1. what is machine learning ?

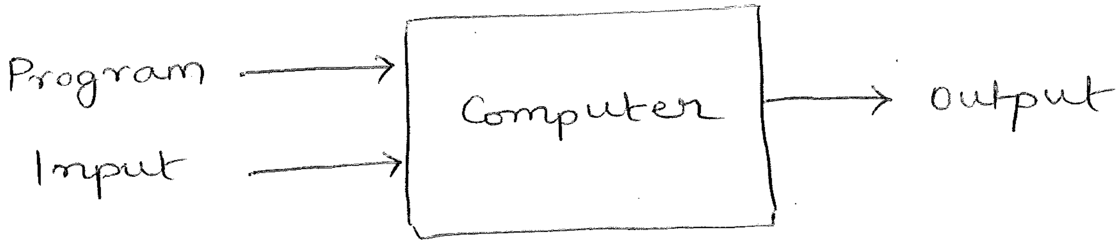
ML is the branch of engineering that develops technology for automated inference (prediction)

- It Combines

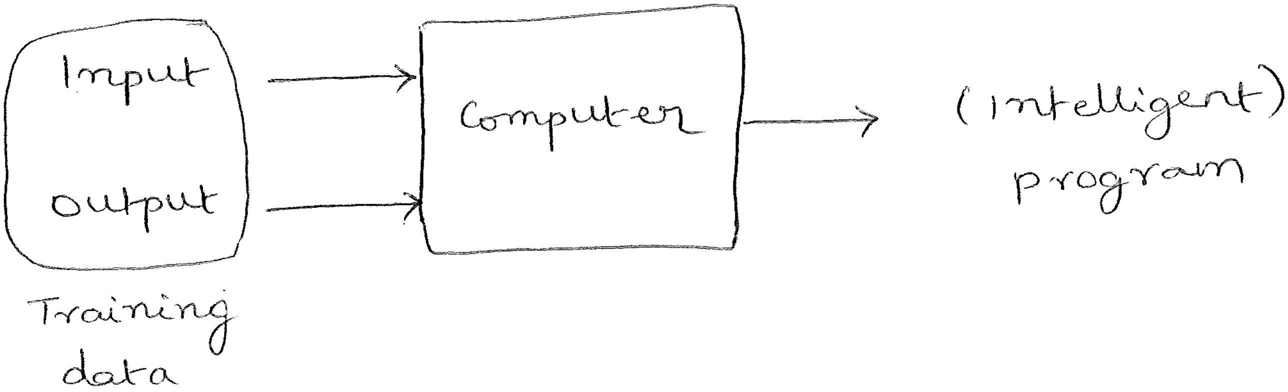
Probability + Statistics + Optimization + Algos

2. ML = Automating automation

Traditional programming



Machine learning



2. Learning paradigms

Supervised learning → main focus of our class

Semi-Supervised Learning

Active learning

Reinforcement Learning

3. Supervised Learning

x → input

y → output

Classification : y is a discrete label

Binary : 2 Labels (positive/negative)

E.g: Spam vs. Non-Spam & Male vs. Female

Multi-class : More than 2 Labels (say K)

E.g: Face recognition, Document classification

Regression : y is a continuous Label

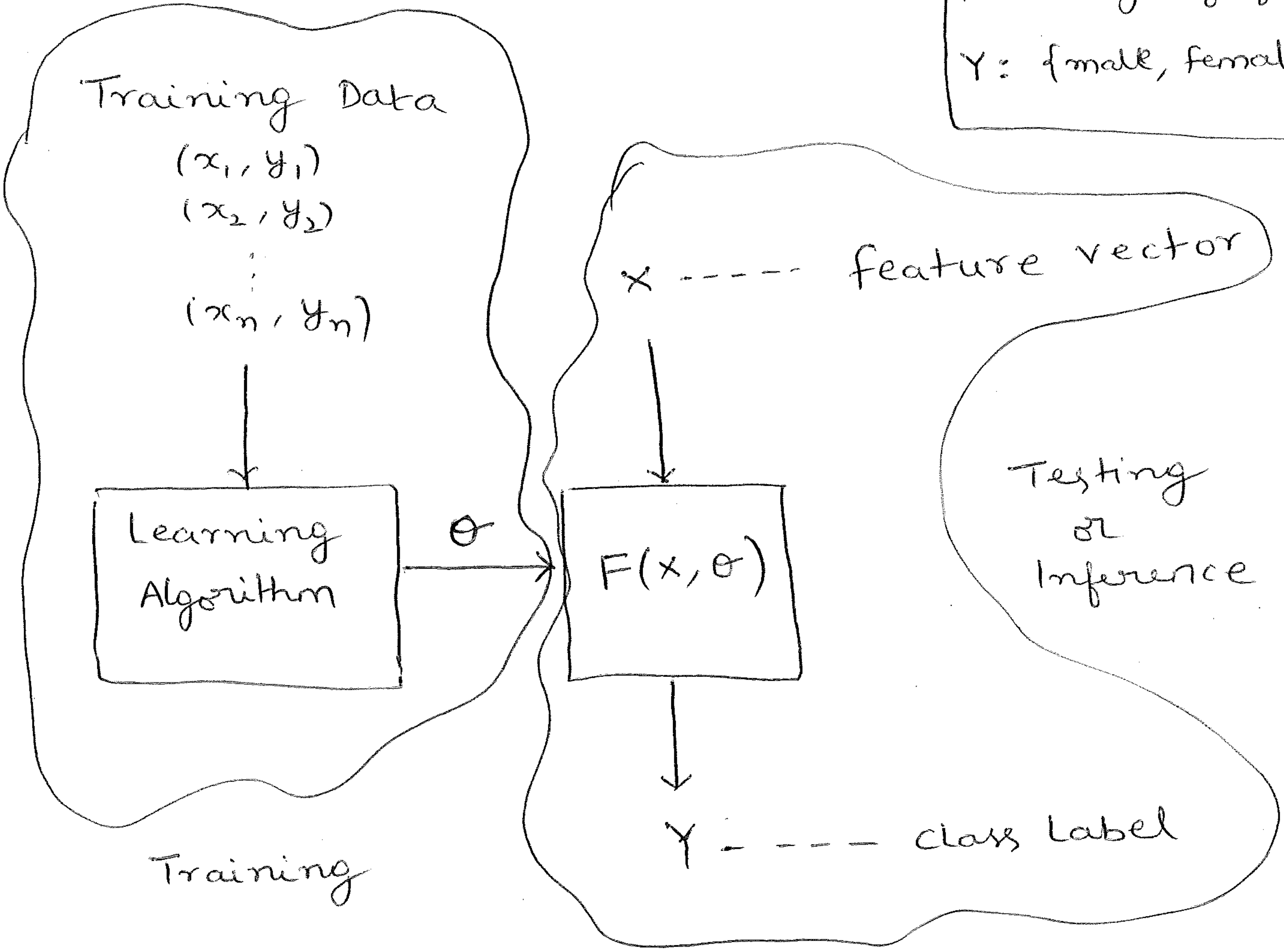
E.g: Stock market price as a functions of financial specs.

Ranking : y is an ordering of a set of objects

E.g: Search engines rank documents based on Keywords

4. Learning a Classifier

Example problem
x : image of face
Y : {male, female}



We will study algorithms :

- Perceptron
- K-nearest neighbor
- Support Vector Machines
- Decision Trees

5. Semi-Supervised Learning

Small amount of Labeled data and large amount of unlabeled data

- find a classifier that separates labeled points & unlabelede points "well"

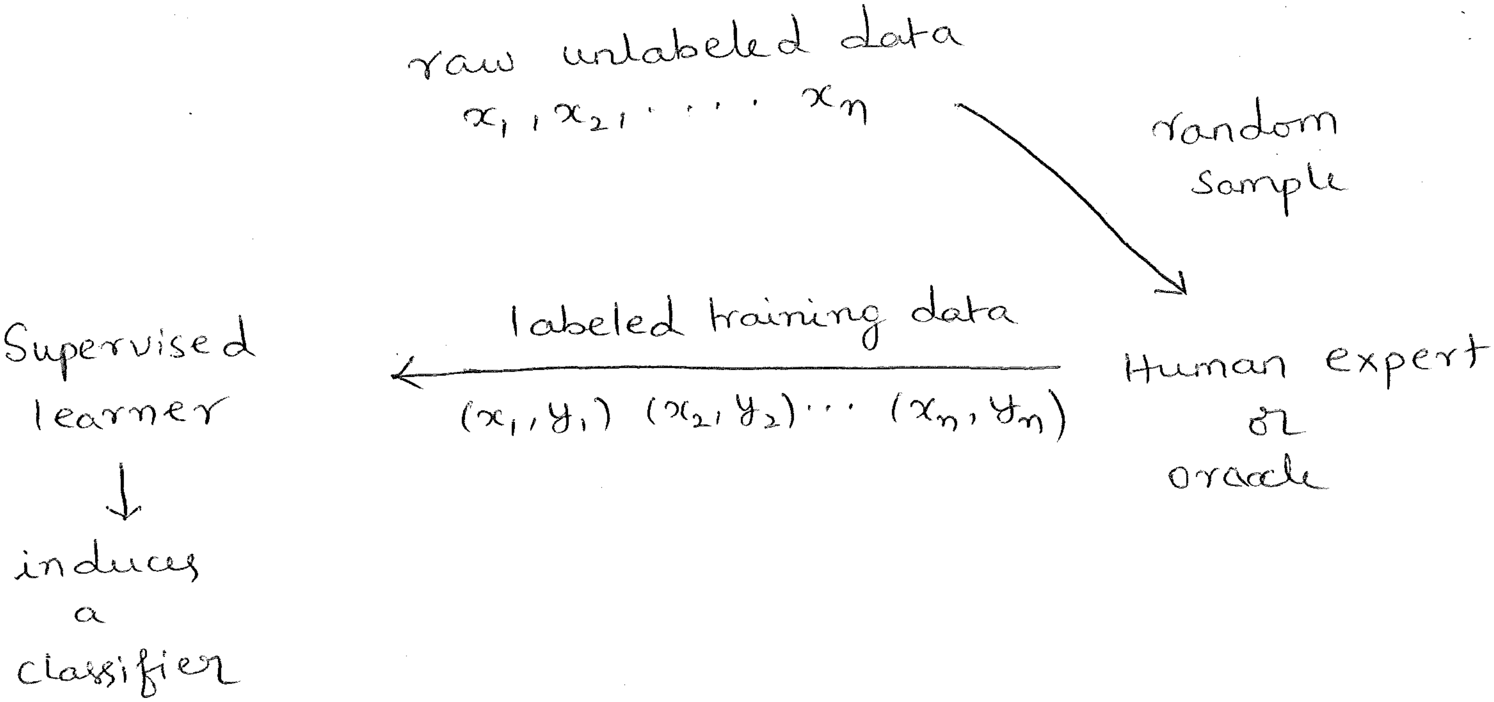
- Co-traing: leverage diversity in learners to learn from each other

Eg: Diversity: multiple redundant views

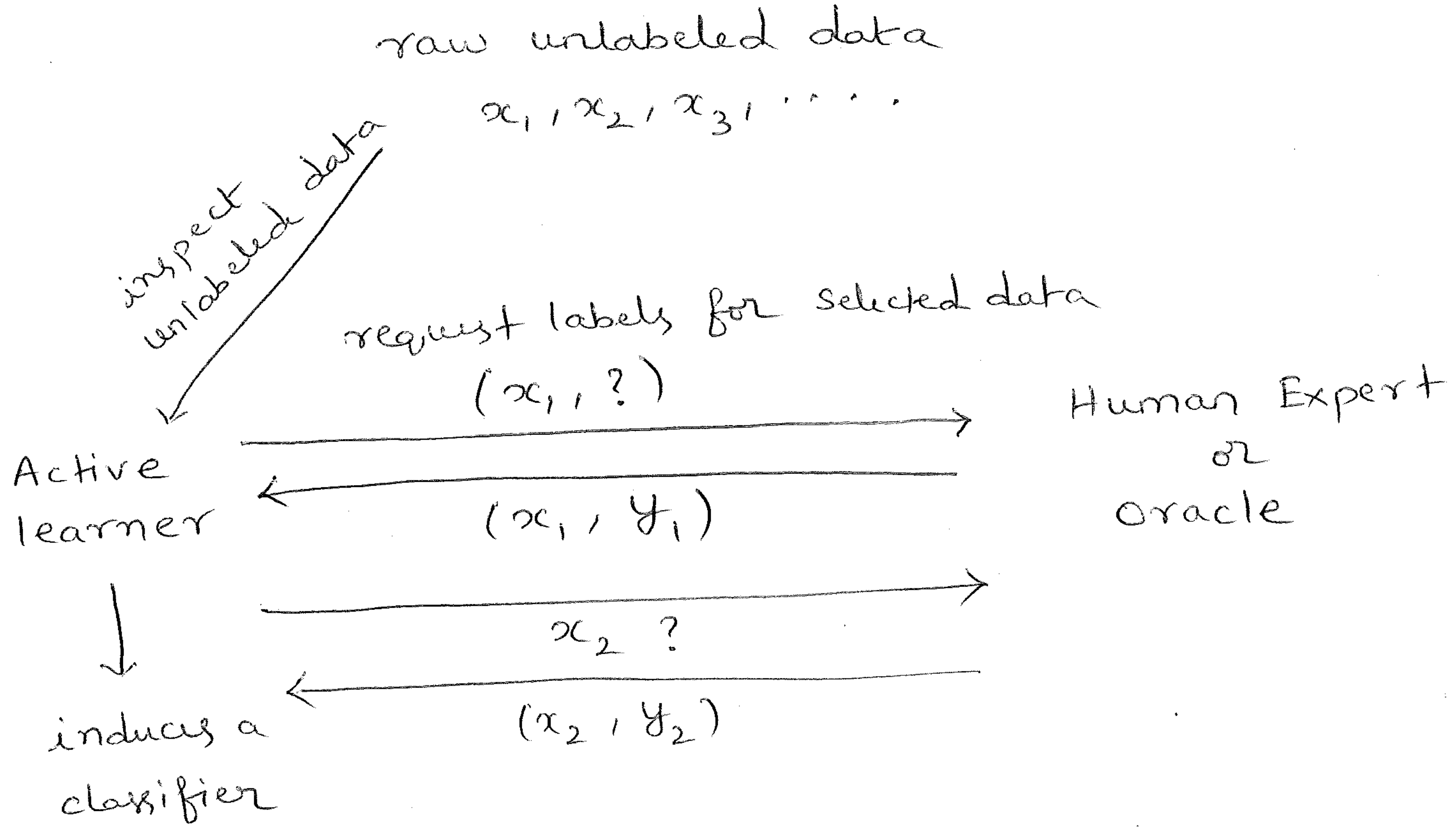
webpage classification: 1) words
2) links that point to the page.

6. Active learning

(passive) Supervised learning



Active learning



Why ??

- Labeling is expensive
- Want to learn a highly-accurate classifier with few labeled examples
- Intelligently select the examples to get labels

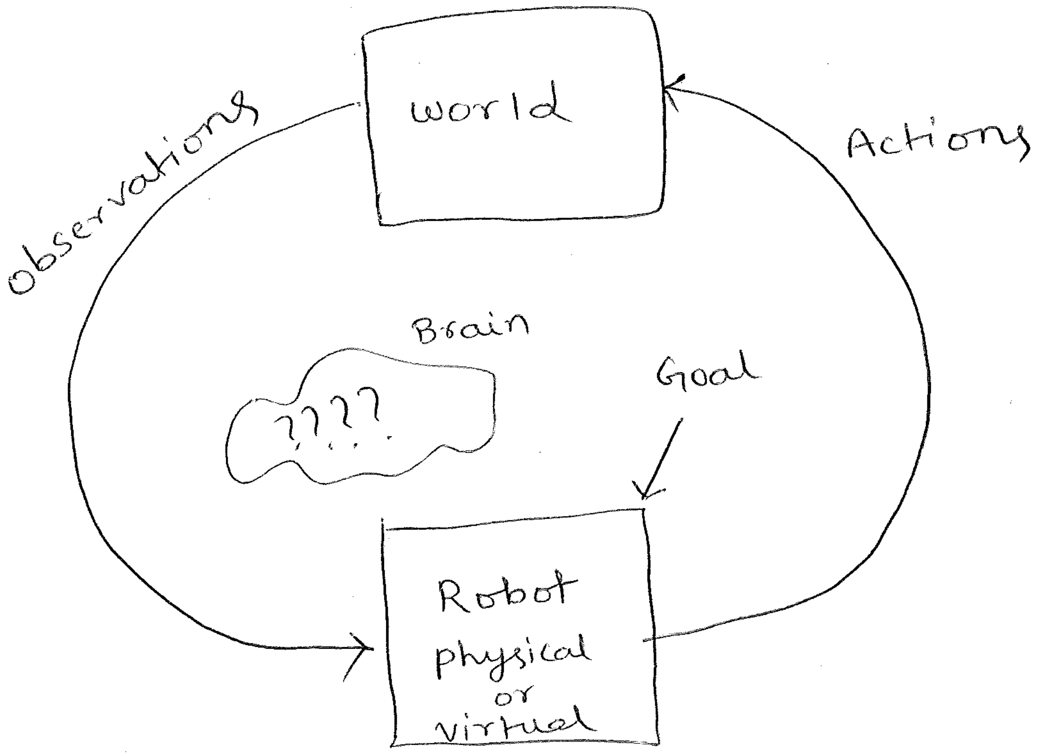
Advange : Exponential efficiency

$O(n)$ examples for supervised passive learning

$O(\log n)$ " " Active learning

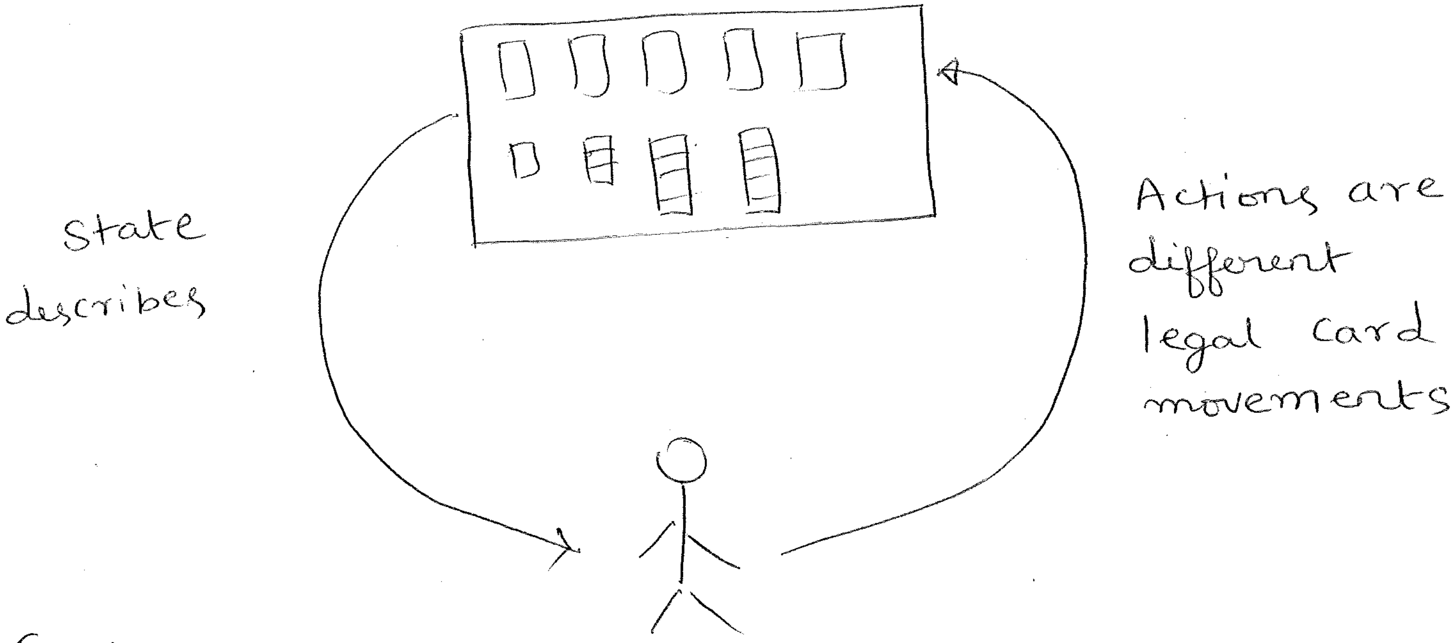
↳ Learn accurate classifier

7. Reinforcement Learning



<u>Goal</u>: maximize expected reward over lifetime

<u>Example</u>: card game



Goal: win the game or play max # of cards

Alpha~~g~~ GO : Deep learning + Monte-Carlo Tree Search

AlphaZero : No search needed

Note : Supervised learning is often used in the inner-loop of different learning paradigms.

8. Input Examples : Representation

Input examples (emails , text documents, images) are often represented as real-valued vectors

$x \in R^d$

- each co-ordinate of "x" is called a feature.

Examples :

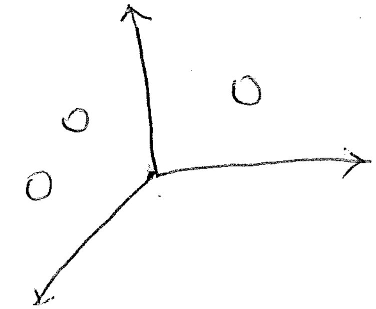
Bag of words representation of text
Histograms of colors in image
Sound frequency histogram

Bag-of-words model :

- Sentences to points

1. To be, or not to be,
2. To be a woman,
3. To not be a man

To	be	or	not	woman	a	man
2	2	1	1	0	0	0
1	1	0	0	1	1	0
1	1	0	1	0	1	1



9. Every machine learning algorithm has three components

1. Representation

2. Evaluation

3. Optimization

<u>Rep Examples:</u>

Linear hyperplanes
Decision trees
Neural networks.

$x \in R^d$

$w \in R^d$

$\boxed{\text{Sign } (w \cdot x)}$ classifier

<u>Evaluation examples:</u>

Accuracy
Precision and Recall
Likelihood
Entropy

<u>Optimization examples:</u>

Combinatorial: greedy Search
dynamic Programming

Convex optimization: gradient descent

Constrained " : linear programming
Quadratic programming

Supervised learning :

Given : a set of training examples $\{(x, y^*)\}$ drawn from an unknown target distribution D.

Find : a function F that maps input (x) to output (y) such that predictions have ~~accur~~ high accuracy on unseen inputs from D.

Learning goal : Generalization

(not memorization)

Two types of learning algorithms :

1. Online learning

   - processes one training example at a time incrementally. $(f_t \rightarrow f_{t+1})$
   - Game between teacher & Student

2. Batch learning

   - processes all training examples at once and produces a globally optimized function F

Formal Setting: Classification

1. Instances input x

   e.g: Emails

2. Output Labels $y \in \{+1, -1\}$

   e.g: Spam vs. Non-Spam

3. Prediction rule $f(x) = \hat{y}$

   e.g: linear prediction rule

4. Loss $l(y^*, \hat{y}) \in [0, \infty)$

   e.g: zero-one error

Linear Classifier:

$$\hat{y} = \text{Sign}(f(x))$$

$$= \text{Sign}(\underset{\substack{\swarrow \\ \text{weights} \\ \in R^d}}{w} \cdot \underset{\substack{\searrow \\ \text{features} \\ \in R^d}}{x})$$

$$\text{Confidence} = |f(x)| = |w \cdot x|$$

Perceptron Algorithm:

Simple and most popular ML algorithm

Online learning algorithm

Online learning Framework:

Initialize classifier $f_1(x)$

Algorithm works in rounds

on round "t", the online algorithm:

- Receives input x
- outputs a prediction $\hat{y} = f_t(x)$
- Receives a feedback Label y^*
- Computes Error $l(\hat{y}, y^*)$
- if loss > 0, update rule

$$\boxed{f_t \rightarrow f_{t+1}}$$

Learning goal : Suffer small Cumulative loss/Error

$$\sum_{t=1}^{T_{max}} l(\hat{y}_t, y^*)$$

Why online learning ?

- Fast
- Memory efficeent : process one example at a time
- Simple to implement : less than 30 mins ☺
- Online to Batch Conversions
- Adaptive

Design Principle :

If the learner suffers non-zero loss at any round, then we want to balance two goals :

1. Corrective : update function so that we don't make this error again
2. Conservative : Don't change the function too much

* Different online learning algorithms make different trade-off's between these two Competing goals.

Perceptron algorithm :

The rule to update function

$$f_t \rightarrow f_{t+1}$$

Linear Classifiers:

Find w_{t+1} from w_t based on the training example (x, y^*)

Algorithm:

If no mistake : $y^*(w_t \cdot x) > 0$

then DO nothing

$$\Rightarrow w_{t+1} = w_t$$

If mistake : $y^*(w_t \cdot x) \leq 0$

Update weights

$$w_{t+1} \leftarrow w_t + y^* x$$

Running Example :

Training data :	x_1	y_1
	$(4, 0)$	$+1$
	x_2	y_2
	$(1, 1)$	-1
	x_3	y_3
	$(0, 1)$	-1
	x_4	y_4
	$(-2, -2)$	$+1$

$w_1 = 0$

For $t = 1$,

$$y^* \cdot (w_1 \cdot x)$$

$$= 0 \quad \text{// mistake}$$

$\Rightarrow$

$$w_2 = w_1 + 1 \cdot (4, 0)$$

$$= (4, 0)$$

For $t = 2$

$$y^* (w_2 \cdot x) < 0$$

$$\Rightarrow \quad w_3 = w_2 + (-1) \cdot (1, 1)$$

$$= (4, 0) - (1, 1)$$

$$= (3, -1)$$

For $t = 3$

$$y^* \cdot (w_3 \cdot x) > 0 \qquad // \text{ correct}$$

$$\Rightarrow \quad w_4 = w_3 = (3, -1)$$

For $t = 4$

$$y^* (w_4 \cdot x) < 0$$

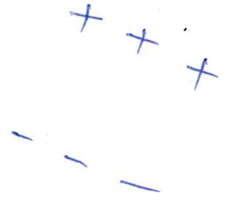
$$\Rightarrow \quad w_5 = w_4 + 1 \cdot (-2, -2)$$

$$= (3, -1) + (-2, -2)$$

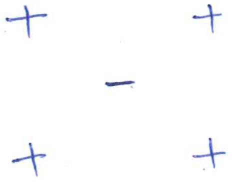
$$= (1, -3)$$

When does Perceptron Converge:

linear separability : If there exists a weight vector that can separate positive and negative points.



linearly separabde



Not linearly separable.

Measure of Separability :

Margin : For a weight vector w & training set $S = \{(x, y^*)\}$, margin of w with respect to S is defined as follows :

$$\gamma(w) = \min_{(x, y^*) \in S} y.(w \cdot x)$$

+
-
High-margin data
low-margin data

Convergence : If training set is linearly Separable with margin γ, then perceptron makes $\leq \frac{1}{\gamma^2}$ mistakes

1. lower margin implies more mistakes
2. May need more than one pass over the training data to get a classifier with no mistakes.

What if data is not linearily Separable ?

Perceptron still works

- there will be few mistakes close to the decision boundary
- will never converge to a single "w" as we make more passes.

Voted Perceptron :

Initialization : $m = 1$; $w_1 = 0$; $c_m = 1$

Training examples : for $t = 1, 2, 3, \ldots$

```
△ If mistake, update weights
        $w_{m+1} = w_m + y^*_t x_t$
        $m = m + 1$
        $c_m = 1$
△ Else
        $c_m = c_m + 1$      // Counting how long $w_m$ survived
```

Output: $(w_1, c_1), (w_2, c_2), (w_3, c_3) \ldots$

Voted Perceptron classifier

weighted majority vote of all the classifiers.

Drawbacks:

1. we have to store many classifiers (space)
2. we need to make many predictions (time)

Averaged Perceptron:

$$W = \frac{\sum_{i=1}^{K} c_i * w_i}{K}$$

Averaging ⇒ robustness & regularization (leading to better generalization)

Some Practical tricks:

1. Shuffling: shuffle the training examples in each iteration
2. Variable learning rate: decrease as learning progresses

Learning Curve :

- Training iterations vs. no of mistakes
- You want to see that mistakes decrease as we increase the no of iterations.
- very useful in debugging & seeing the learning behaviour.

Hyper-parameter Optimization :

- Split the training data : Sub-train
  +
  validation
- Tune hyper-parameters (e.g., no of iterations) on the validation data
- The learner should not look at the test data.

Multi-class classification:

Suppose we have $(K > 2)$ classes.

K weight vectors: $w_1, w_2, \ldots, w_K \in R^d$

input instance $x \in R^d$

$\text{score}(\text{label } r) = w_r \cdot x$

Class r	$w_r \cdot x$
1	-1.08
2	1.66
3	0.37
4	-2.09

Prediction: output label (class) with highest score.

Learning:

$$w_{y^*} = w_{y^*} + x$$

$$w_{\hat{y}} = w_{\hat{y}} - x$$

Regression Learning:

y is Continuous value.

Prediction Rule : $F(x) = w \cdot x$

Widrow - Hoff Algorithm :

```
- Initialize $w_1 = 0$
for $t = 1$ to $T$ do
    - get $x_t \in R^d$
    - predict $\hat{y}_t = w_t \cdot x_t$
    - observe $y^*_t$
    - Incur loss of $(\hat{y}_t - y^*_t)^2$
    - $w_{t+1} = w_t - \eta (w_t \cdot x_t - y^*_t) * x_t$
end
```