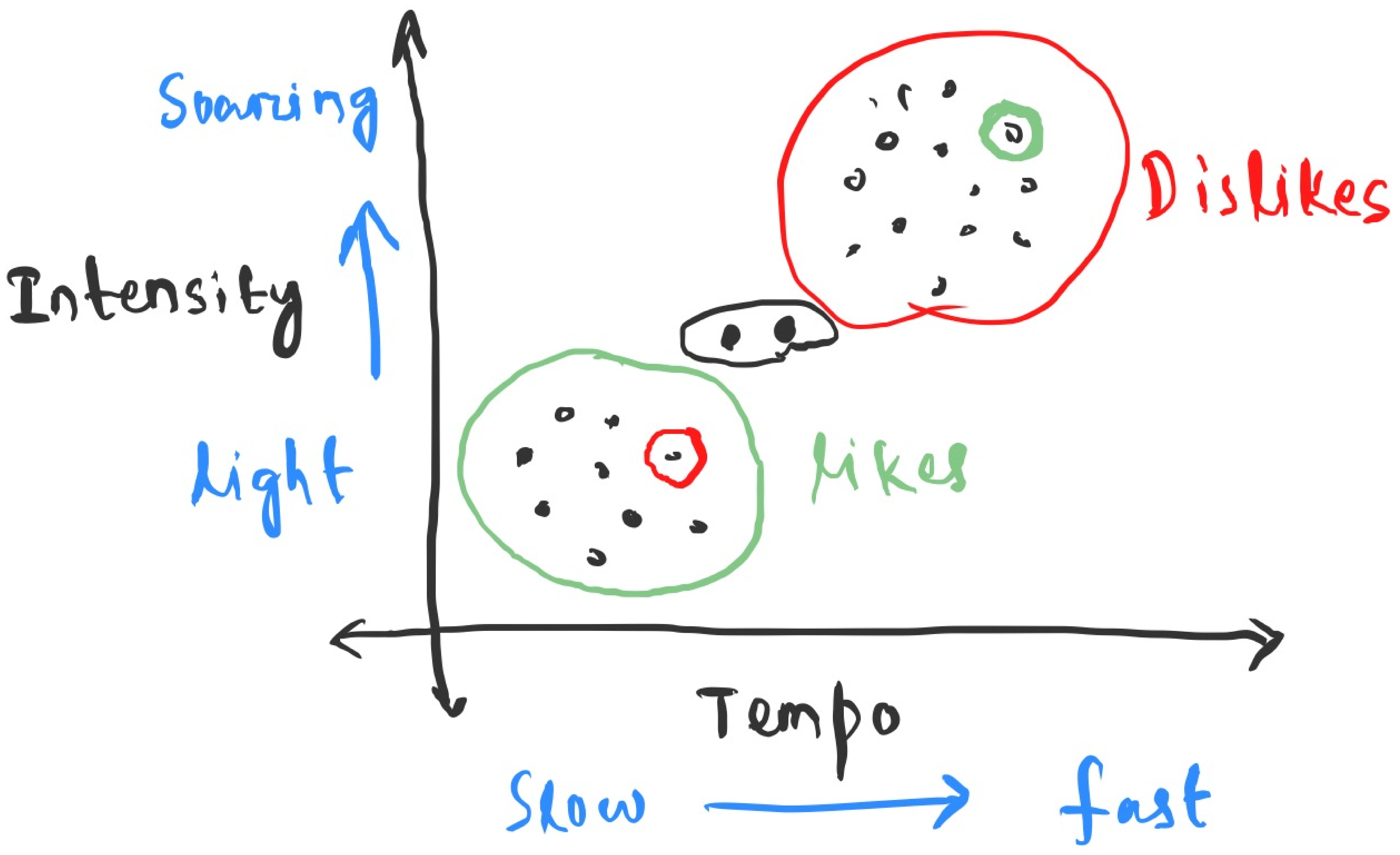

Probability models
statistics
Linear algebra
optimization
Learning theory
Computer vision
Machine learning
Game theory
. . .
. . .
Artificial Intelligence
Exercise - 50 %
Main exam - 50 %

1) Probably approximately correct (PAC) learning framework
2) Vapnik-Chervonenkis (VC) dimension
3) Neural networks
4) Stochastic gradient descent
5) Support vector machines
6) Kernel methods

Question - What is ML and why ML ?



<u>Machine learning</u> : Computational methods using experience to improve performance or to make predictions.

$\downarrow$ <u>Past information available</u>

quality and size are crucial

<u>Aim</u> : designing efficient and accurate algorithms.

Critical measures of the quality of algorithms.

$$\begin{tikzcd}
\text{time complexity} \arrow[r] \arrow[rdd] & \text{other area in computer science} \\
\text{space complexity} \arrow[ru] \arrow[rd] & \\
\text{Sample complexity} \arrow[r] & \text{ML.}
\end{tikzcd}$$

Question - What kind of problems can be tackled using ML?

1) Text or document classification
2) Natural language Processing (NLP)
3) Speech processing applications
4) Computer vision applications
5) Computational biology applications
6) Many problems (i.e., fraud detection, insurance companies, learning to play games (chess), Search engines, self driving cars etc.)

Question : Standard learning tasks ?

1) classification : It is the problem of assigning category to each item.

example - Image classification.

2) Regression : It is the problem of predicting a real value for each item.

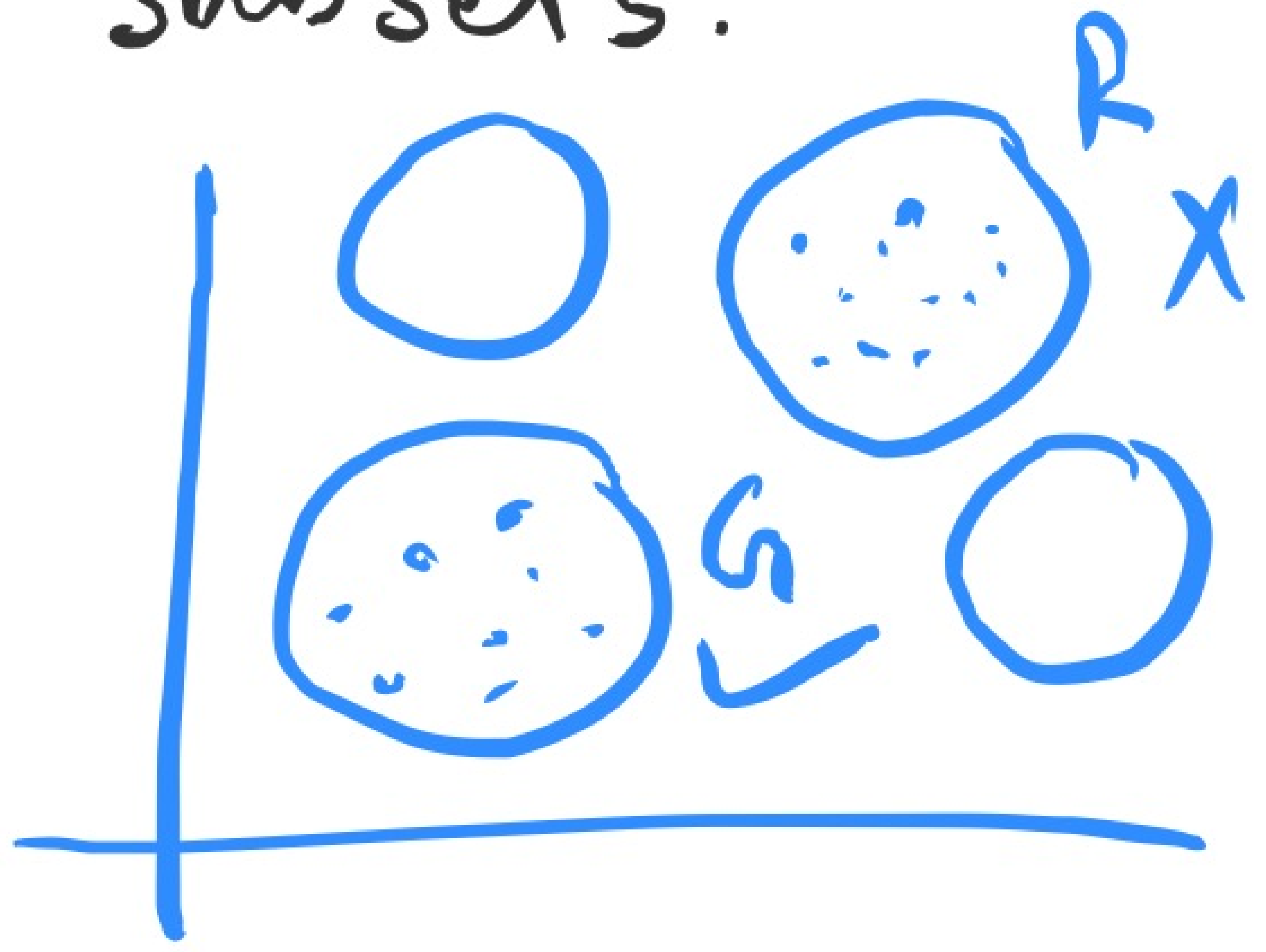
Example - [h, w] , [h₁, w₁] = R

3) Ranking : It is the problem of learning to order items according to some criterion.

Example - Search engine.

4) Clustering : It is the problem of partitioning a set of items into homogeneous subsets.

Example -



5) Dimensionality reduction : This problem consists of an initial representation of items into a lower- dimensional representation.

Example - Buying a house

Practical objective : • Generating accurate predictions for unseen items.

• designing efficient and robust algorithms to produce these predictions, even for large-scale problems.

Questions - • Which concept families can be learned actually ?

• Under what conditions ?

• How well can these concepts be learned computationally ?

Learning stages : (spam detection)

1) Examples : Items or instances of data used for learning or evaluation.

→ Collection of emails we will use for learning or testing.

2) Features : The set of attributes, often represented as a vector, associated to the examples.

3) Labels : Values or categories assigned to examples.

Example - Spam - +1 , non-spam - −1

4) Hyperparameters : Free parameters that are not determined by the learning algorithm, but rather specified as inputs to the learning algorithm.

5) Training sample : Examples used to train a learning algorithm.

→ a set of email examples along with their associated labels.

6) Validation sample : Examples used to tune the parameters of a learning algorithm when working with labeled data.

(Used to select appropriate values for hyperparameters)

7) Test sample : Examples used to evaluate the performance of a learning algorithm. The test sample is separate from the training and validation data and is not made available in the learning stage.

→ Collection of emails for which the learning algorithm must predict labels based on features. These predictions are then compared with the labels of the test sample to measure the performance of algorithm.

8) Loss function : A function that measures the difference, or loss, between a predicted label and a true label. Denoting the set of all lables as Y and the set of possible predictions as Y', a loss function L is a mapping

$$L : Y \times Y' \longrightarrow \mathbb{R}_+ . \quad (\text{bdd.})$$

Example - 1) zero-one loss

$\{-1,+1\} \times \{-1,+1\}$

by $L(y,y') = \mathbb{1}_{y' \neq y}$

2) Squared loss, defined over some bounded interval $J \subseteq \mathbb{R}$ by

$L(y,y') = (y'-y)^2$.

9) Hypothesis set :

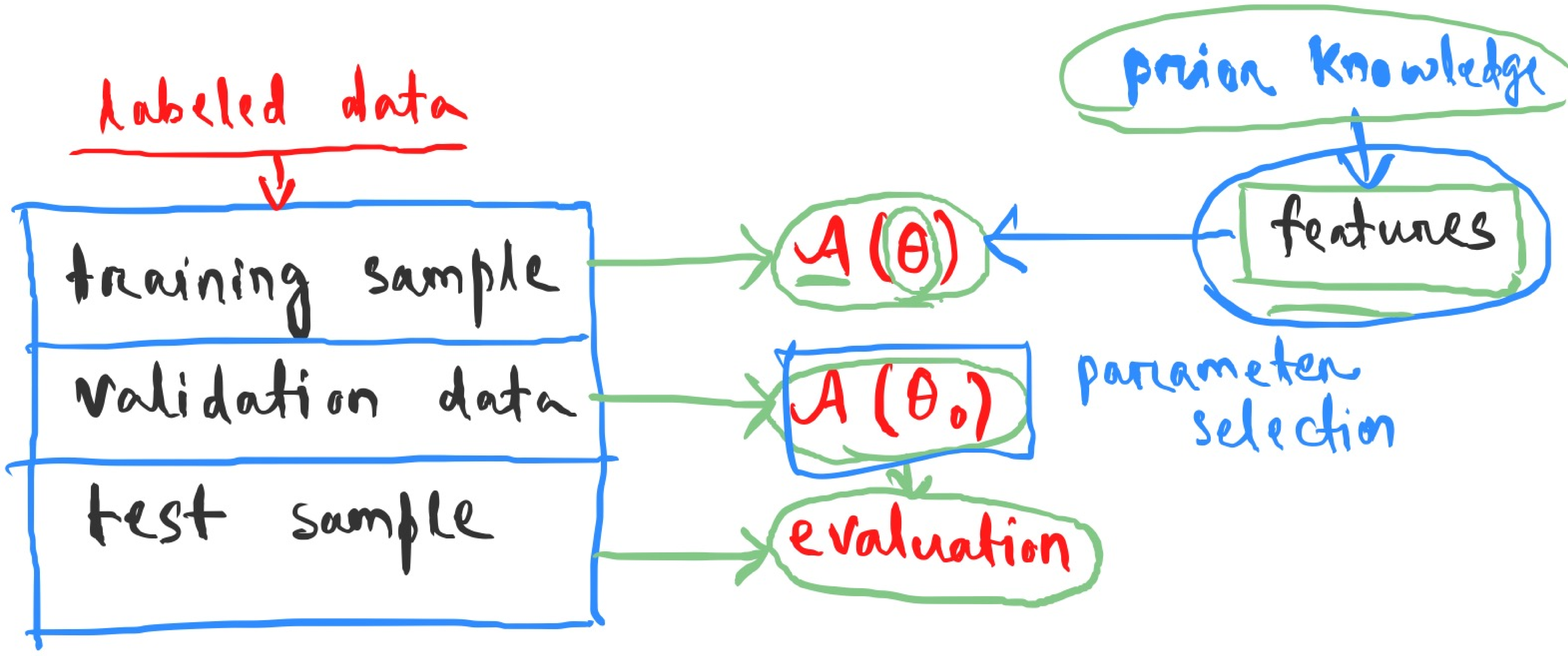
A set of functions mapping features (feature vectors) to the set of labels Y.

→ These may be a set of functions mapping email features to

$Y = \{\text{spam} = -1, \text{non-spam} = +1\}$.

→ Linear functions mapping email feature vectors to real numbers interpreted as scores.

with higher score ⇒ spam.



→ randomly partition the data.

→ Validation data depends on hyperparameter.
hyperparameter (θ).

→ Small sample (training data > test data)
because learning performance
$\Leftrightarrow$ training sample.

→ Associate relevant features to the examples tuning the values of hyperparameters θ.

→ choose the one resulting in the best performance on the validation sample (θ_0).

→ Using the hypothesis, predict the labels of the examples in the test sample.

→ The performance of algorithm ← loss func.

Learning scenarios -

Supervised learning : The learner receives a set of labeled examples as training data and makes predictions for all unseen points.

Unsupervised learning :

Receives unlabeled data . . . . . .

Semi-supervised learning :

Receives a training sample consisting of both labeled and unlabeled data.

Transductive inference :

As in the semi-supervised, the learner receives a labeled training sample along with a set of unlabeled test points. Objective is to predict labels only for these particular test points.

On-line learning :

Involves multiple rounds where training and testing phases are intermixed. At each round, the learner receives an unlabeled training point, makes a prediction, receives the true label, & incurs a loss. The objective is to minimize the cumulative loss over all rounds.

Reinforcement learning :

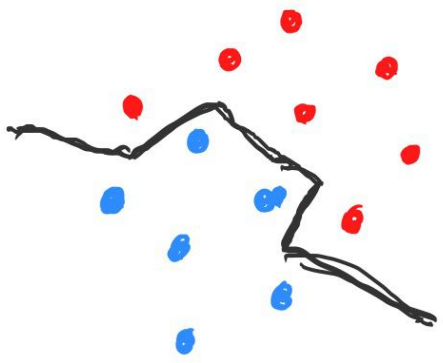
The training and testing phases are intermixed. To collect information, the learner actively interacts with the enviroment and in some cases affects the environment, and receives an immediate reward for each action. The objective is to maximize his reward over a course of actions and interactions.

exploration vs exploatation

Active learning :

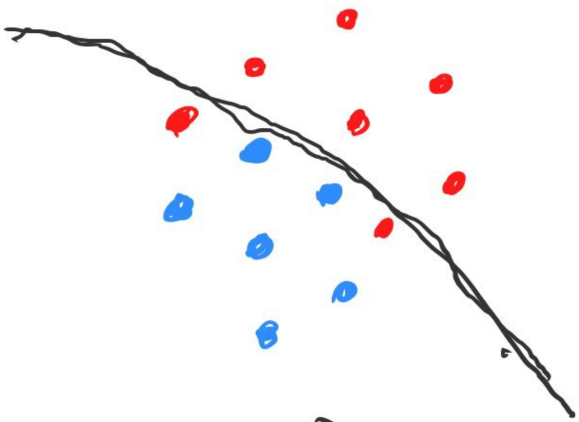
The learner adaptively or interactively collects training samples. The goal is to achieve a performance comparable to the standard supervised learning; but with a fewer labeled examples.

Used in - where the labels are expensive.



(a)

over-fitting



(b)

under-fitting

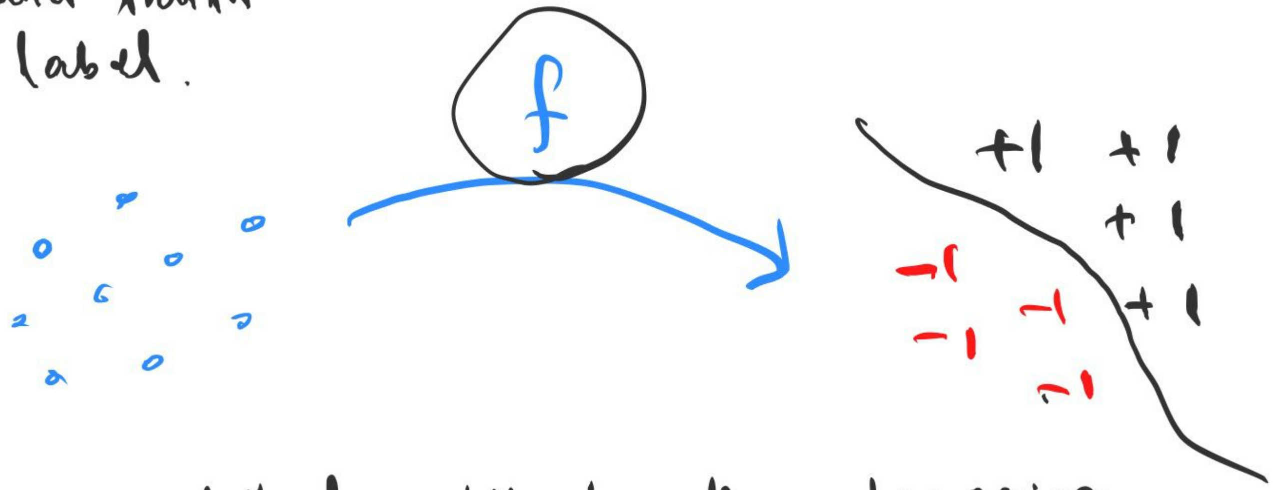
Dataset :

- Input vectors : $\underline{x_1, \ldots, x_N}$ (x_i are high-dim. vectors)
- Labels : $\underline{y_1, \ldots, y_N}$ (binary)
- Training set : $D = \{ (\underline{x_n}, \underline{y_n}) \}_{n=1}^{N}$
- Target function f :

  maps x_n to y_n

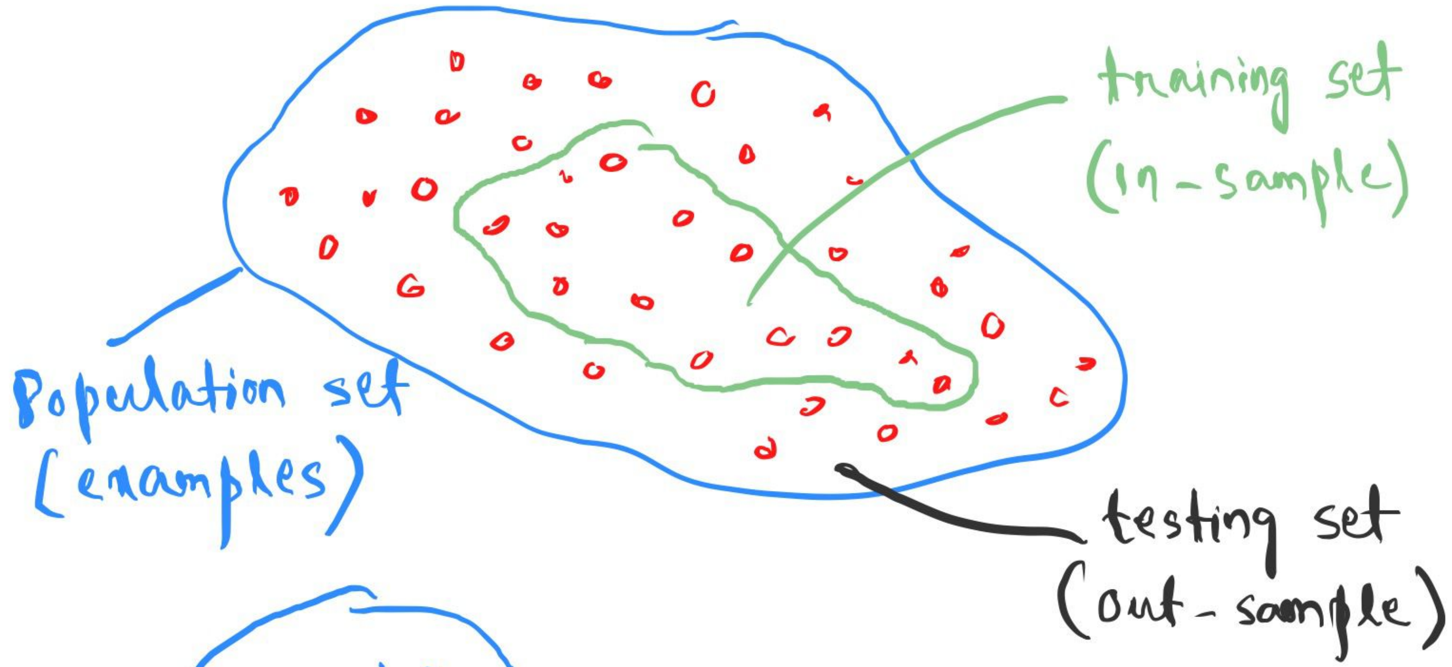
(unknown)

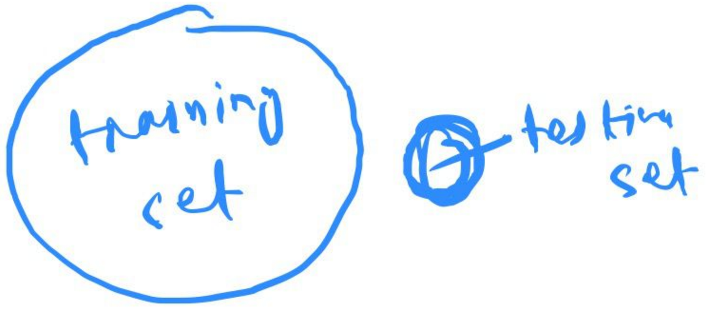
$$\underbrace{y_n}_{\text{ground truth label}} = f(\underbrace{x_n}_{\text{inputs}})$$

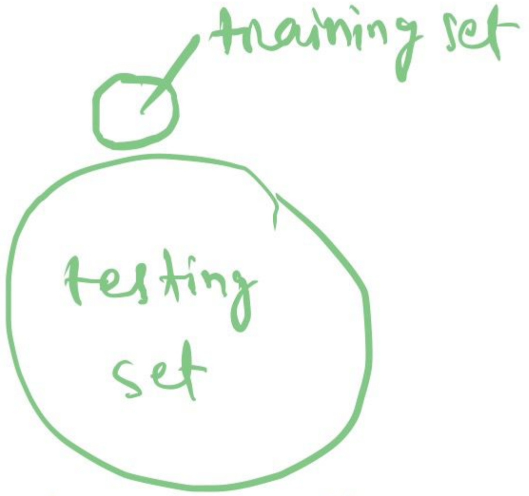


Question : What will be the learning problem?

Training and testing set







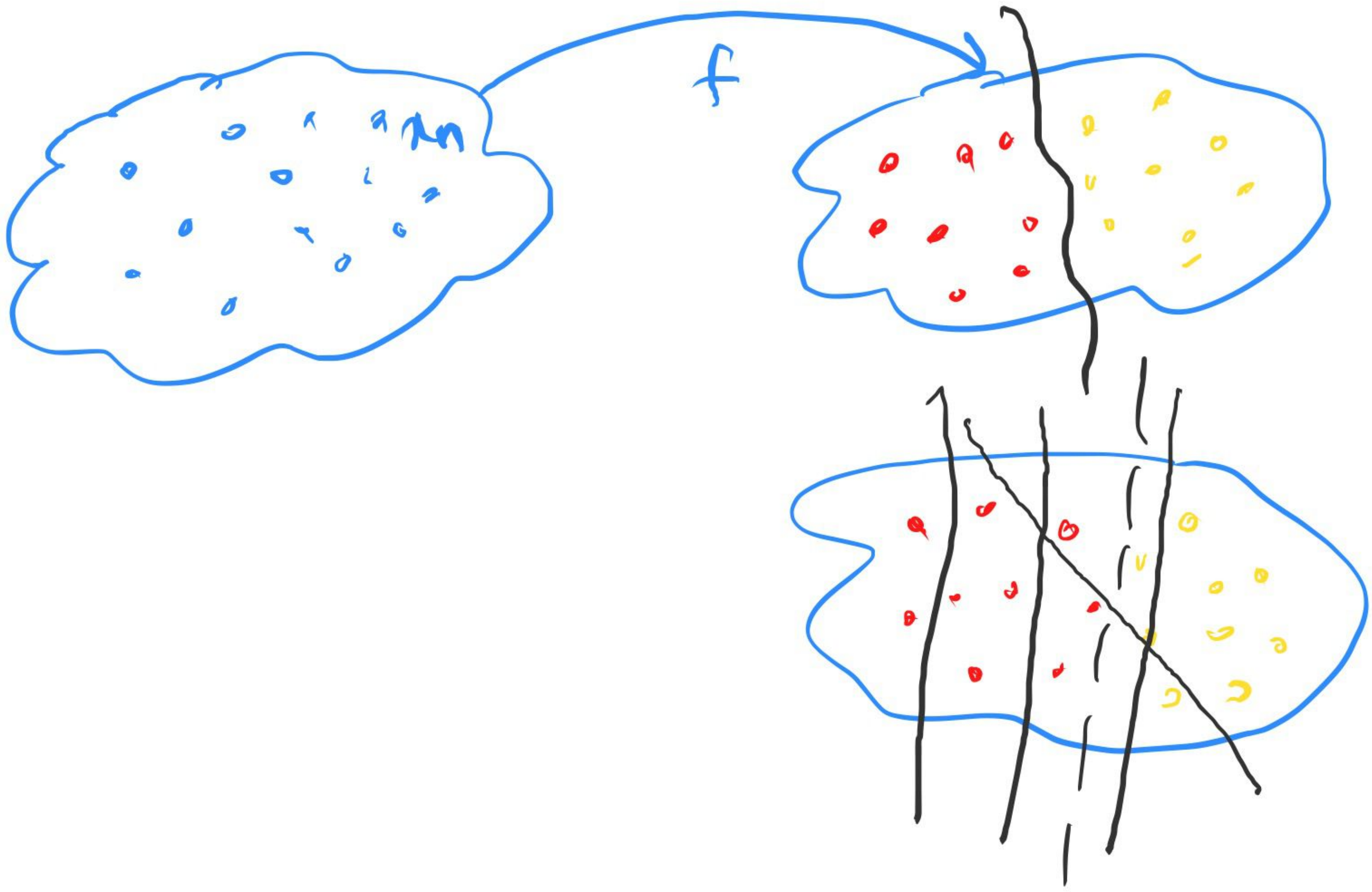
→ The training and the testing set has to share some properties in order to run the algorithm.

- Hypothesis set :

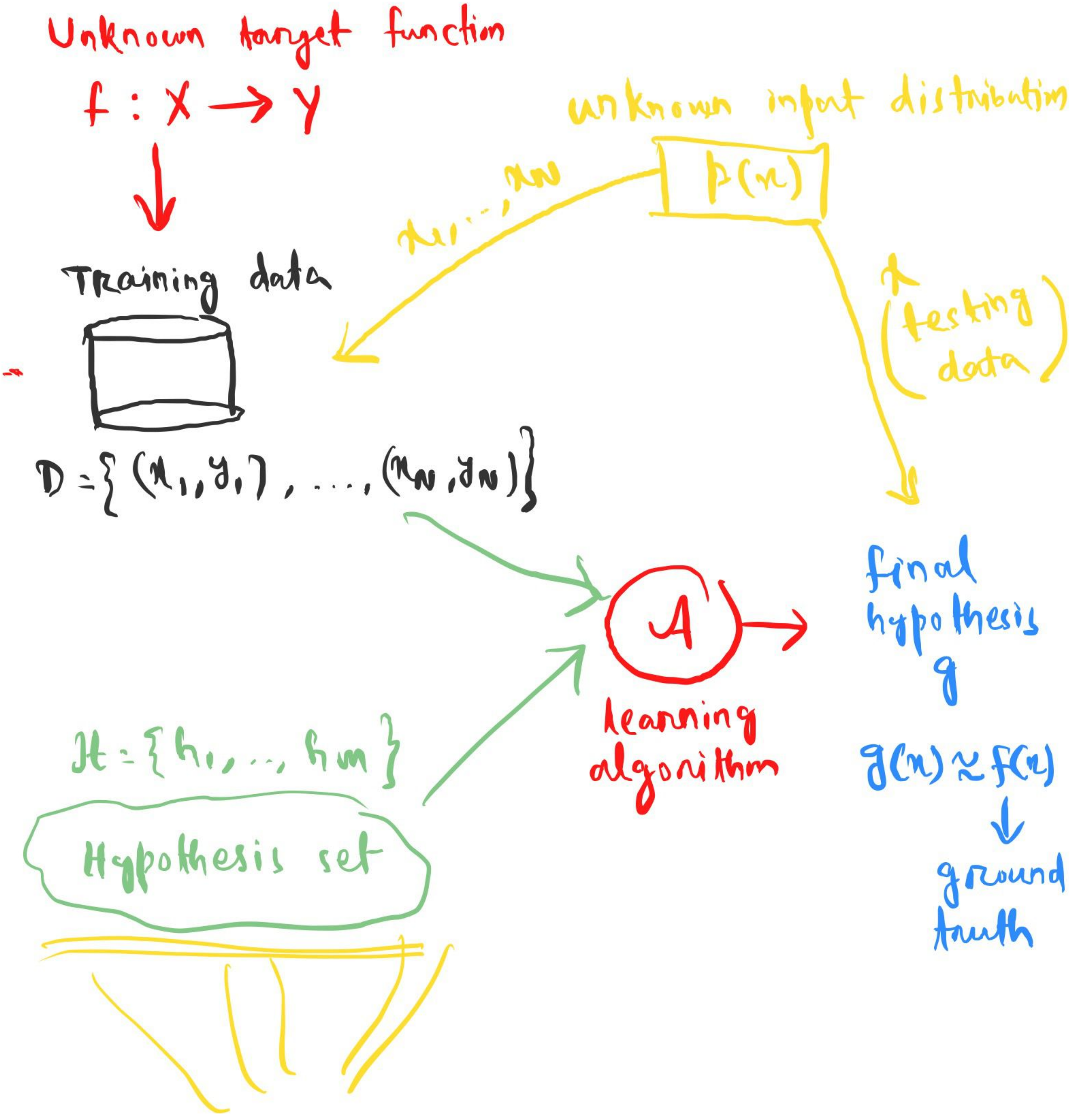
  $\mathcal{H} := \{h_1, \ldots, h_M\}$: possible decision boundaries

  Can be finite or infinite

- Learning algorithm : picks h_m from $\mathcal{H}$.
- Final hypothesis : The one you found.



Learning model :



Question : When can we claim "learning is feasible" ?

Suppose we have a training set D, can we learn the target function f ?

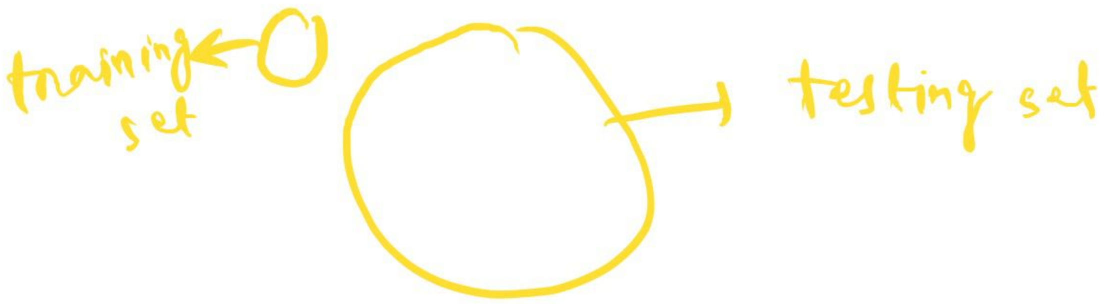
→ 'Learn' means : I use the data you give me to come up with an f.

→ 'Successful' means : All in-samples are correctly predicted.

and also out-samples are correctly predicted.

→ If Yes, learning is feasible.

If No, " " not-feasible.



Example :

- Let $X = \{0,1\}^3$, each $x \in X$ is a binary vector like $x = [0,0,1]^T$ or $x = [1,0,1]^T$ etc.
- There are $2^3 = 8$ vectors, we call them $x_1, \dots, x_8$.
- Suppose there is a target func f which is unknown. That will map x to y, where $y = \{+1, -1\}$.

  ex - $f([0,0,1]) = +1$

  $f([0,1,1]) = -1$ etc.
- How many possible f's ?

$$f = [+1, -1, -1, +1, -1, -1, +1, -1]$$

$2^8 = 256$ possible f's

(Hypothesis set has 256 choices & you are asked to pick 1)

Visual understanding –

- We have 8 input vectors & 256 hypotheses i.e., $\mathcal{H} = \{h_1, \dots, h_{256}\}$.
- Is learning feasible?
- Give me a subset $D \subset X$, can I find a hypothesis $g \in \mathcal{H}$ s.t. $g = f$.
- suppose 🔵 = -1, 🔴 = $+1$.

x_n			y_n
0	0	0	🔵
0	0	1	🔴
0	1	0	🔴
0	1	1	🔵
1	0	0	🔴
1	0	1	🔵
1	1	0	?
1	1	1	?

input vectors

training set

true labels

Training error (In-sample error):

- Let x_n be a training sample.
- h is your hypothesis
- f is unknown target function.
- If $h(x_n) = f(x_n)$, then the training sample x_n is correctly classified.

Def^n: Consider a training set $\mathcal{D} = \{x_1, \dots, x_N\}$, and a target function f. The training error of a hypothesis func^c $h \in \mathcal{H}$ is the empirical average of $\{h(x_n) \neq f(x_n)\}$:

$$E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}_{\{h(x_n) \neq f(x_n)\}}$$

Testing error (Out-sample error)

- Let x be a testing sample drawn from $p(x)$.
- h is your hypothesis (same)

- f is unknown target function.
- If $h(x) = f(x)$, then x is correctly classified.
- Since $x \sim p(x)$, you need to compute the probability of error.

<u>Defⁿ</u>: Consider an input space X consisting of elements x drawn from a distribution $P_X(x)$, and a target function f. The out-sample error of a hypothesis $h \in \mathcal{H}$ is

$$E_{out}(h) := \mathbb{P}[h(x) \neq f(x)].$$

Connection between E_{in} & E_{out}:

$$E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\{h(x_n) \neq f(x_n)\}$$

- f is unknown target func^c^tion.
- If $h(x) = f(x)$, then x is correctly classified.
- Since $x \sim p(x)$, you need to compute the probability of error.

<u>Def^n^</u>: Consider an input space X consisting of elements x drawn from a distribution $P_x(x)$, and a target function f. The out-sample error of a hypothesis $h \in \mathcal{H}$ is

$$E_{out}(h) := \mathbb{P}[h(x) \neq f(x)].$$

Connection between E_{in} & E_{out}:

$$\boxed{E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\{h(x_n) \neq f(x_n)\}}$$

$$E_{out}(h) = \mathbb{P}[h(x) \neq f(x)]$$

$$= \mathbb{1}_{\{h(x) \neq f(x)\}} \mathbb{P}[h(x) \neq f(x)] + \mathbb{1}_{\{h(x) = f(x)\}} \left(1 - \mathbb{P}[h(x) \neq f(x)]\right)$$

Bernoulli event

$$\text{R.V. } Z \rightarrow \begin{cases} 1 & \text{with } p \\ 0 & \text{with } 1-p \end{cases}$$

$$\mathbb{E}(Z) = (1)(p) + (0)\cdot(1-p) = p$$

$$= \mathbb{E}\left[\mathbb{1}_{\{h(x) \neq f(x)\}}\right].$$

→ As $n \to \infty$, these two in principle should account the same.

→ If $n << \infty$, then it's unclear how are they related.