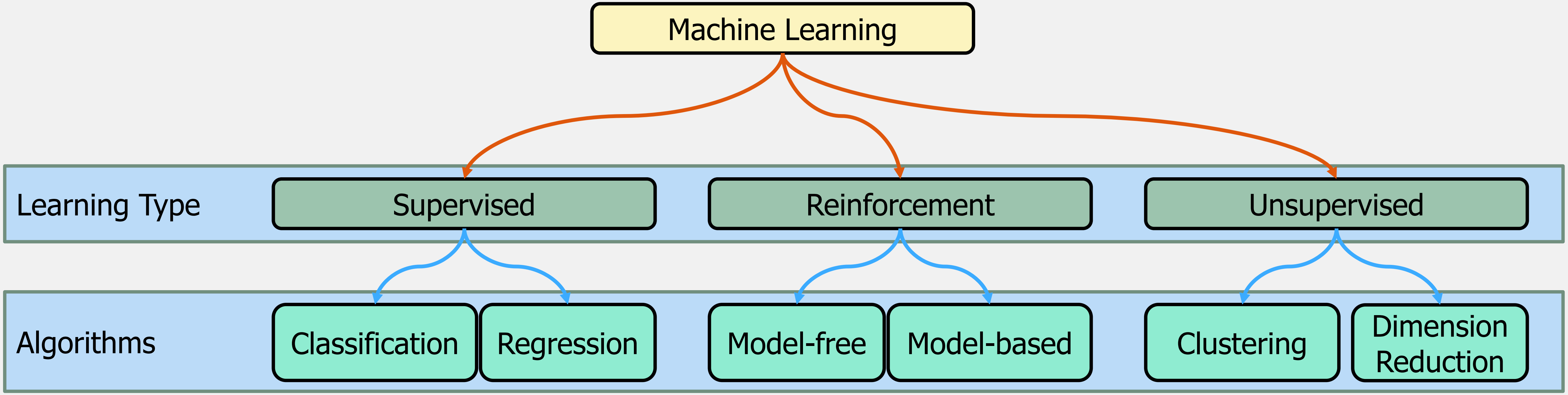
Model Learning Strategy

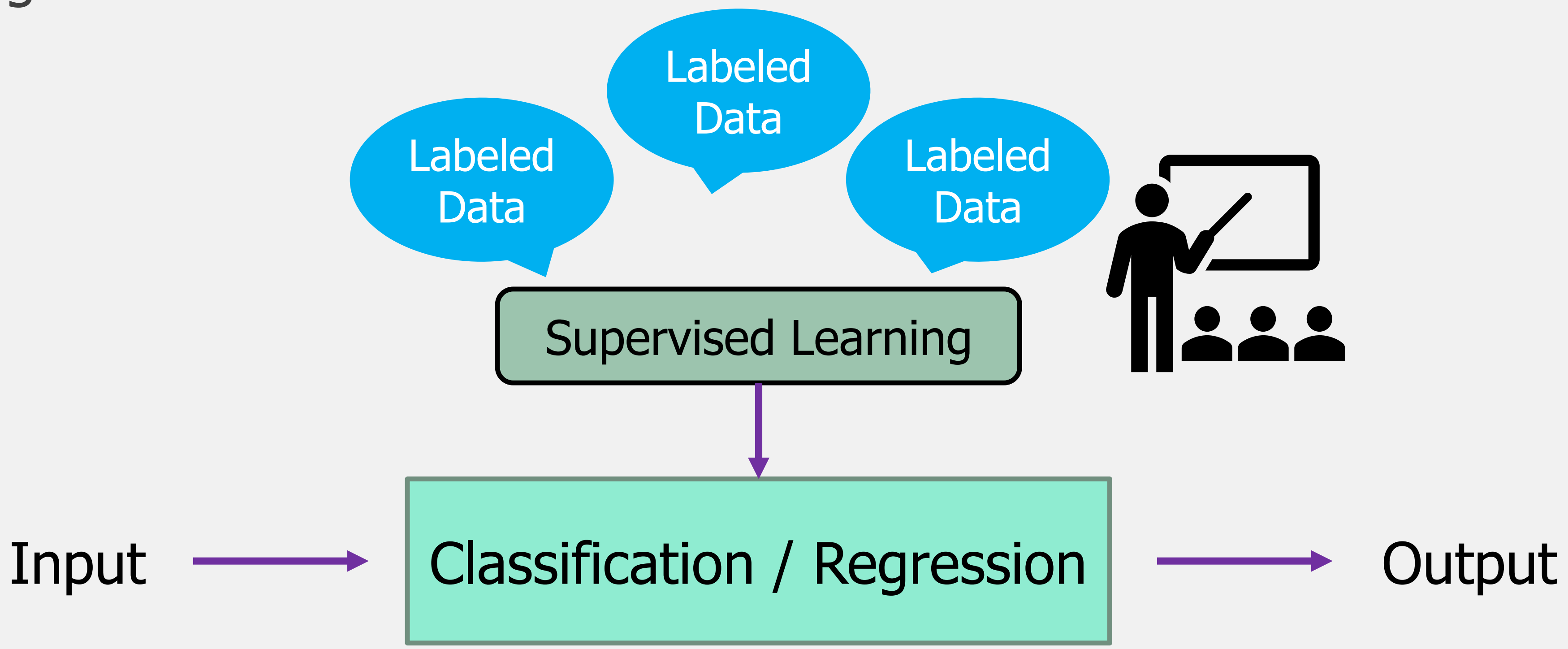
Strategy


Machine Learning
Learning Type
Supervised
Reinforcement
Unsupervised
Algorithms
Classification
Regression
Model-free
Model-based
Clustering
Dimension Reduction

Strategy

Supervised Learning

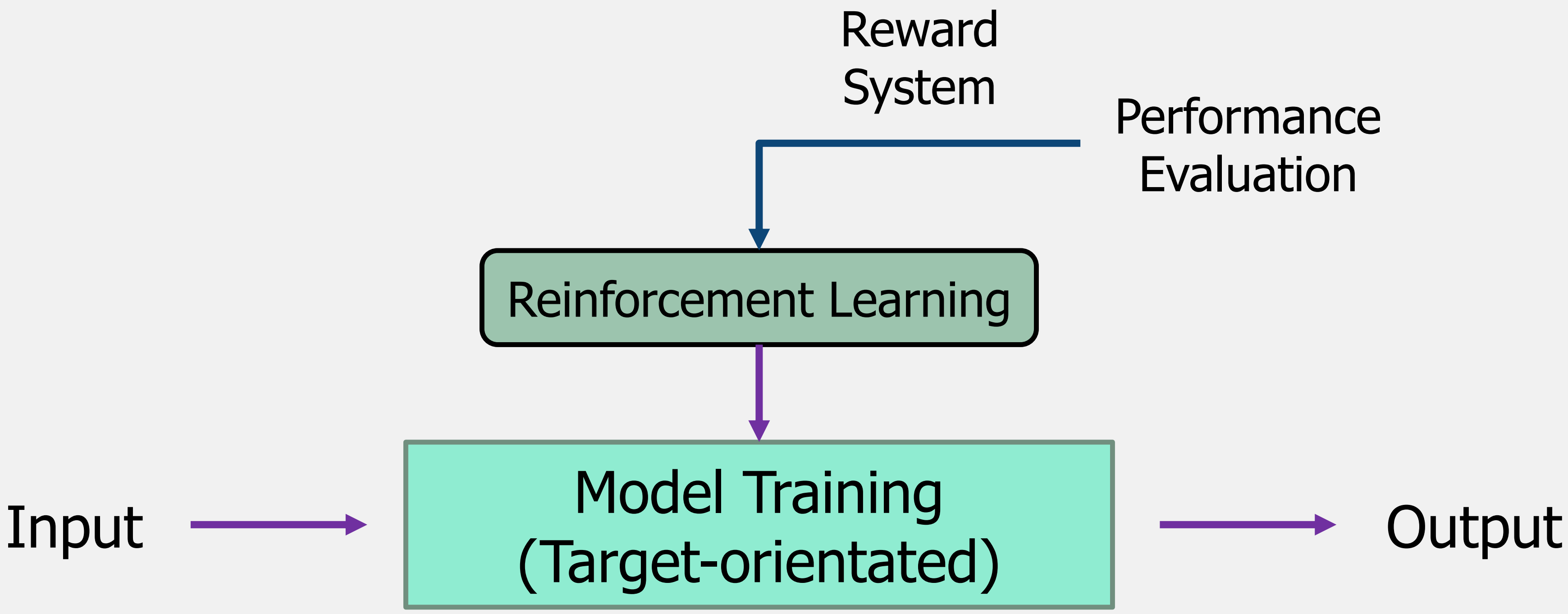
- Using labeled data to train model
- Needing human to label data



Strategy

Reinforcement Learning

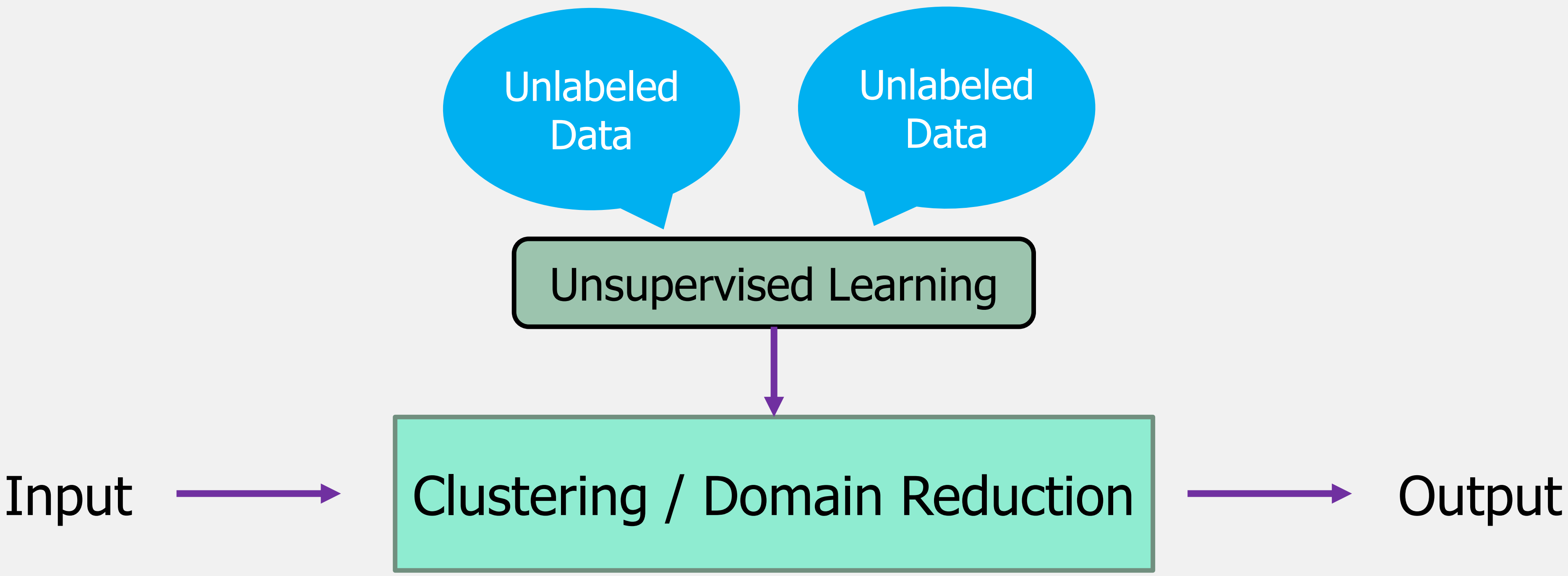
- ➢ Learning by interaction
- ➢ Needing reward to act



Strategy

Unsupervised Learning

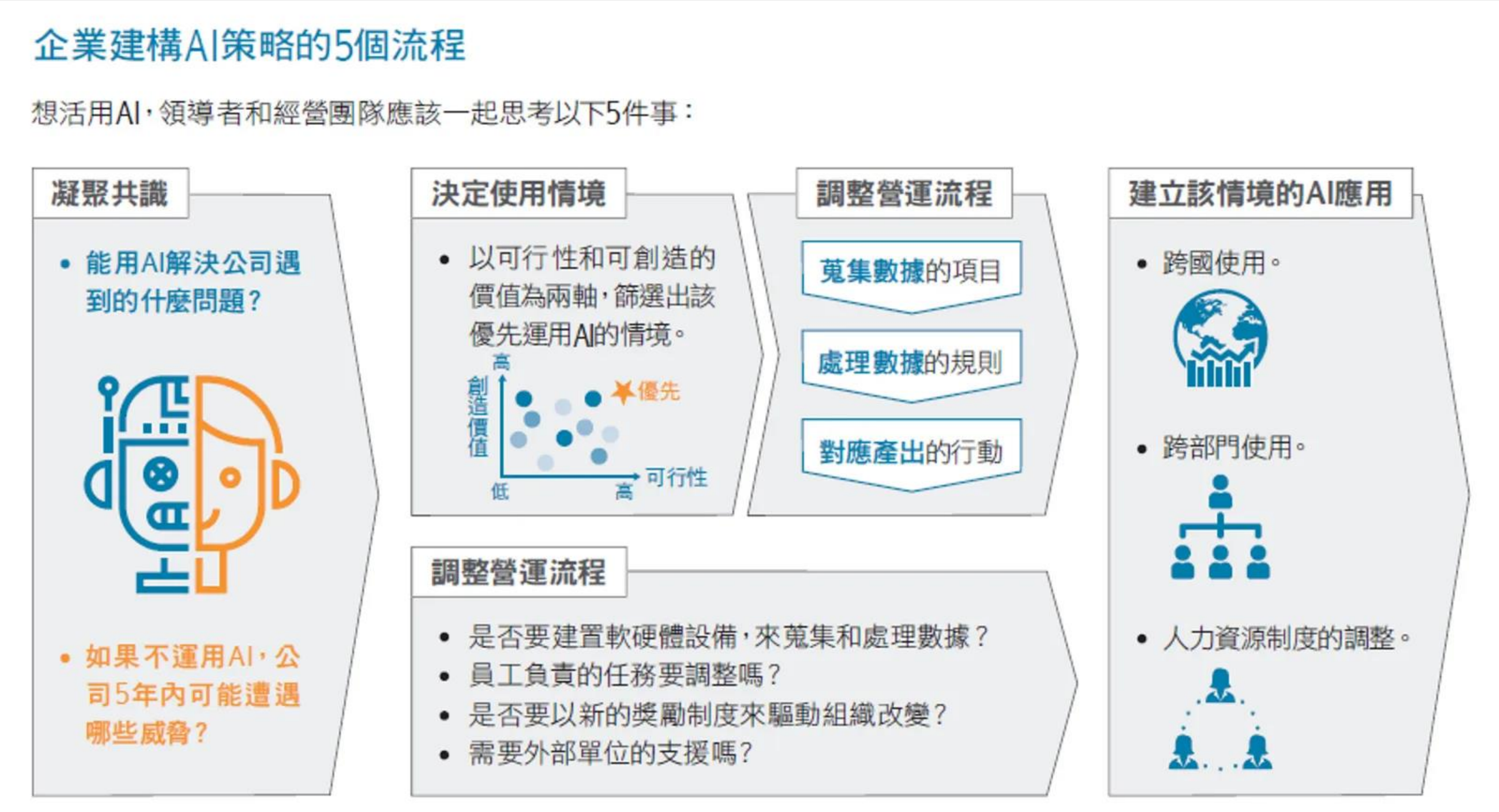
- Clustering by association
- No needing human to label data



Strategy

How to decide strategy?

- Depend on the problems you want to solve

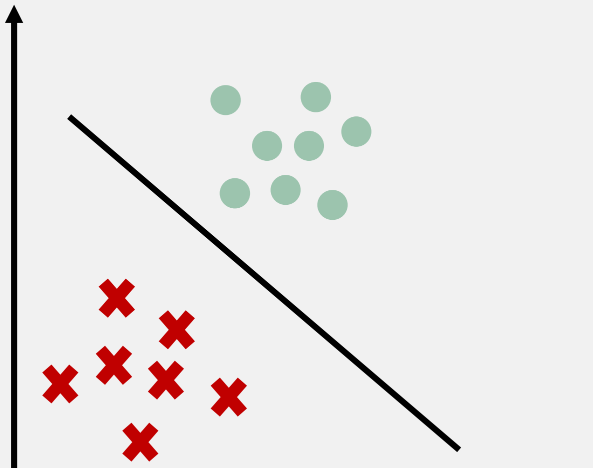


Source

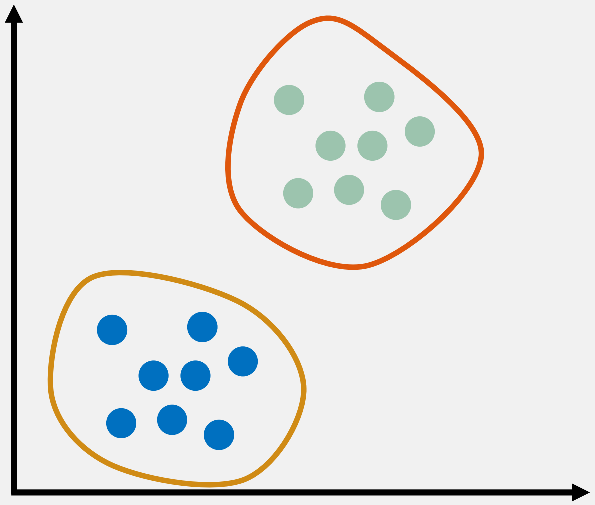
Strategy

Model

- ➢ Supervised
  - Linear regression, k-nearest neighbor, support vector machine, decision tree, etc.
- ➢ Unsupervised
  - K-means, principle component analysis, etc.



Supervised



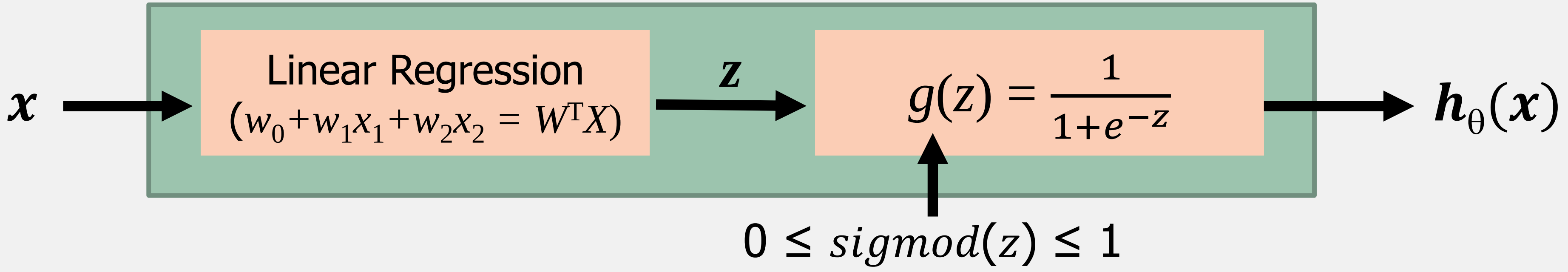
Unsupervised

01

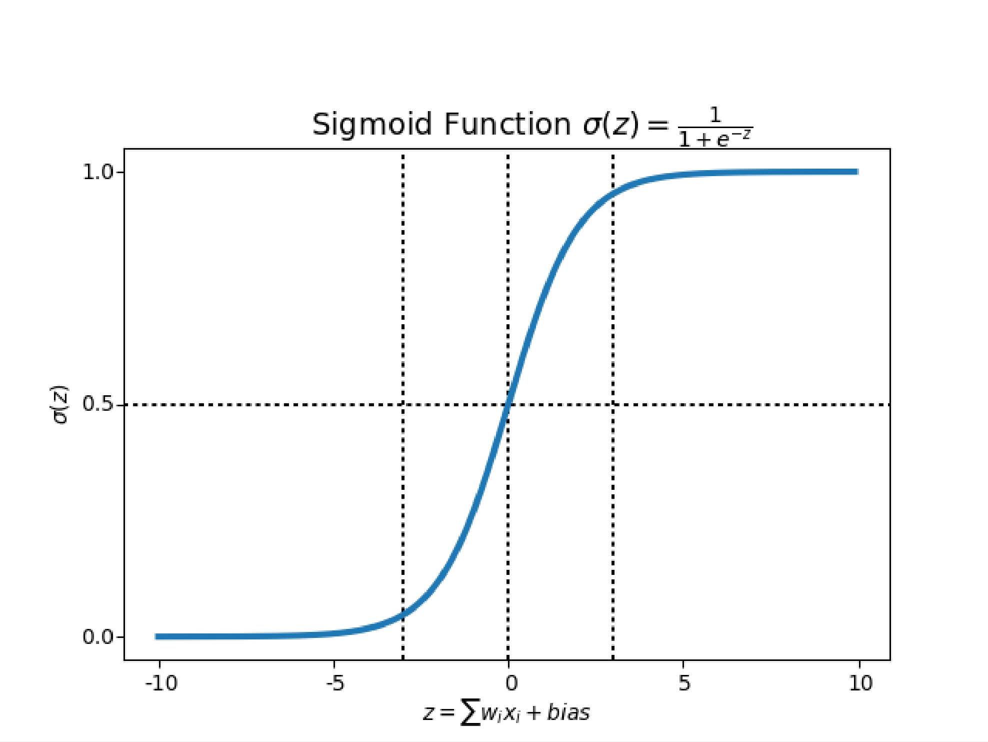
Supervised Learning

Supervised

Logistic Regression

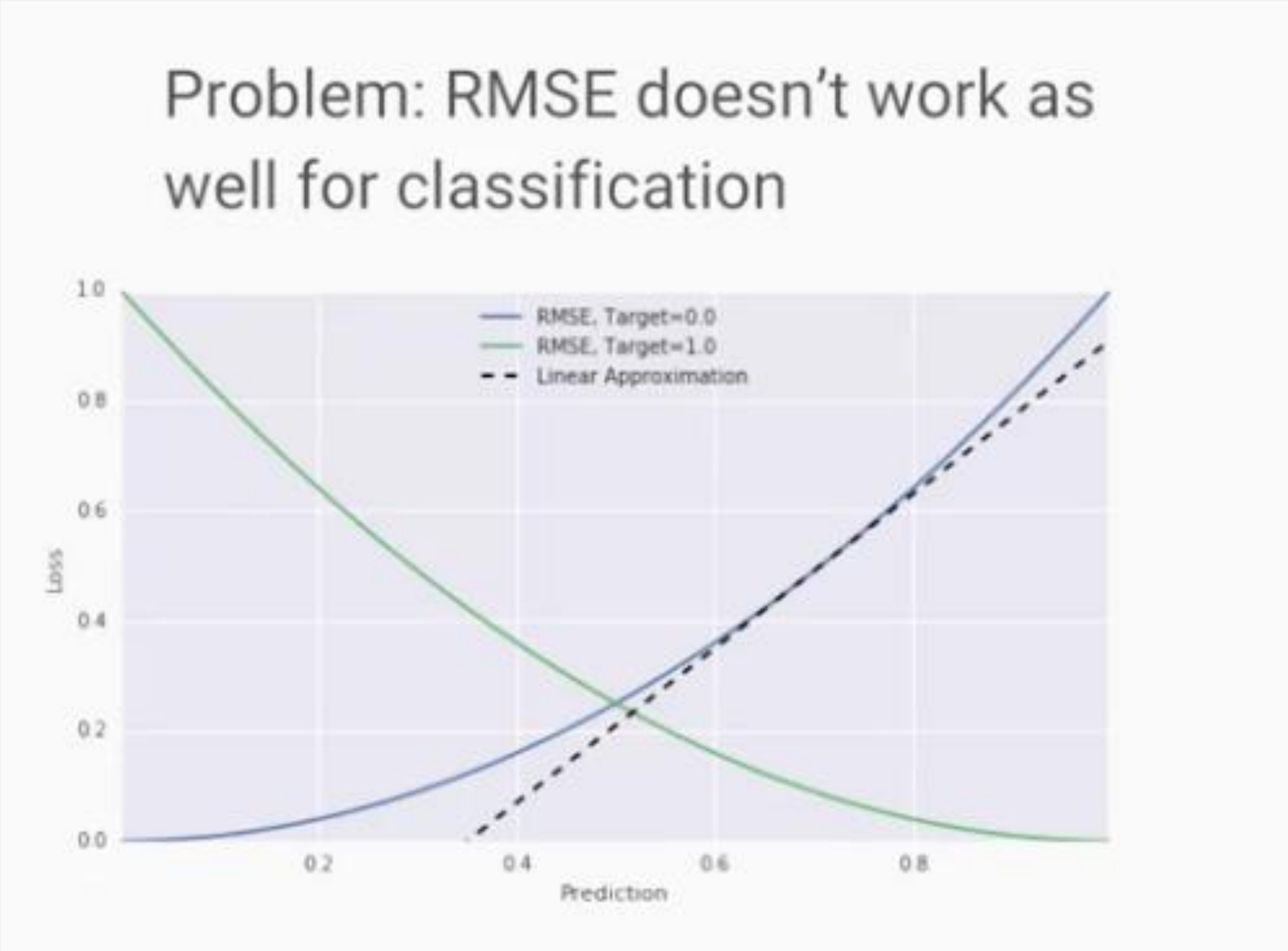


If > threshold output 1;
else output 0



Supervised
Logistic Regression

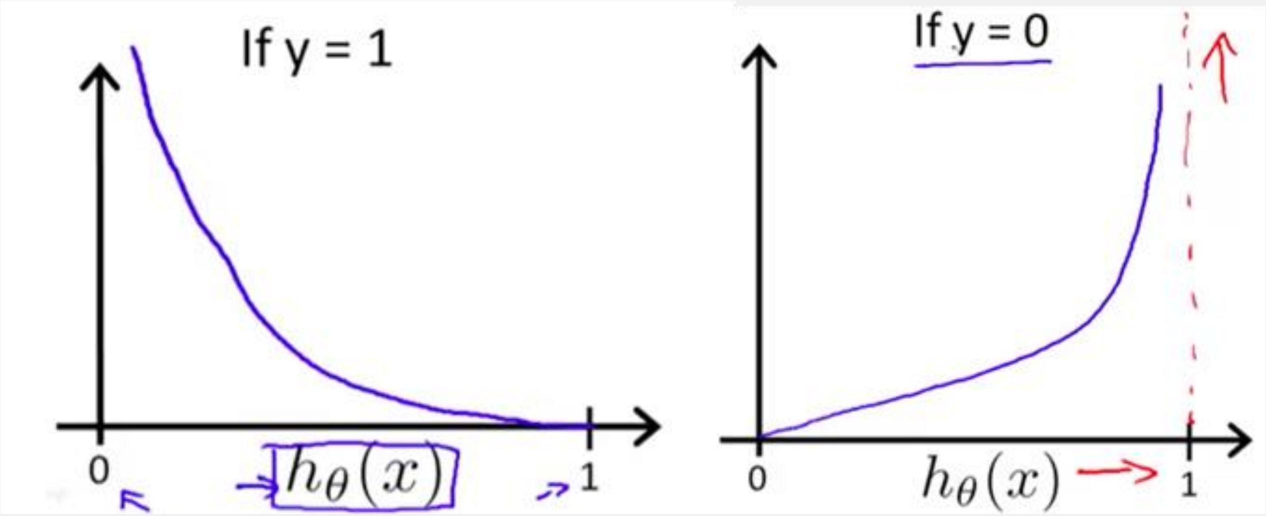
Linear regression: $J(\theta) = \sqrt{\frac{1}{m}\sum_i^m (h_\theta(x^{(i)}) - y^{(i)})^2}$



Supervised

Logistic Regression

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



If $h_\theta(x) = y$, then $\mathrm{cost}(h_\theta(x), y) = 0$ (for $y = 0$ and $y = 1$).

If $y = 0$, then $\mathrm{cost}(h_\theta(x), y) \to \infty$ as $h_\theta(x) \to 1$.

If $y = 0$, then $\mathrm{cost}(h_\theta(x), y) \to 0$. as $h_\theta(x) \to 0$.

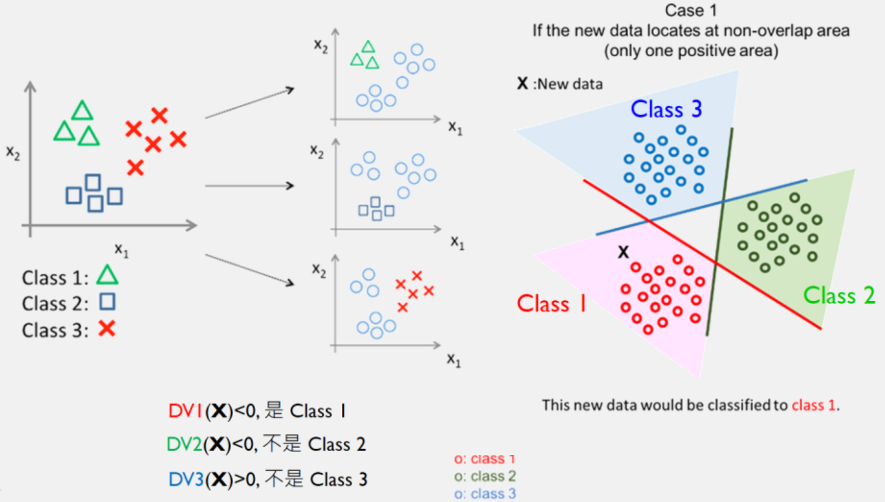
Regardless of whether $y = 0$ or $y = 1$, if $h_\theta(x) = 0.5$, then $\mathrm{cost}(h_\theta(x), y) > 0$.

Supervised
Logistic Regression

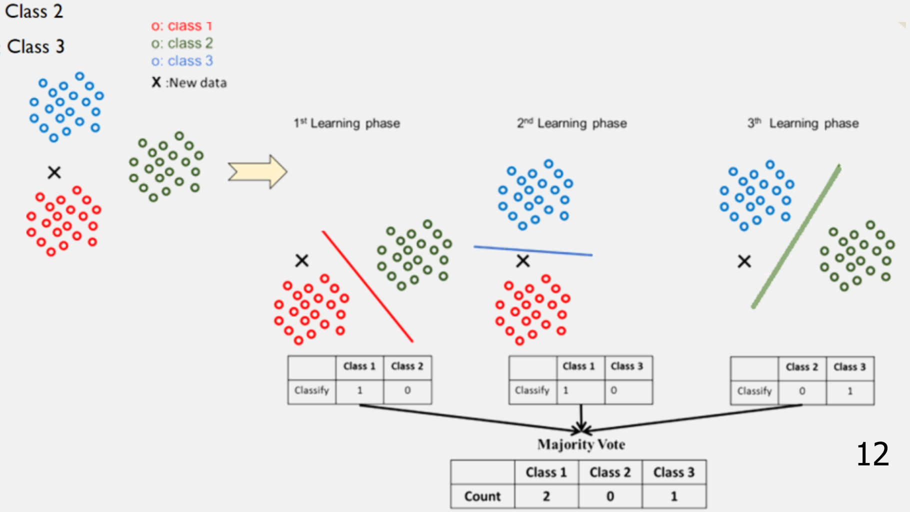
Multiclass problem

➢ When the problem is multi-class problem, there are generally 2 algorithms

One versus rest
The algorithm compares every class with all the remaining classes, building a model for every class
If you have n classes to guess, you have n models



One versus one
The algorithm compares every class against every -individual remaining class, building a number of models equivalent to n * (n-1) / 2, where n is the number of classes



	Class 1	Class 2
Classify	1	0

	Class 1	Class 3
Classify	1	0

	Class 2	Class 3
Classify	0	1

	Class 1	Class 2	Class 3
Count	2	0	1

Supervised

K-nearest Neighbor

K-nearest Neighbor Classification (KNN)

- KNN does not build model from the training data
- To classify a test instance *d*, define k-neighborhood *P* as *k* nearest neighbors of *d*
- Count number n of training instances in *P* that belong to class c_j
- Estimate $\Pr(c_j|d)$ as n/k
- No training is needed
- Classification time is linear in training set size for each test case

Supervised

K-nearest Neighbor

Instance-based Learning

- Learning=storing all training instances
- Classification=assigning target function to a new instance
- Referred to as "Lazy" learning

Algorithm kNN(D, d, k)

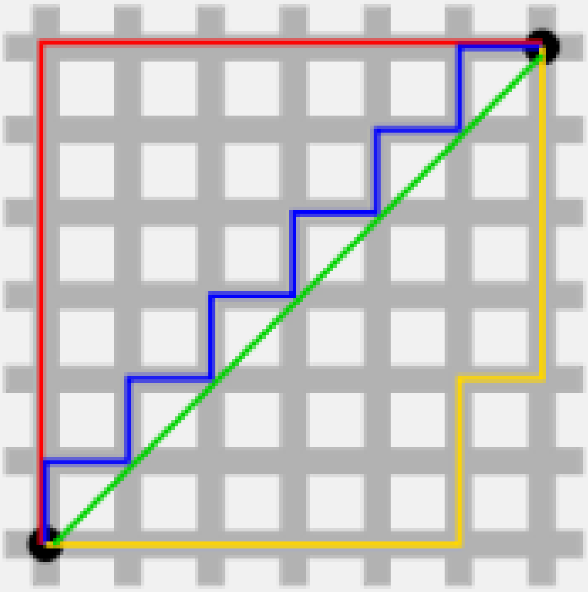
Compute the distance between d and every example in D;
Choose the k examples in D that are nearest to d, denote the set by P ($\subseteq D$);
Assign d the class that is the most frequent class in P (or the majority class);

k is usually chosen empirically via a validation set or cross-validation by trying a range of k values

Distance function is crucial, but depends on applications

Supervised

K-nearest Neighbor

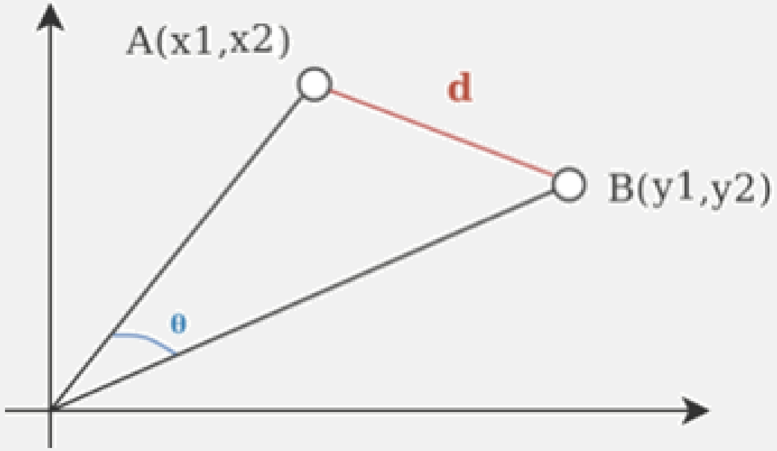


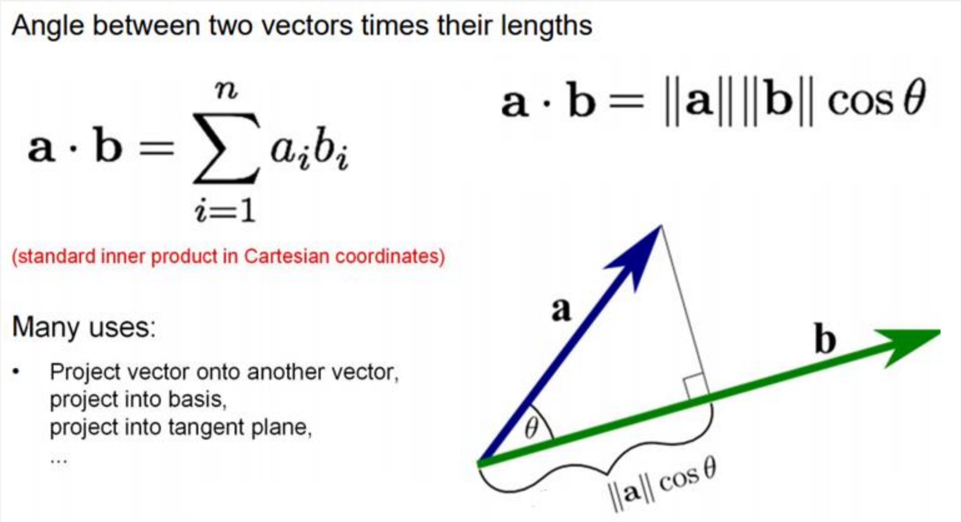
$$x = (x_1,\ x_2,\ x_3, \ldots,\ x_n) \text{ and } y = (y_1,\ y_2,\ y_3, \ldots,\ y_n)$$

1) Manhattan Distance: $d(x, y) = \sum_{i=1}^{n} |(x_i - y_i)|$

2) Euclidean Distance: $d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

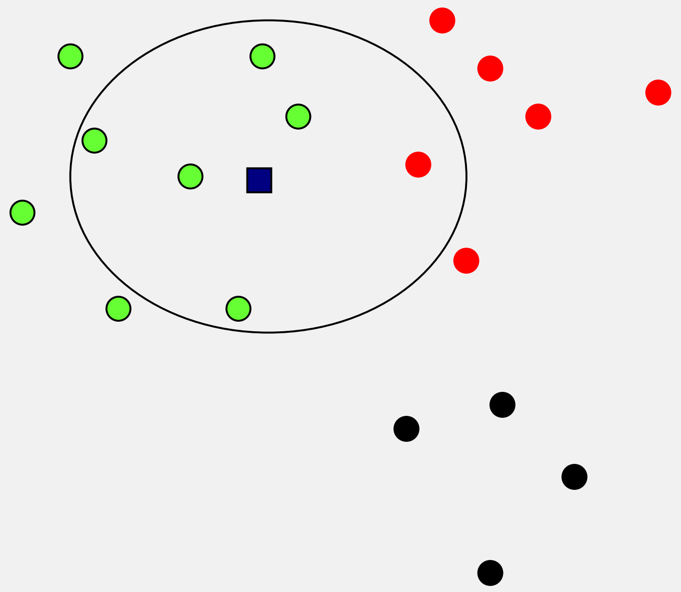
3) Cosine Distance: $\cos\theta = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \|\vec{y}\|}$





Supervised

K-nearest Neighbor



- Government
- Science
- Arts

A new point ■
Pr(science|■)?

Supervised

K-nearest Neighbor

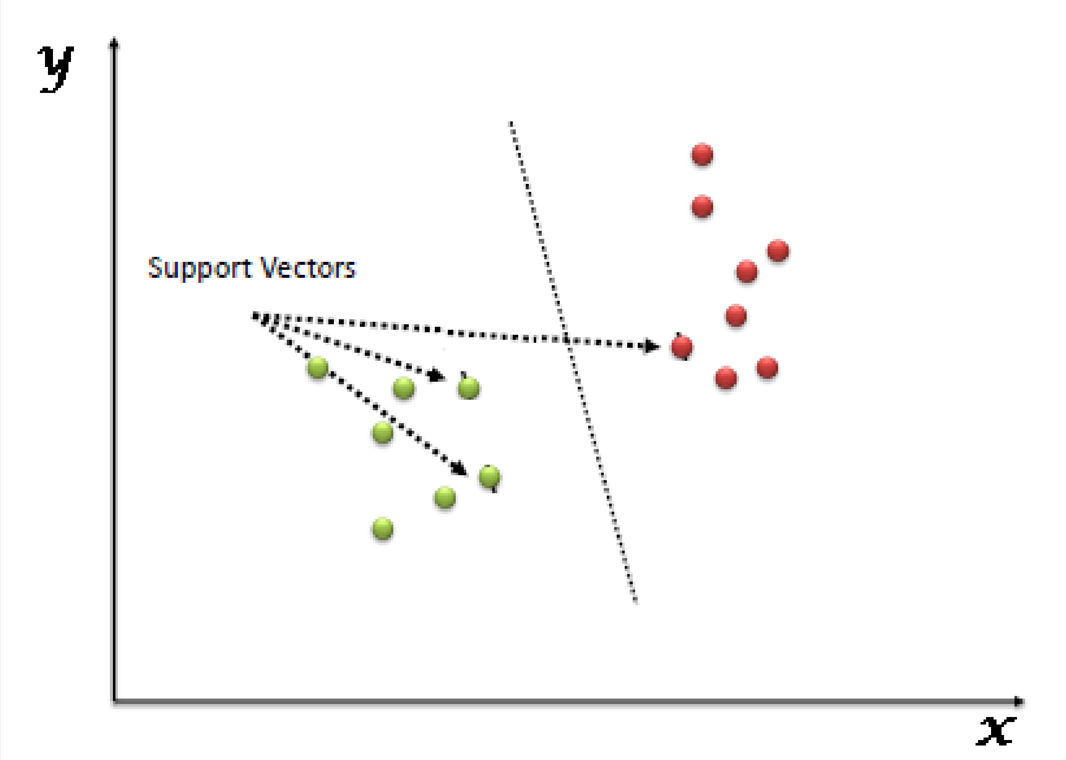
Discussions

- KNN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of KNN can be quite strong and in many cases as accurate as those elaborated methods.
- KNN is slow at the classification time
- KNN does not produce an understandable model

Supervised
Support Vector Machine

Support Vector Machine (SVM)

- It is a supervised algorithm that can be employed for both classification or regression challenges, but mostly in classification
- Support vectors are the coordinates of the individual observation



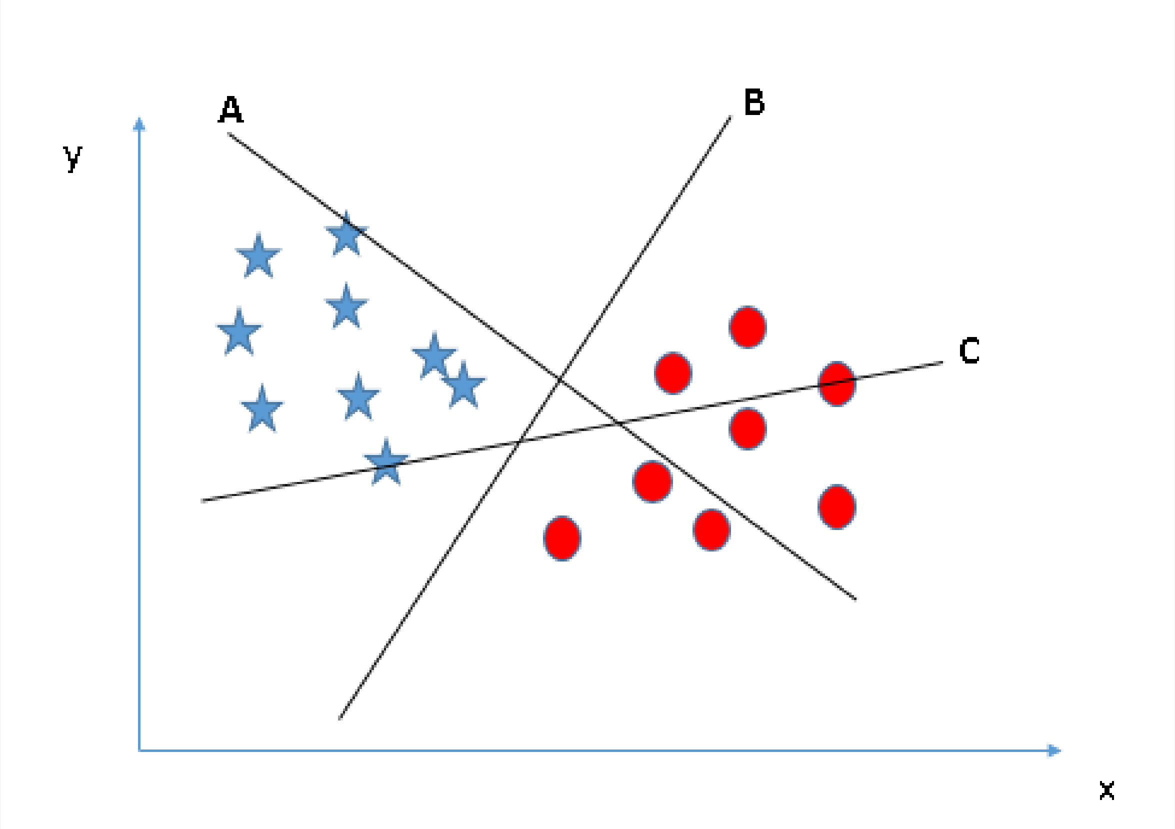
Reference

Supervised

Support Vector Machine

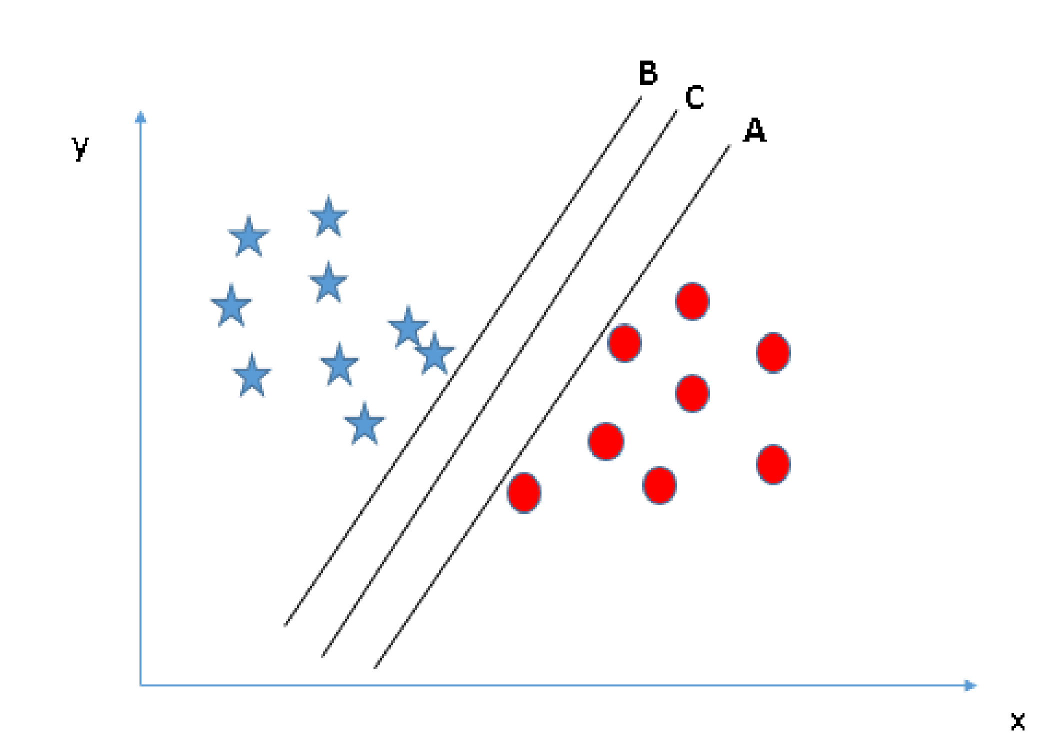
A frontier which best segregates the two classes (hyper-plane)

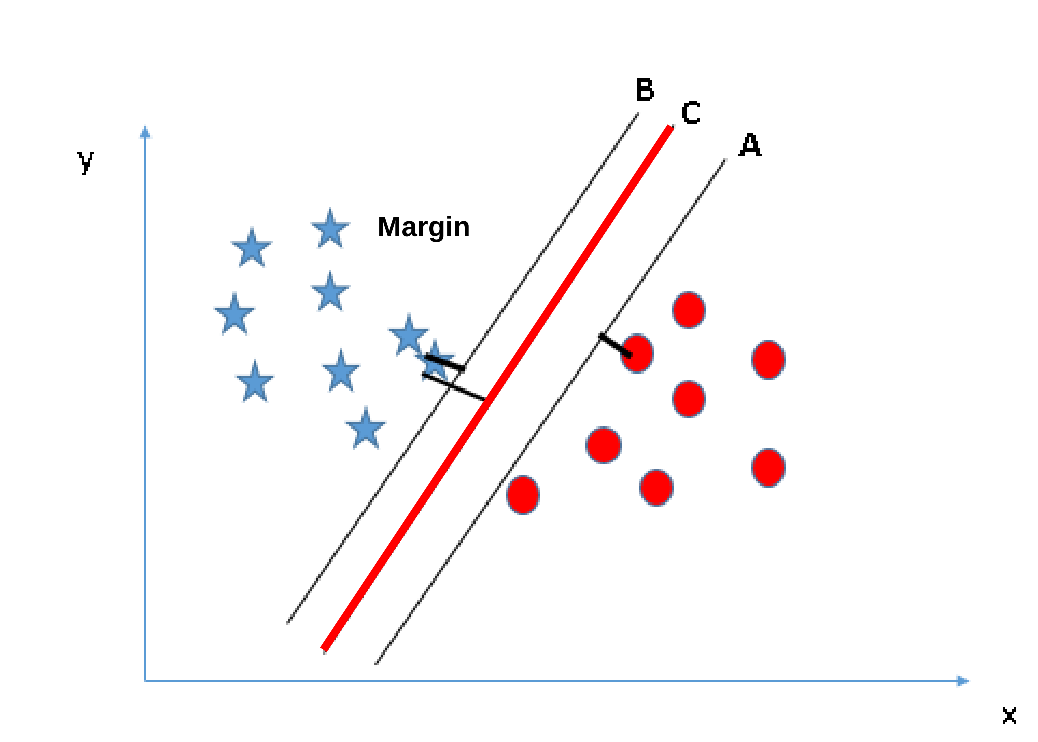
Scenario-1: Identify the right hyper-plane



Supervised
Support Vector Machine

Scenario-2: Identify the right hyper-plane?



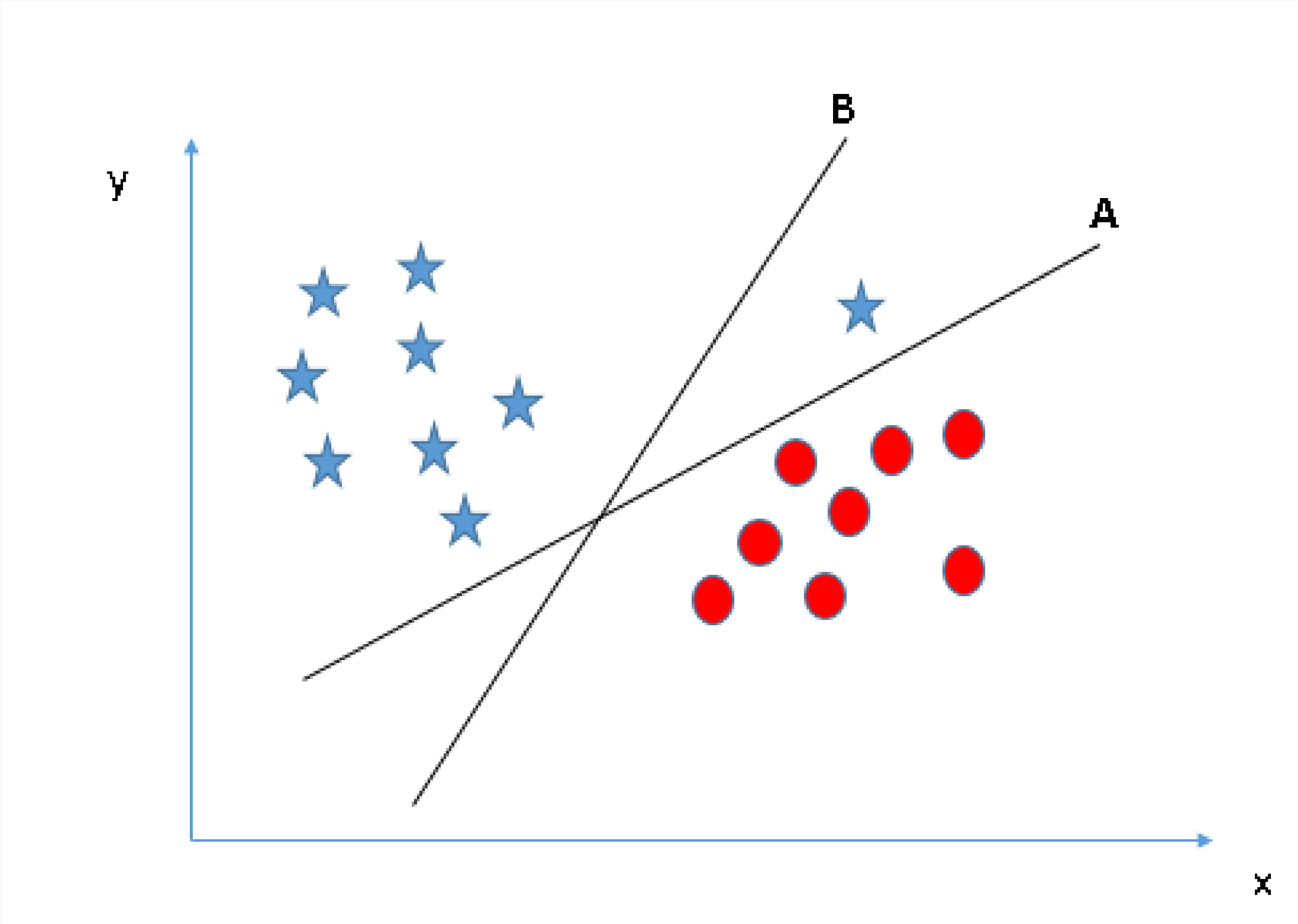


Maximizing the distances between nearest data point (either class)

Supervised

Support Vector Machine

Scenario-3: Identify the right hyper-plane?

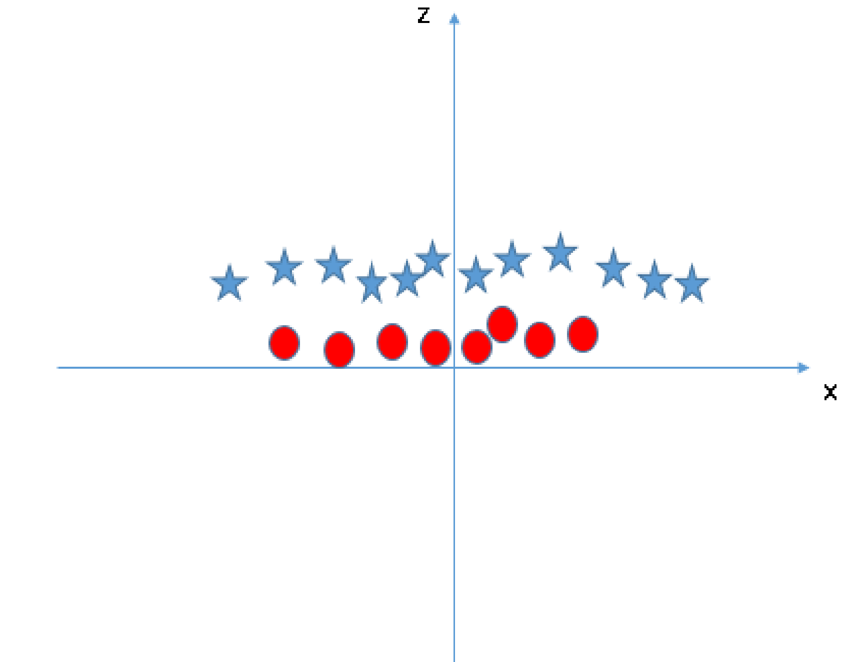


Supervised

Support Vector Machine

SVM solves this problem by introducing additional feature

Here, we will add a new feature $z=x^2+y^2$



只要對所有的資料，有一個函數可以滿足
k(x,y)=⟨φ(x),φ(y)⟩ ≥ 0
這個k(x,y)就是一個kernel函數
⟨a, b⟩表示向量a和b做內積。

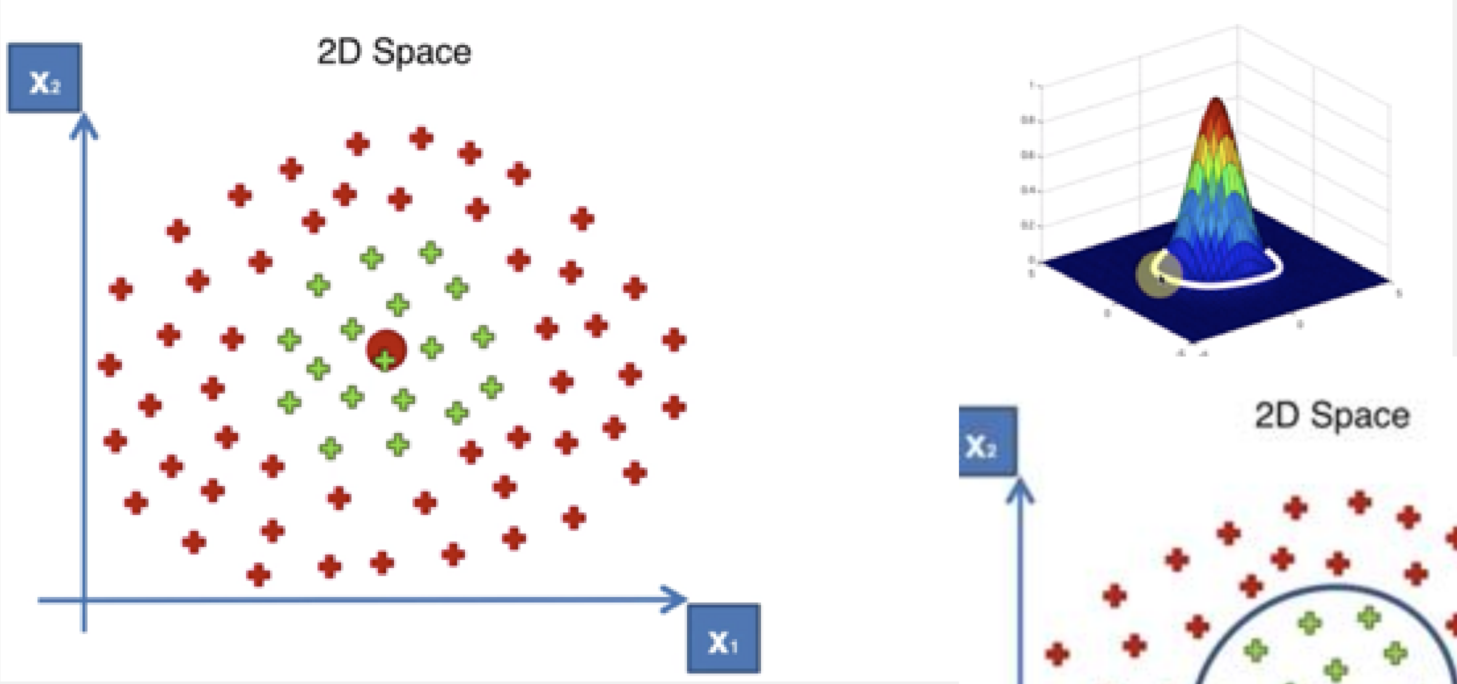
mostly useful in non-linear separation problem

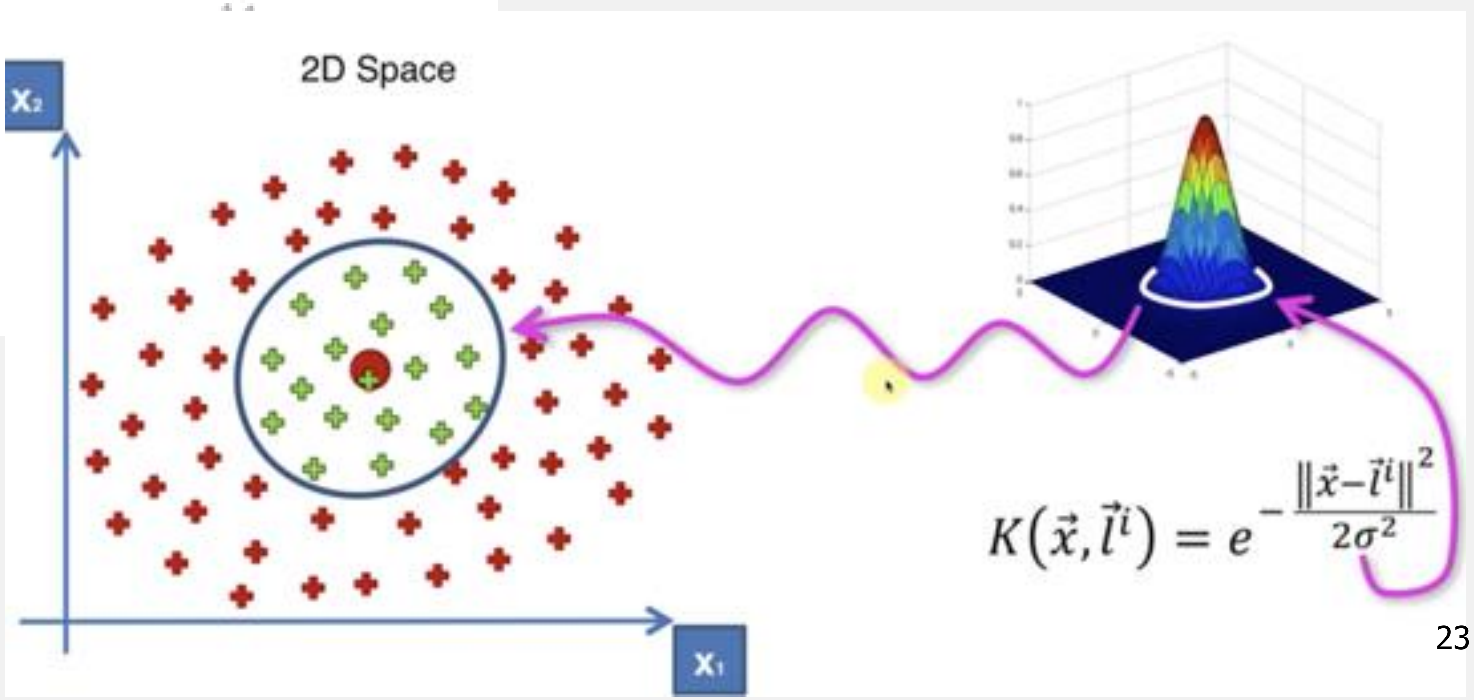
*Should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the **kernel** **trick**.

*These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called **kernels**.

Supervised
Support Vector Machine

How to tune parameters of SVM?

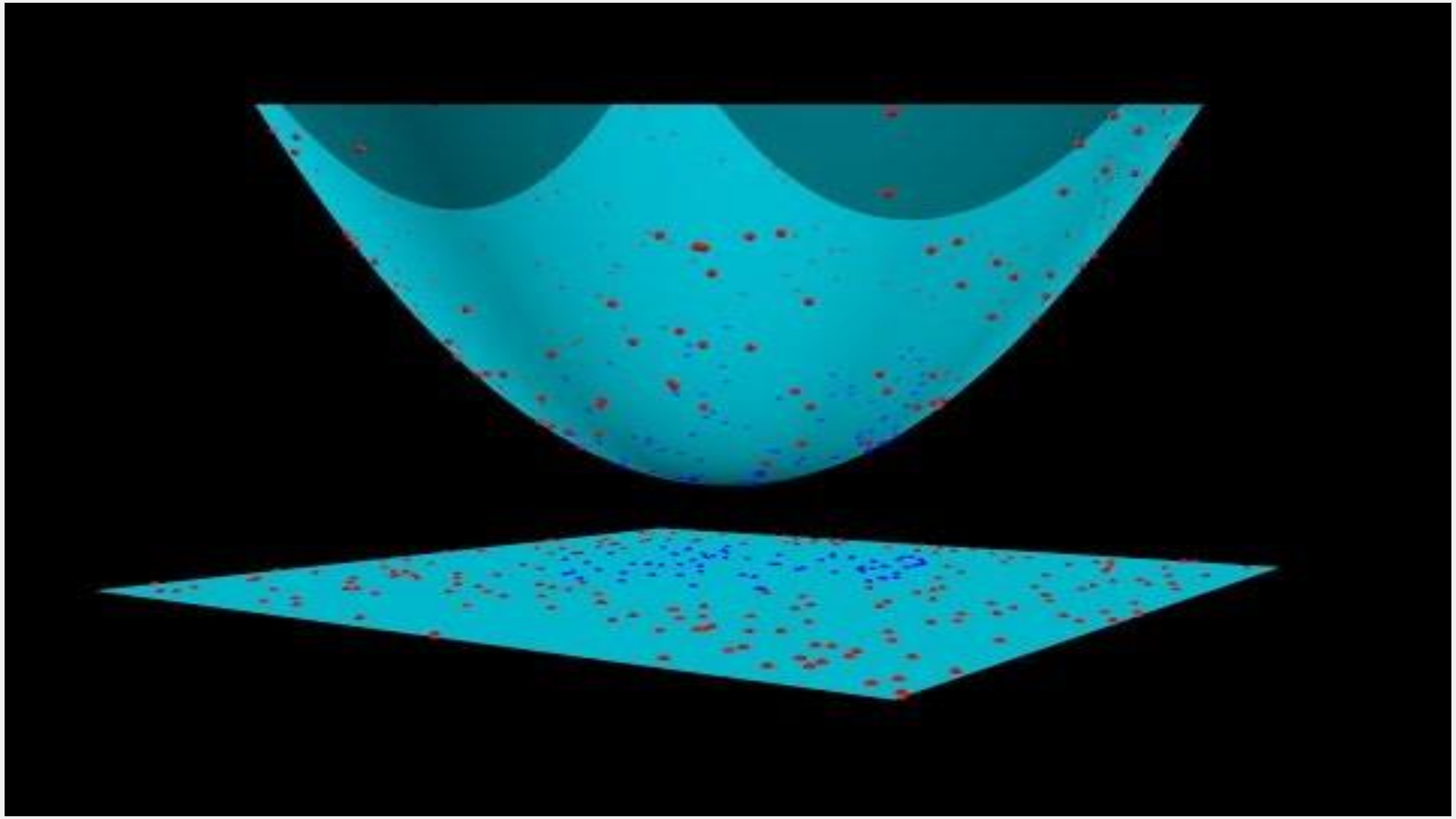




[Reference]

Supervised
Support Vector Machine

Non-linear function



[Reference]

Supervised

Support Vector Machine

How to tune parameters

$$\text{Linear kernel: } k(x,y) = < x, y >$$

$$\text{Polynomial kernel: } k(x,y) = (< x, y > + c)^d$$

$$\text{Gaussian Radial Basis Function kernel (RBF): } k(x,y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$$

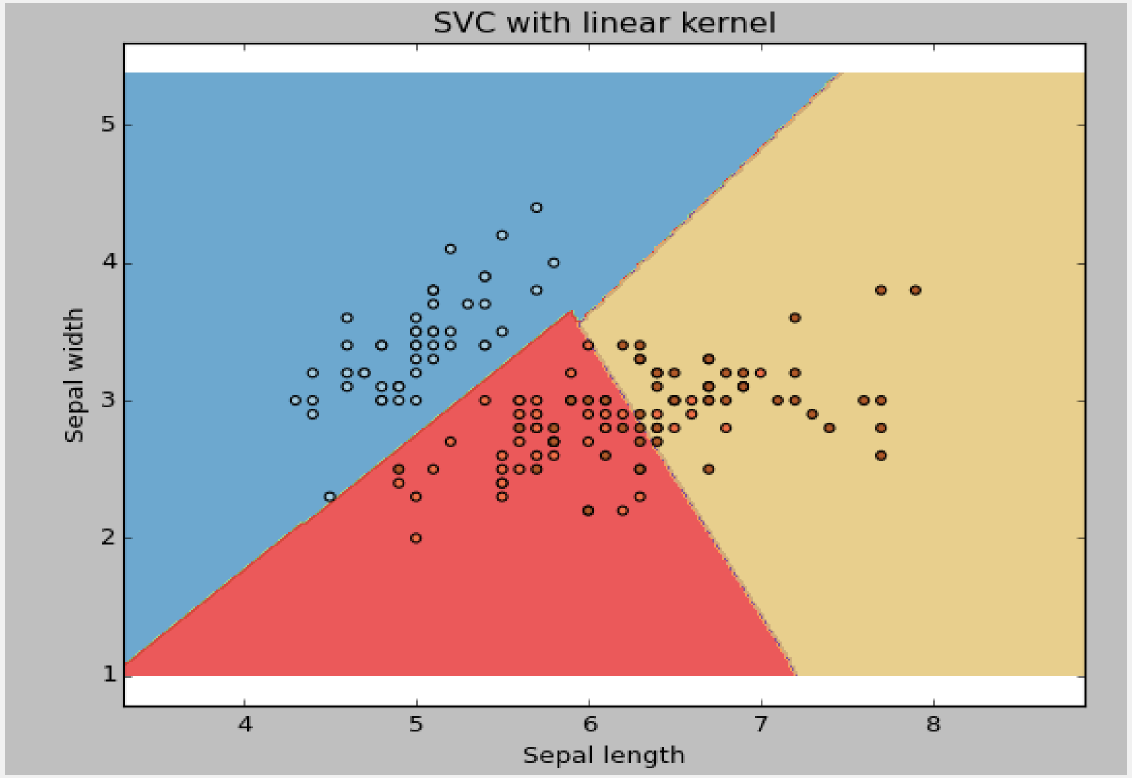
$$d \in Z^+, \sigma \in \mathcal{R} - \{0\}$$

- C: C means low error and if the number of instances that are allowed to fall within the margin, C influences the number of support vectors used by the model
- Gamma: Higher the value of gamma, will try to exact fit the as per training data set i.e. generalization error and cause over-fitting problem

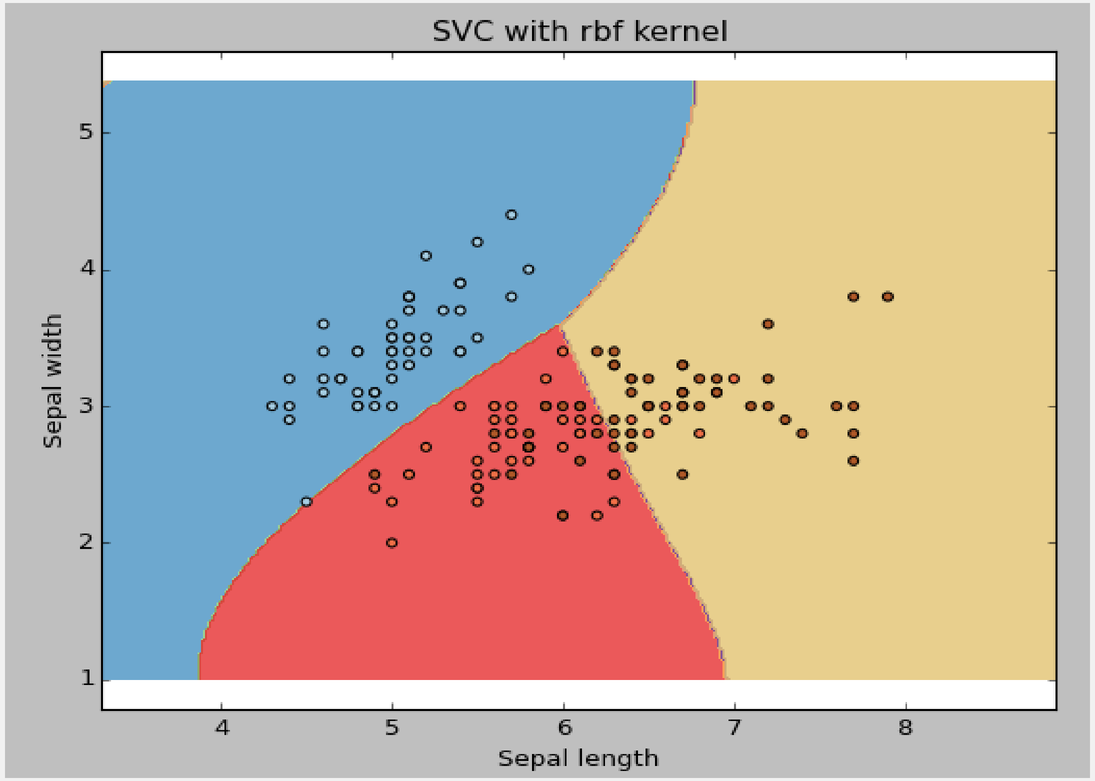
Supervised
Support Vector Machine

How to tune parameters

kernel=‘linear'



kernel='rbf'

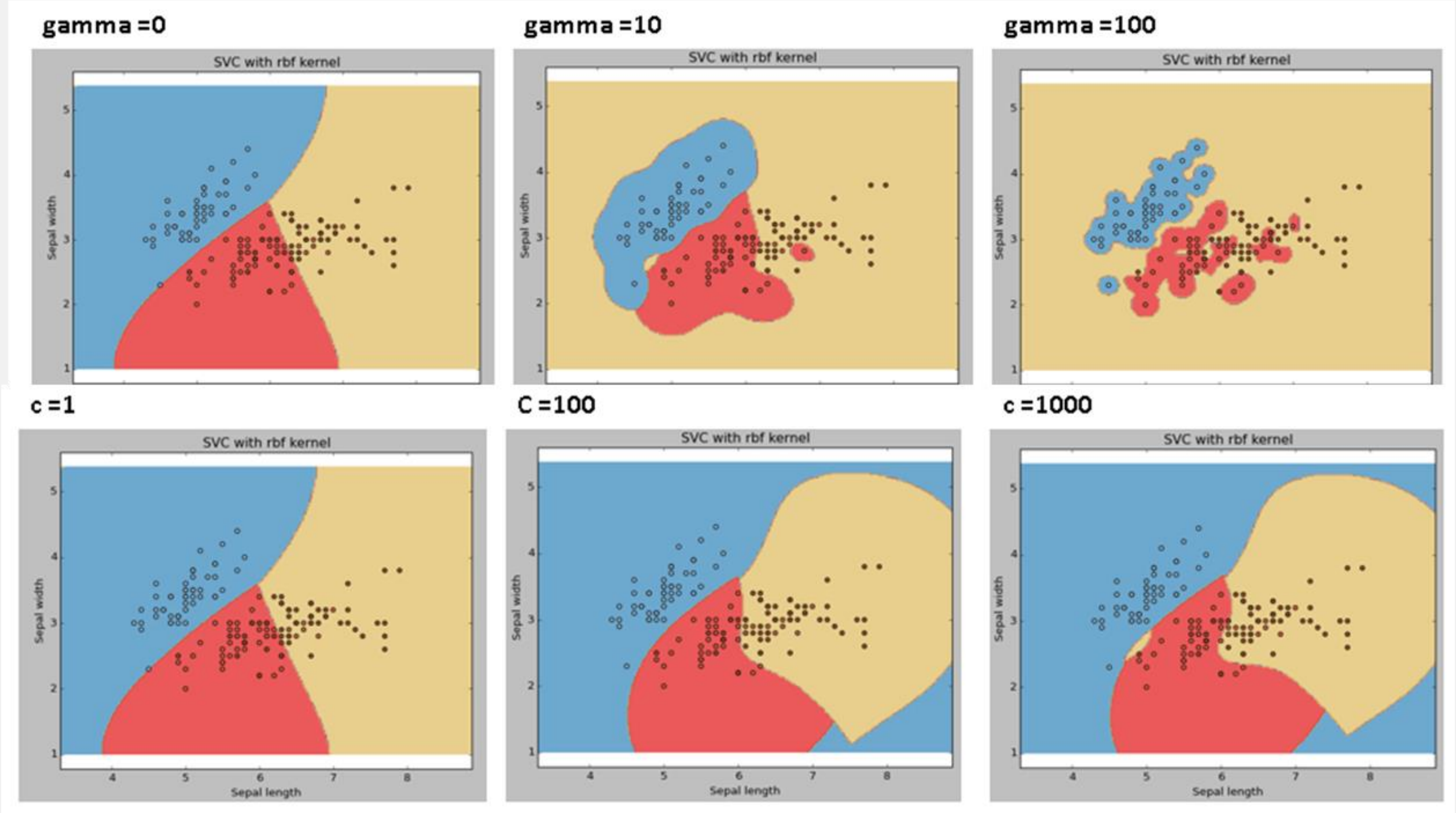


Reference

Supervised

Support Vector Machine

How to tune parameters



Supervised

Support Vector Machine

- grid search

```
from sklearn import svm, grid_search

def svc_param_selection(X, y, nfolds):
    Cs = [0.001, 0.01, 0.1, 1, 10]
    gammas = [0.001, 0.01, 0.1, 1]
    param_grid = {'C': Cs, 'gamma' : gammas}
    grid_search = GridSearchCV(svm.SVC(kernel='rbf'), param_grid,
cv=nfolds)
    grid_search.fit(X, y)
    grid_search.best_params_
    return grid_search.best_params_
```

Supervised

Support Vector Machine

Pros.

- It works really well with clear margin of separation
- It is effective in high dimensional spaces
- It is effective in cases where number of dimensions is greater than the number of samples

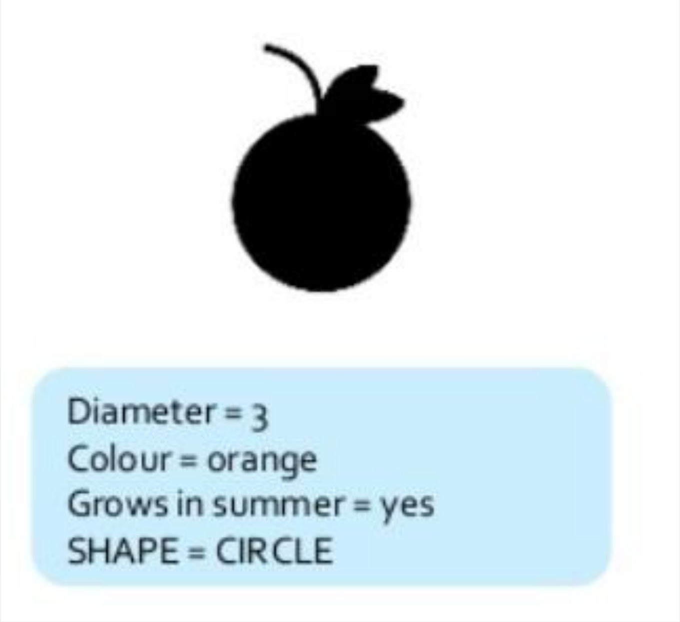
Cons.

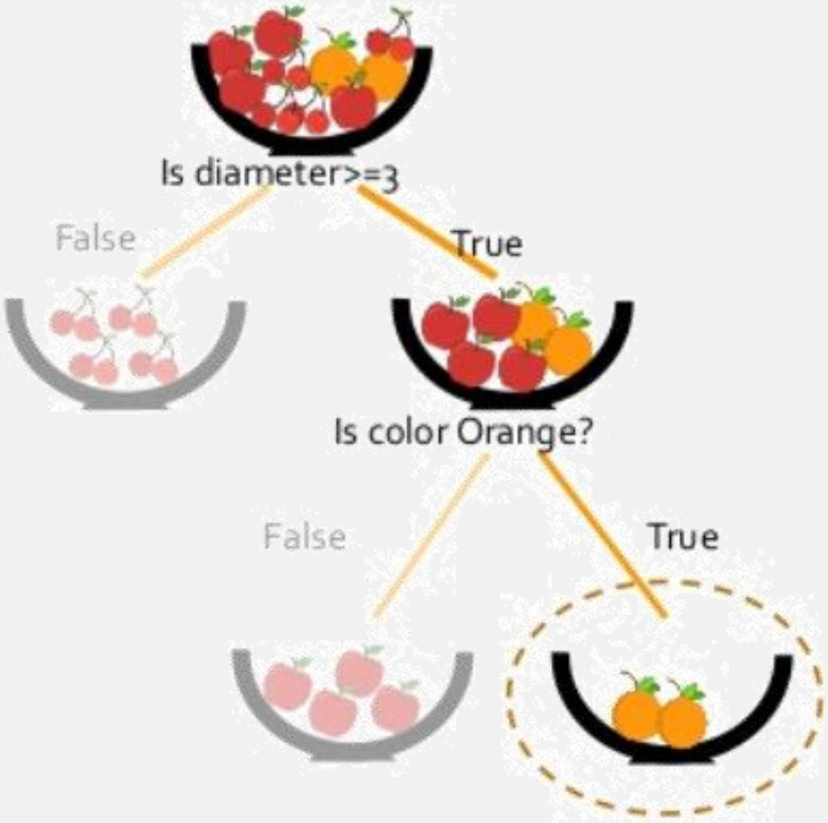
- It doesn't perform well, when we have large data set because the required training time is higher
- It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping

Supervised
Decision Tree

Decision Trees

- It is a conditional classifier that specializes in the classification problems with tree-like structure





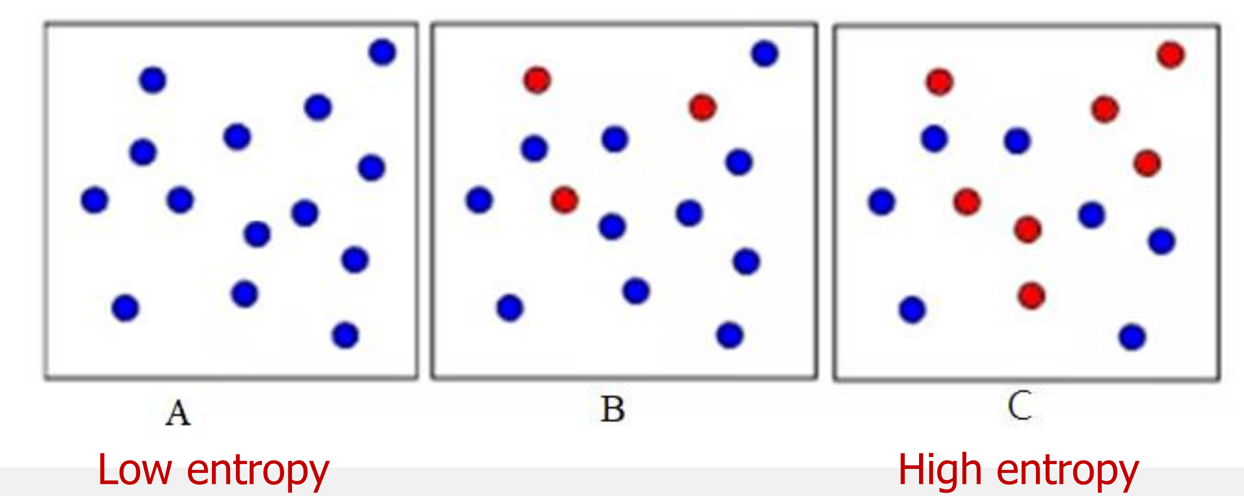
Supervised

Decision Tree

A decision tree is constructed based on the existing data, usually in a "top to down" approach, dividing the whole group into several subgroups according to a certain characteristic

From each subgroup, according to a certain characteristic, the subgroup is divided into small subgroups, until the data in the subgroups are all of the same category

Problem: How to choose a best feature to cluster at each step?



Supervised

Decision Tree

Information Entropy: measure of randomness

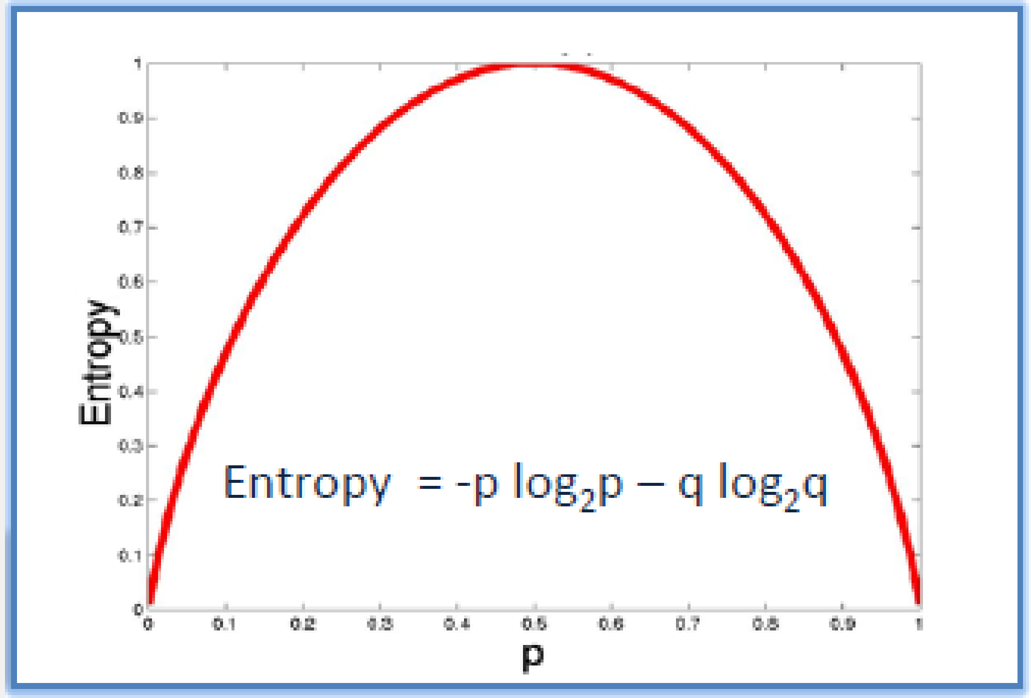
More randomness in the data => more entropy

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

D:代表某一個特徵, 且有m個類別

p: 是某一個類別在這個特徵中出現的機率

若只有兩類時:
當資料全部都是同一類, 則entropy=0;
若資料是各自一半時, entropy=1



Entropy = -0.5 $\log_2$0.5 – 0.5 $\log_2$0.5 = 1

Supervised

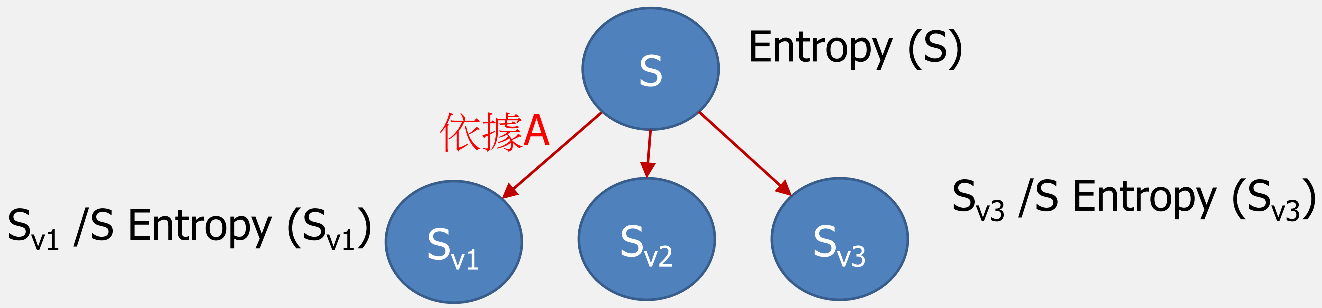
Decision Tree

How to determine the best features?

- After obtaining entropy, the information gain is calculated
- Information gain: to evaluate the eigenvalues for data classification
- The information gain, Gain(S, A) of an attribute A, relative to the collection of examples S

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{S_v}{S} Entropy(S_v)$$

Where Values (A) is the set of all potential values for attribute A, and S_v is the subset of S for which the attribute A has value v.



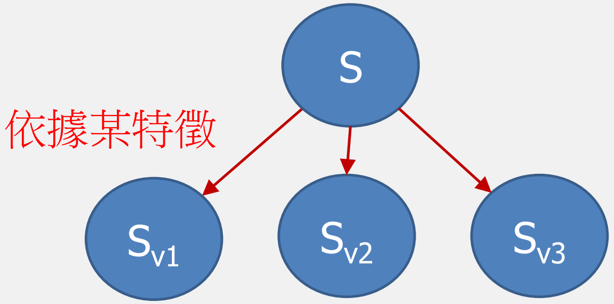
Supervised

Decision Tree

An example

- During two weeks, The target is “play ball?"
- which can be Yes or No

Outlook	Temp	Humidity	Windy	Play Golf
Rain	Hot	High	false	No
Rain	Hot	High	true	No
overcast	Hot	High	false	Yes
Sunny	Mild	High	false	Yes
Sunny	Cool	normal	false	Yes
Sunny	Cool	normal	true	No
overcast	Cool	normal	true	Yes
Rain	Mild	High	false	No
Rain	Cool	normal	false	Yes
Sunny	Mild	normal	false	Yes
Rain	Mild	normal	true	Yes
overcast	Mild	High	true	Yes
overcast	Hot	normal	false	Yes
Sunny	Mild	High	true	NO



Step 1: Calculate entropy of Play Golf (target)

Play Golf	
Yes	No
9	5

$$
\begin{aligned}
\text{Entropy(Play Golf)} &= \text{Entropy}(5,9) \\
&= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\
&= 0.94
\end{aligned}
$$

Supervised

Decision Tree

Step 2: The dataset is then split into the different attributes

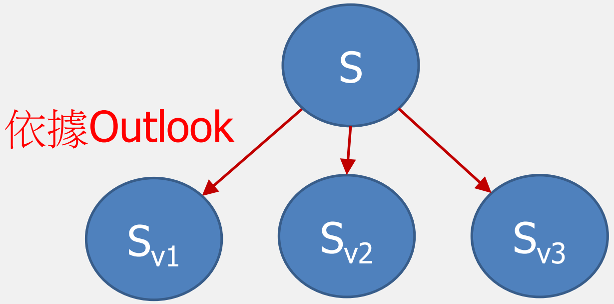
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain=0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain=0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain=0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain=0.048			

IG 愈大表示此特徵內資料凌亂程度愈小



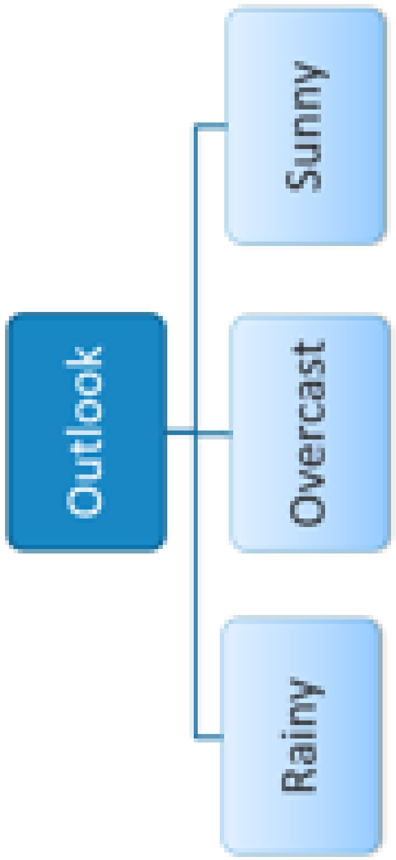
Outlook Gain: 0.94 - (5/14*E(3,2)+4/14*E(4,0)+5/14*E(2,3))=0.247
Temp Gain: 0.94 - (4/14*E(2,2)+6/14*E(4,2)+4/14*E(3,1))=0.029
Humility Gain: 0.94 - (7/14*E(3,4)+7/14*E(6,1))=0.152
Windy Gain: 0.94 - (8/14*E(6,2)+6/14*E(3,3))=0.048

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{S_v}{S} Entropy(S_v)$$

Step 3: pick out attribute with the greatest IG as the decision node

Supervised

Decision Tree



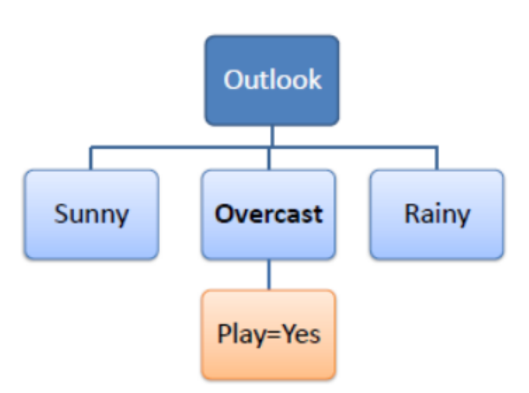
Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Classification

Road Map - Decision Tree

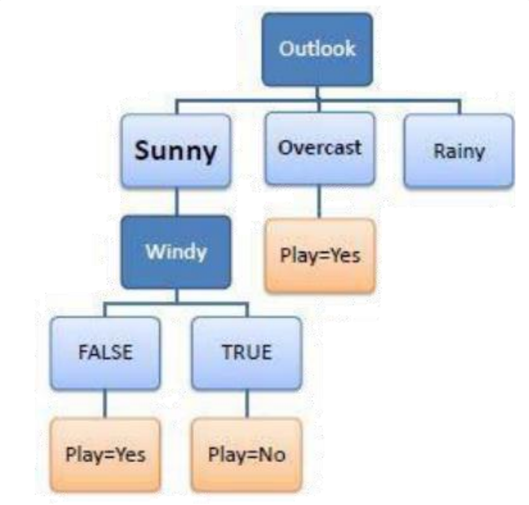
Step 4a: A branch with the entropy of Outlook= overcast

Temp	Humidity	Windy	Play Golf
hot	high	False	Yes
cool	normal	True	Yes
mild	high	True	Yes
hot	normal	False	Yes



Step 4b: A branch with entropy of Outlook= sunny (Windy=False & Windy=True)

Temp	Humidity	Windy	Play Golf
mild	high	False	Yes
cool	normal	False	Yes
mild	normal	False	Yes
mild	high	True	NO
cool	normal	True	No



Supervised

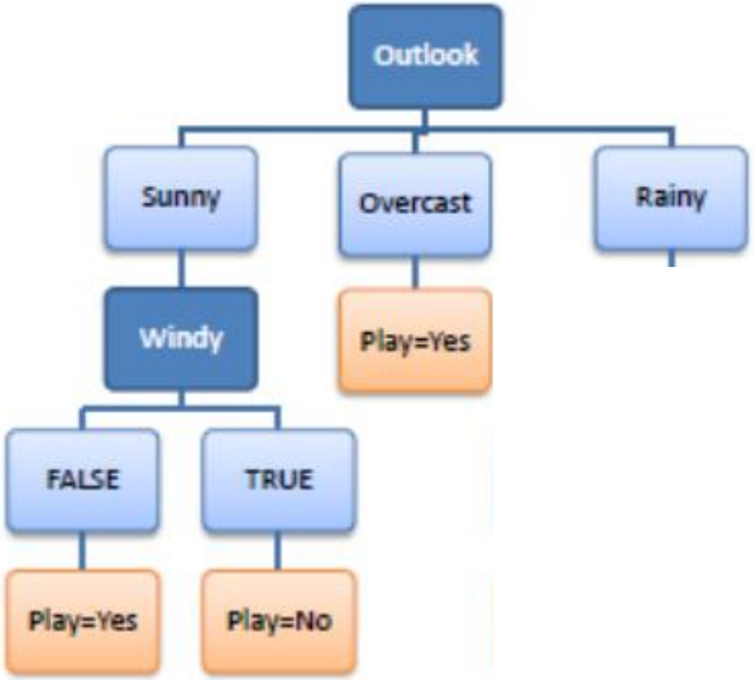
Decision Tree

Step 4c: A branch with entropy of Outlook= rain (Humidity= high & Humidity = normal)

Temp	Humidity	Windy	Play Golf
hot	high	false	No
hot	high	true	No
mild	high	false	No
Cool	normal	false	Yes
mild	normal	true	Yes

Exercise：

試著計算Temp, Humidity, Windy 的 Information Gain，並畫出在Rainy 下的樹狀結構



Decision Tree of Weather data sets

Supervised

Decision Tree

Pros.

- ➢ Implicitly perform feature selection
- ➢ Easy to interpret and explain
- ➢ Can generate rules helping experts to formalize their knowledge
- ➢ Data classification without much calculations
- ➢ Handling both continuous and discrete data
- ➢ Require relatively little effort from users for data preparation
  - they do not need variable scaling
  - they can deal with a reasonable amount of missing values
  - they are not affected by outliers

Supervised

Decision Tree

Cons.

- The high classification error rate while training set is small in comparison with the number of classes
- Exponential calculation growth while problem is getting bigger

02

Unsupervised Learning

Unsupervised

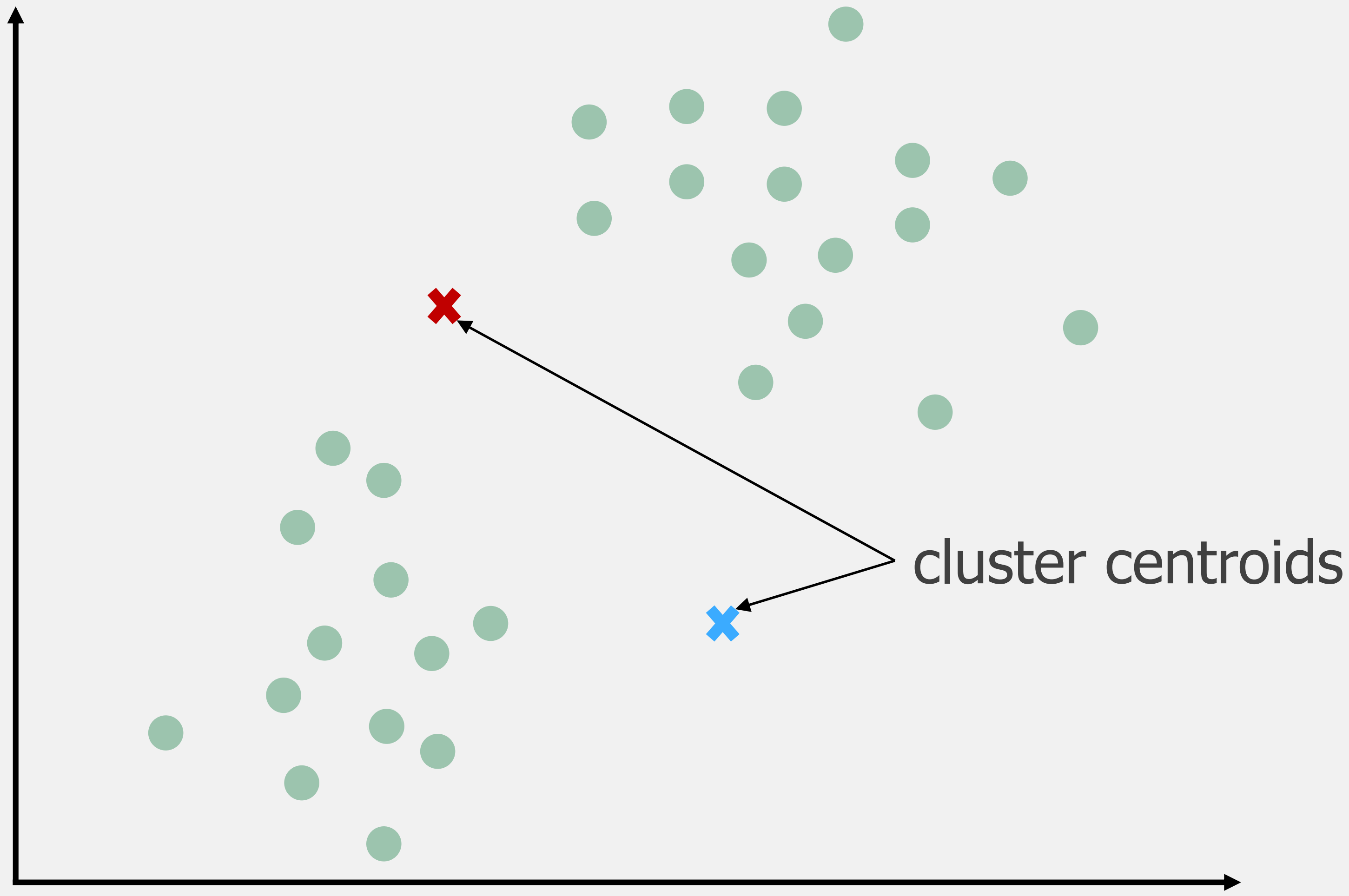
K-means

The concept of the K-means Clustering method is simple, which is "birds of a feather flock together"

1. Choose K centroids randomly or with other algorithms
2. Assign each data point to the nearest centroid, forming K clusters
3. Calculate the centroid of each cluster
4. Repeat steps 2 and 3 until the centroids no longer move or the maximum iteration is reached

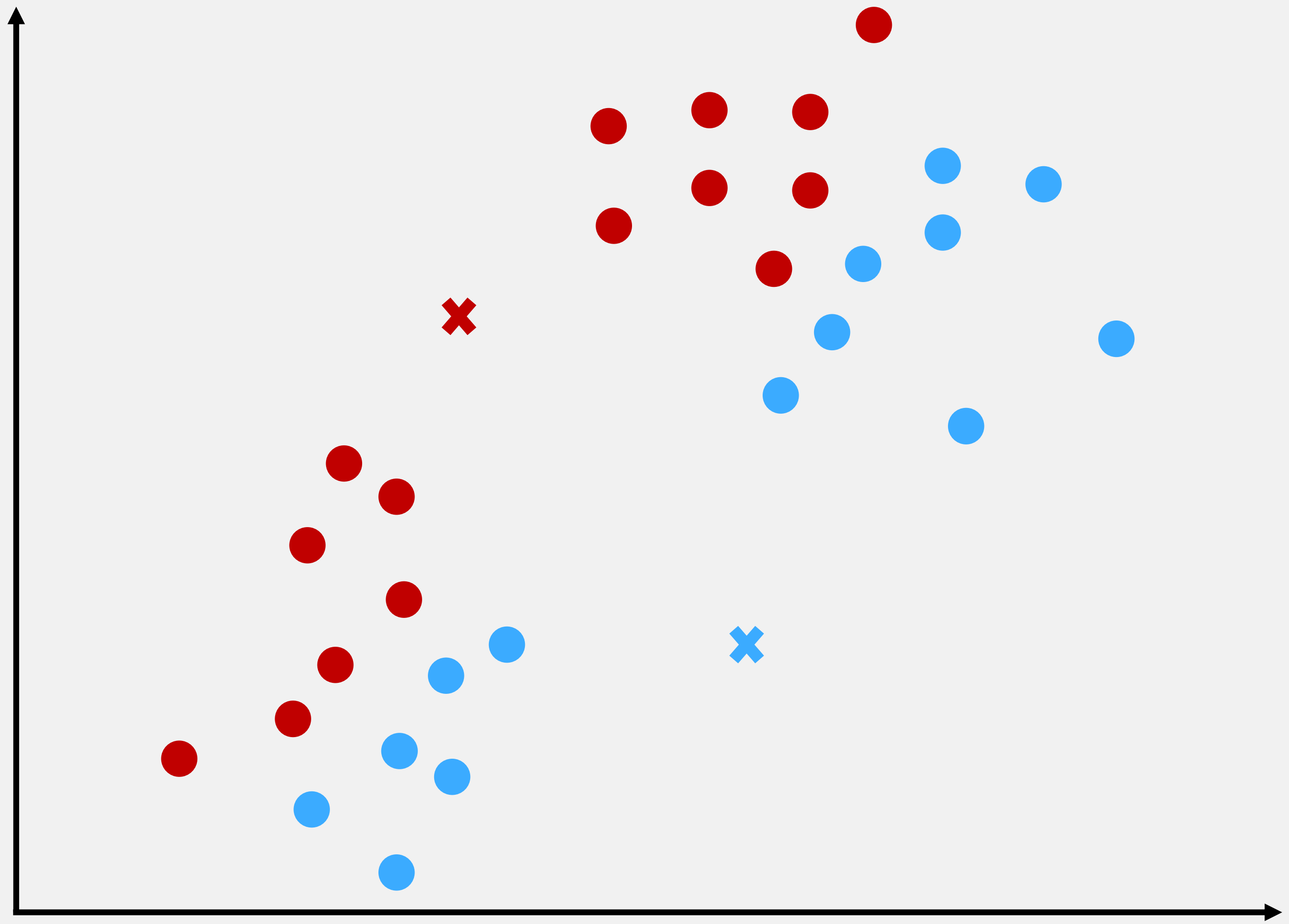
Unsupervised

K-means



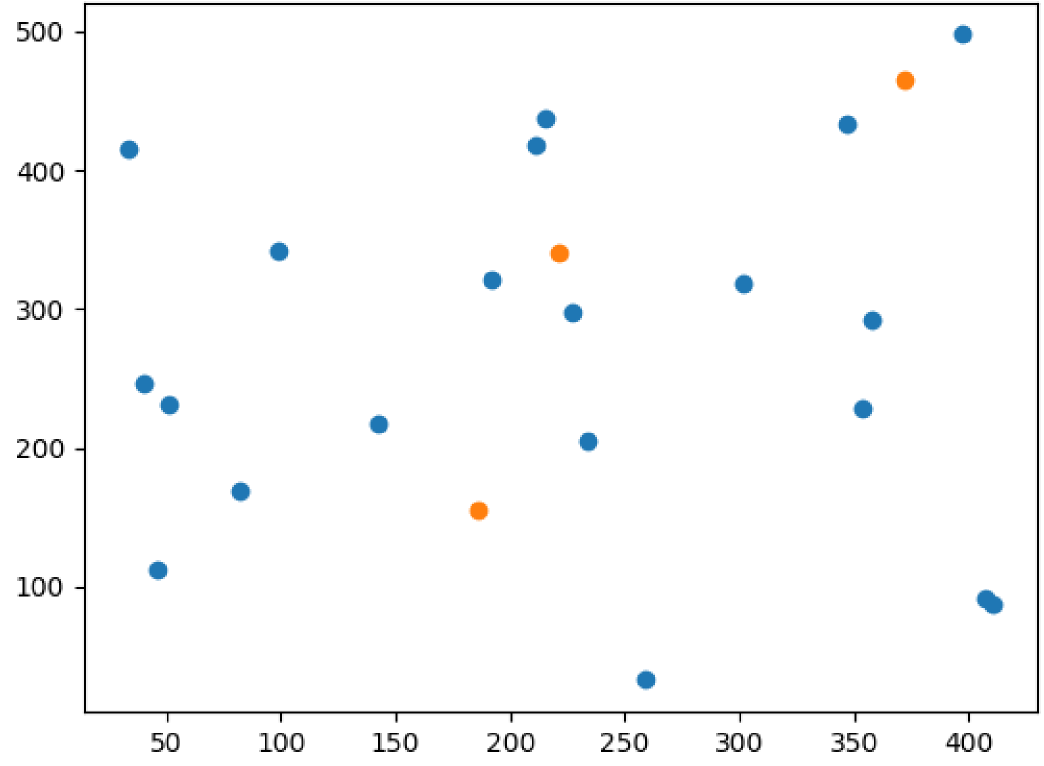
Unsupervised

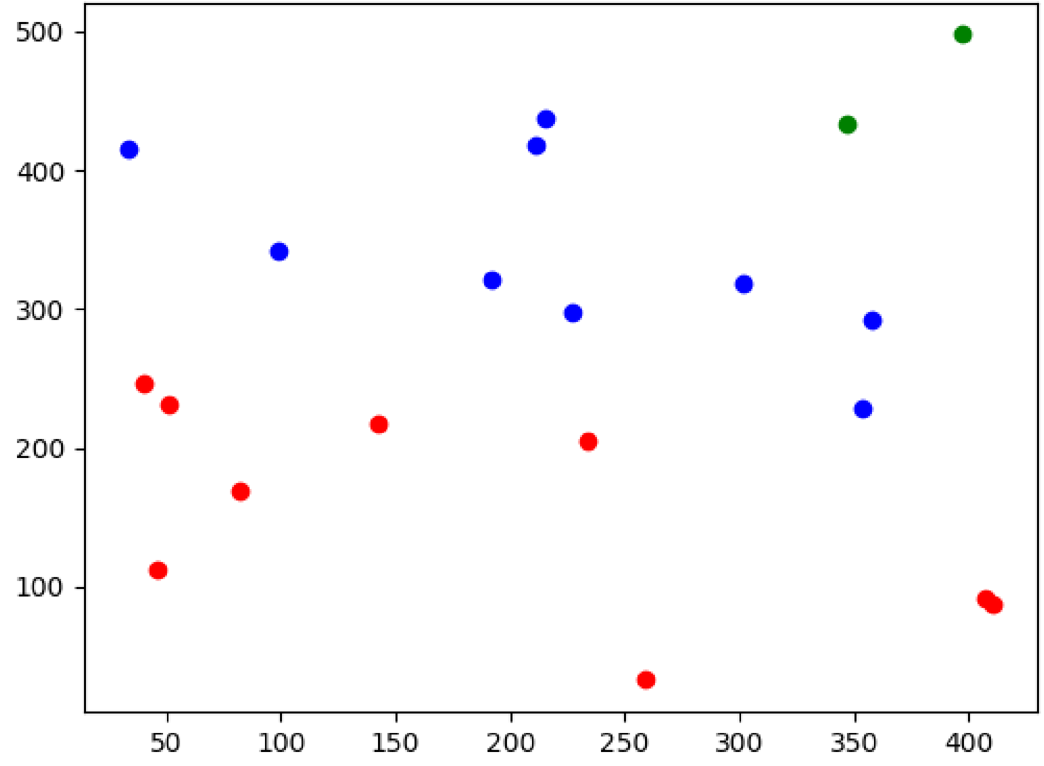
K-means



Unsupervised

K-means





Unsupervised

K-means

Pros.

- Fast computation, ease of implementation, and interpretation
- Handle large datasets

Cons.

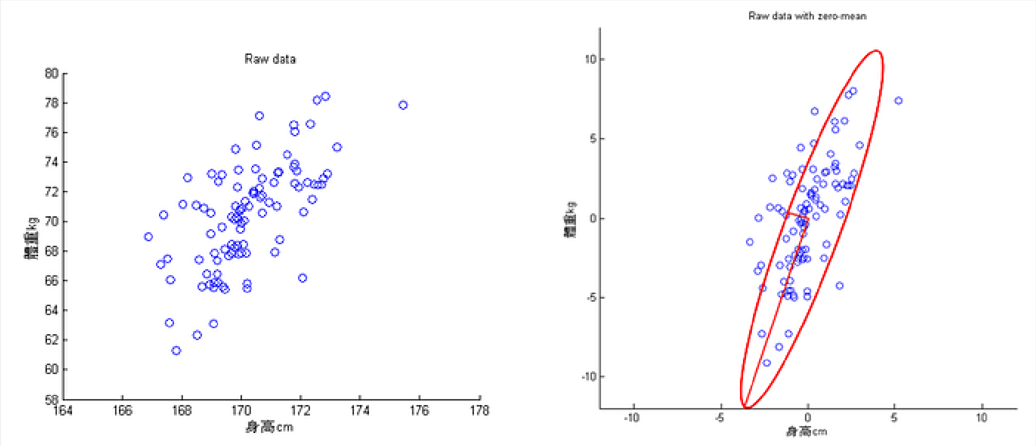
- Require manually setting the number of clusters K, and the result usually depends on the initialization of centroids
- Sensitive to noise and outliers

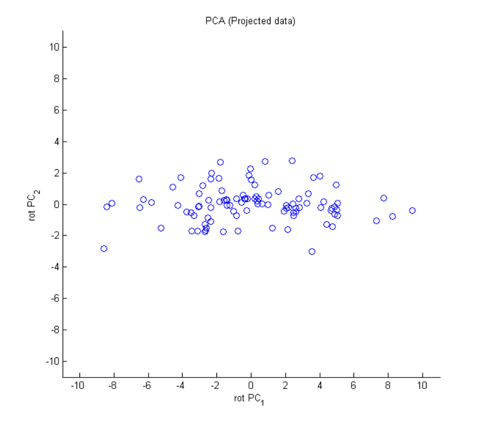
Unsupervised

Principle Component Analysis

PCA is a linear dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional representation while retaining most of the original data's variability

1. Data standardization
2. Covariance matrix calculation
3. Eigenvalues and eigenvectors computation
4. K principal components selection
5. Data translation
6. Visualization and analysis





Source

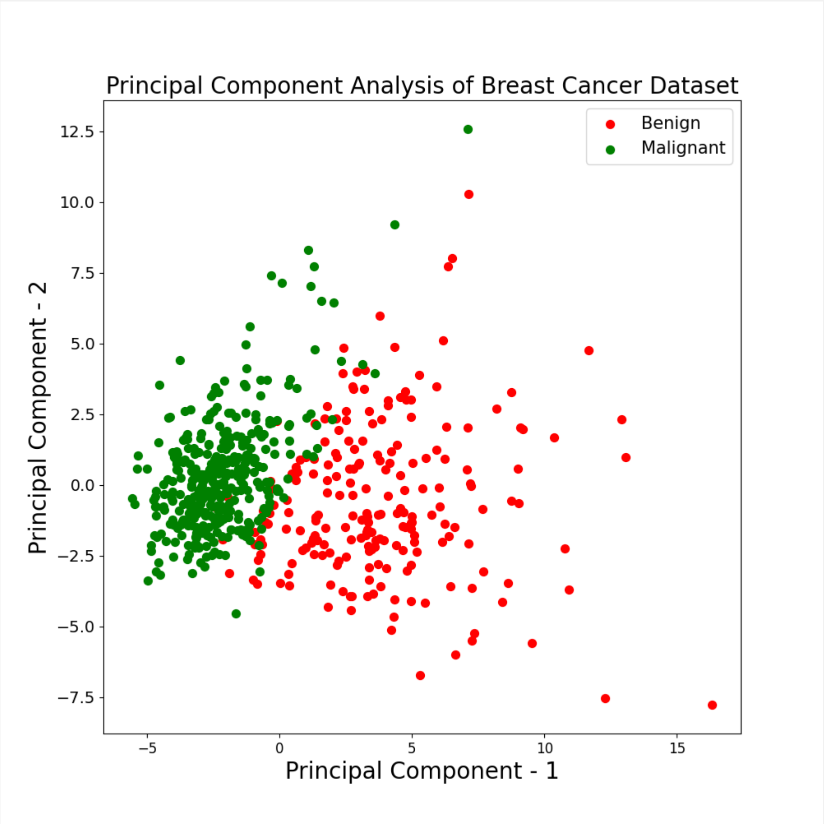
Unsupervised

Principle Component Analysis

```
[5 rows x 31 columns]
     mean radius  mean texture  mean perimeter  ...  worst symmetry  worst fractal dimension      label
0          17.99         10.38          122.80  ...          0.4601                  0.11890     Benign
1          20.57         17.77          132.90  ...          0.2750                  0.08902     Benign
2          19.69         21.25          130.00  ...          0.3613                  0.08758     Benign
3          11.42         20.38           77.58  ...          0.6638                  0.17300     Benign
4          20.29         14.34          135.10  ...          0.2364                  0.07678     Benign
..           ...           ...             ...  ...             ...                      ...        ...
564        21.56         22.39          142.00  ...          0.2060                  0.07115     Benign
565        20.13         28.25          131.20  ...          0.2572                  0.06637     Benign
566        16.60         28.08          108.30  ...          0.2218                  0.07820     Benign
567        20.60         29.33          140.10  ...          0.4087                  0.12400     Benign
568         7.76         24.54           47.92  ...          0.2871                  0.07039  Malignant

[569 rows x 31 columns]
     feature0  feature1  feature2  feature3  feature4  ...  feature25  feature26  feature27  feature28  feature29
564  2.110995  0.721473  2.060786  2.343856  1.041842  ...  -0.273318   0.664512   1.629151  -1.360158  -0.709091
565  1.704854  2.085134  1.615931  1.723842  0.102458  ...  -0.394820   0.236573   0.733827  -0.531855  -0.973978
566  0.702284  2.045574  0.672676  0.577953 -0.840484  ...   0.350735   0.326767   0.414069  -1.104549  -0.318409
567  1.838341  2.336457  1.982524  1.735218  1.525767  ...   3.904848   3.197605   2.289985   1.919083   2.219635
568 -1.808401  1.221792 -1.814389 -1.347789 -3.112085  ...  -1.207552  -1.305831  -1.745063  -0.048138  -0.751207
```

```
[5 rows x 30 columns]
     principal component 1  principal component 2
0                 9.192837               1.948583
1                 2.387802              -3.768172
2                 5.733896              -1.075174
3                 7.122953              10.275589
4                 3.935302              -1.948072
..                     ...                    ...
564               6.439315              -3.576817
565               3.793382              -3.584048
566               1.256179              -1.902297
567              10.374794               1.672010
568              -5.475243              -0.670637
```

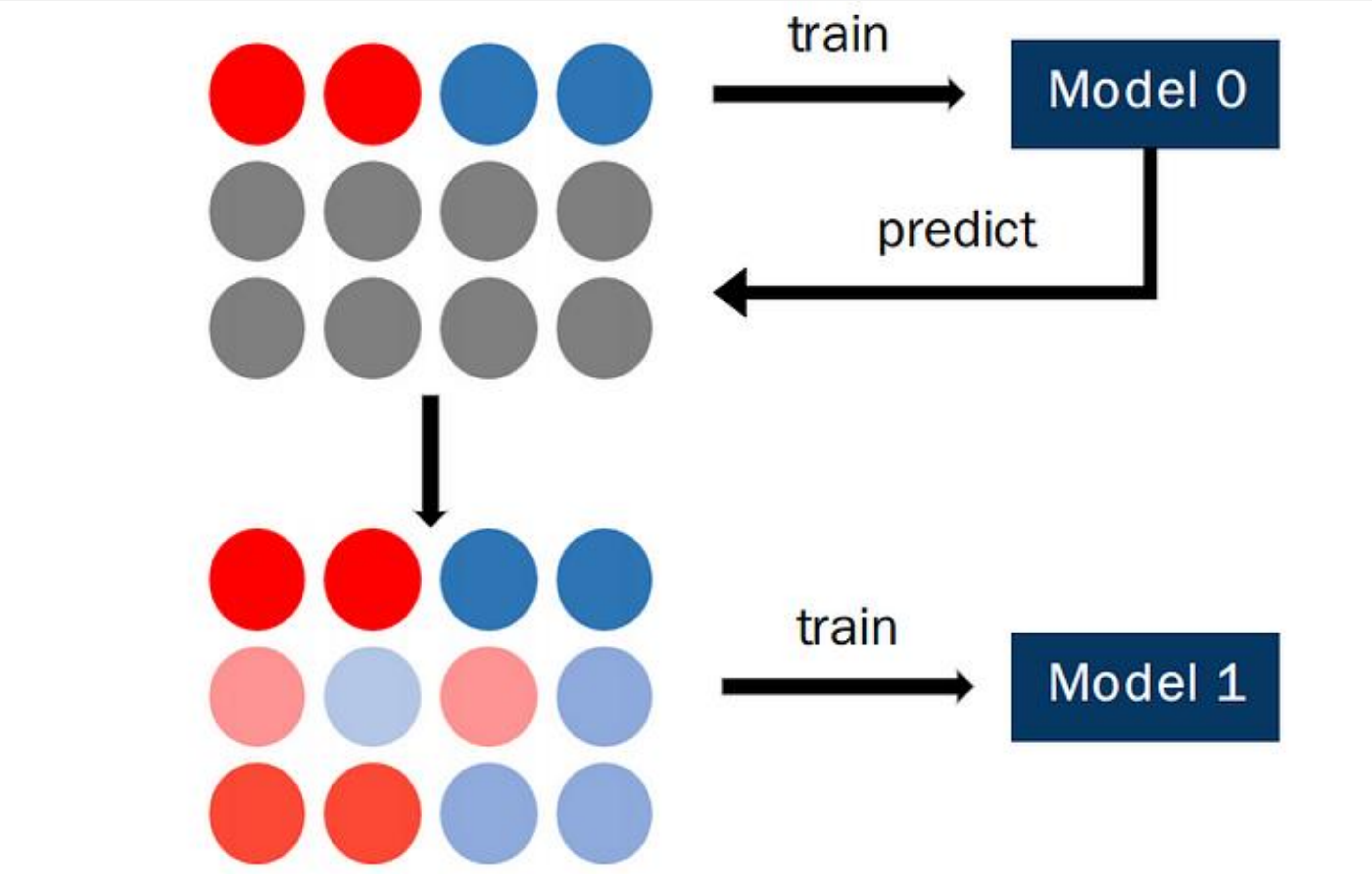


03

Semi-supervised Learning

Semi-supervised

Semi-supervised learning refers to the use of a small amount of labeled data and a large amount of unlabeled data to train a machine learning model



Source

04
Self-supervised Learning (SSL)

Self-supervised Learning

Introduction

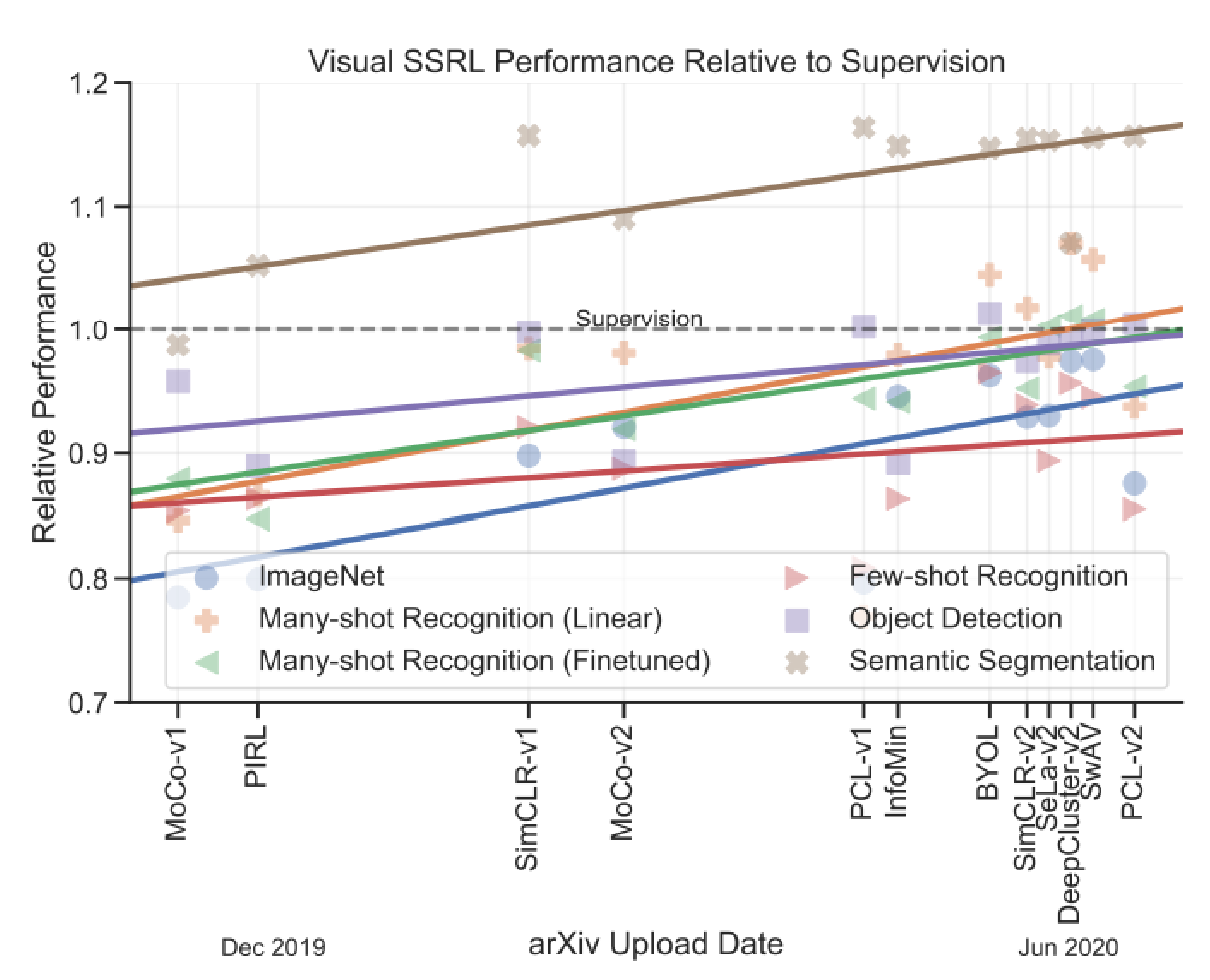
Self-supervised learning (SSL)

- It is an evolving machine learning technique to solve the challenges posed by the over-dependence of labeled data
- A special type of representation learning via unlabeled data
- Model trains itself to learn one part of the input from another part of the input

Why do we need SSL?

- High cost - The cost of good quality labeled data is very high in terms of time and money
- Lengthy lifecycle - The preparation lifecycle is a long process including data clean, annotation, review, and reconstruction

Self-supervised Learning
Introduction



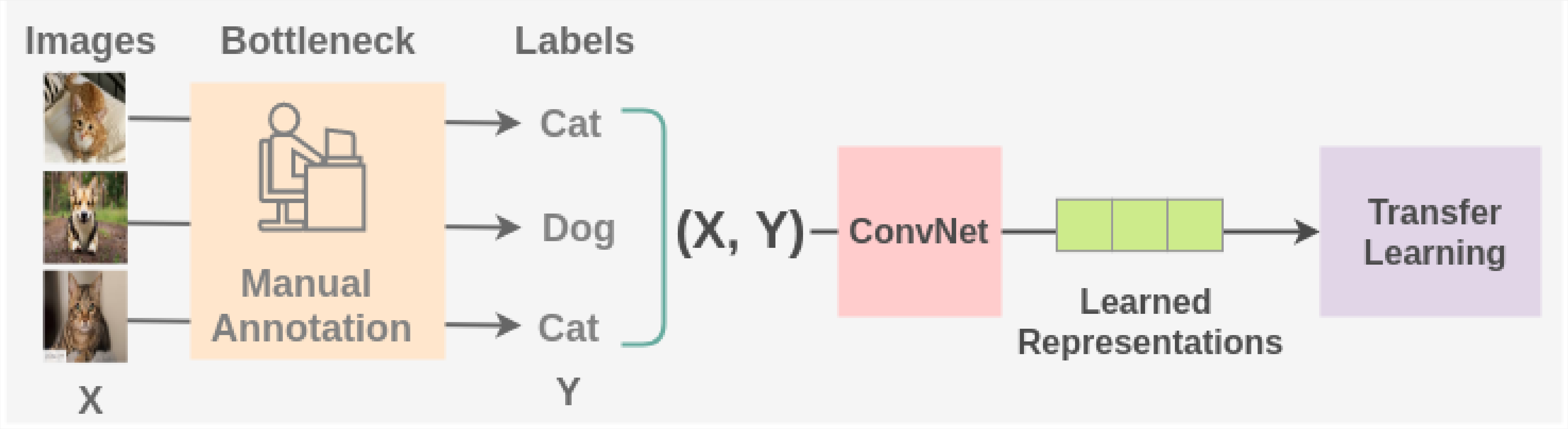
Reference

Self-supervised Learning

Introduction

The workflow of SSL

- Training with unlabeled data to obtain a general representation
- Fine-tuning with few labeled data



Reference

Self-supervised Learning
Introduction

Approaches

- Generative
- Predictive
- Contrastive
- Bootstrapping
- Regularization

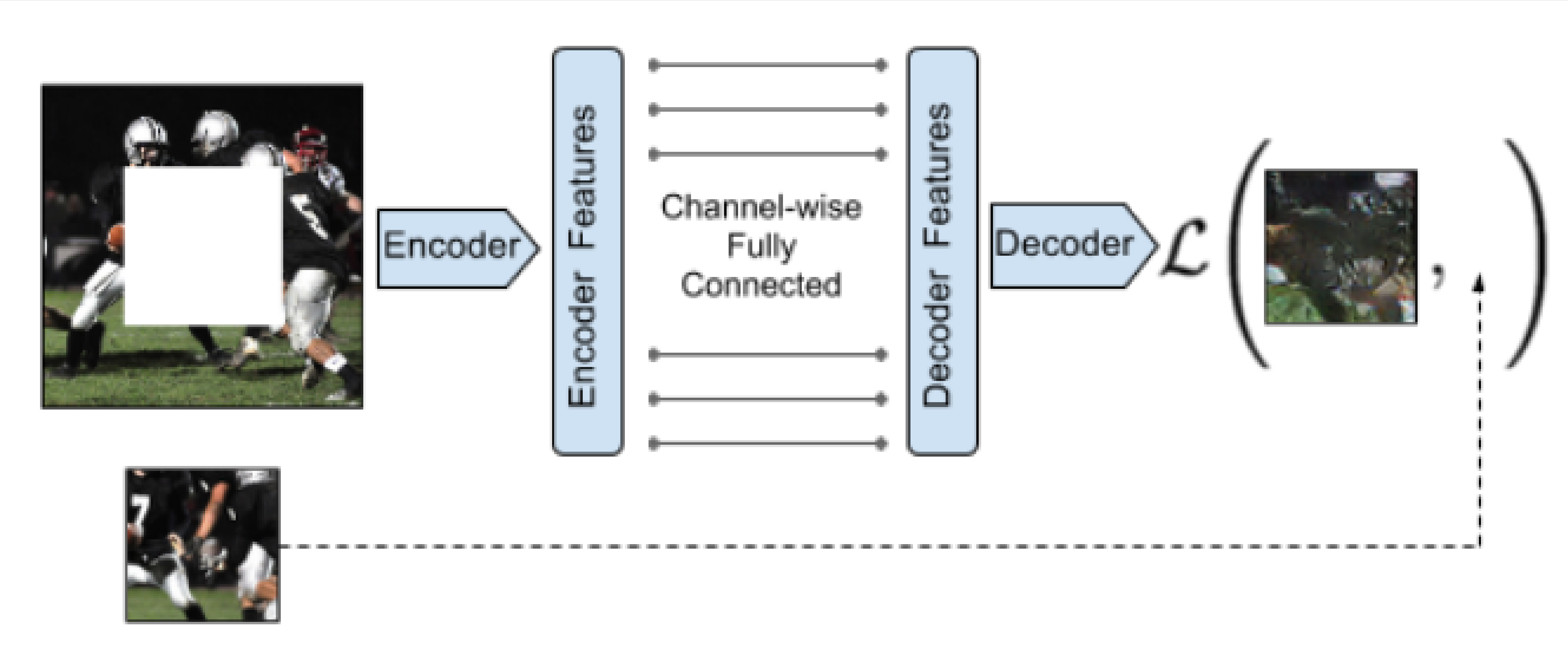
A

Generative

Self-supervised Learning
Generative

Generative

- Training model to reconstruct the pixel space
  - Image inpainting



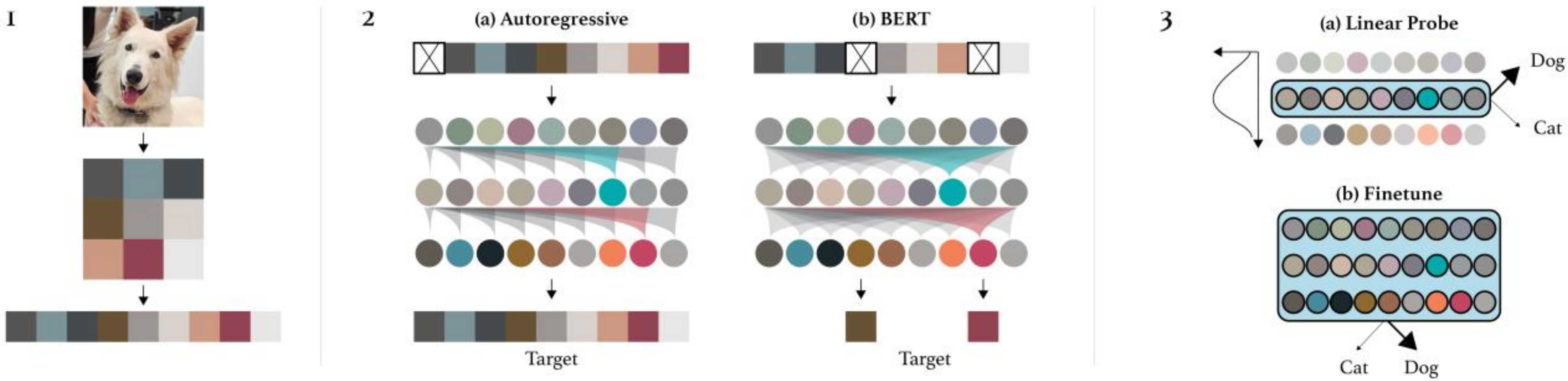
[Reference]

Self-supervised Learning

Generative

Generative

- ➢ Training model to reconstruct the pixel space
  - Image inpainting



[Reference](#)

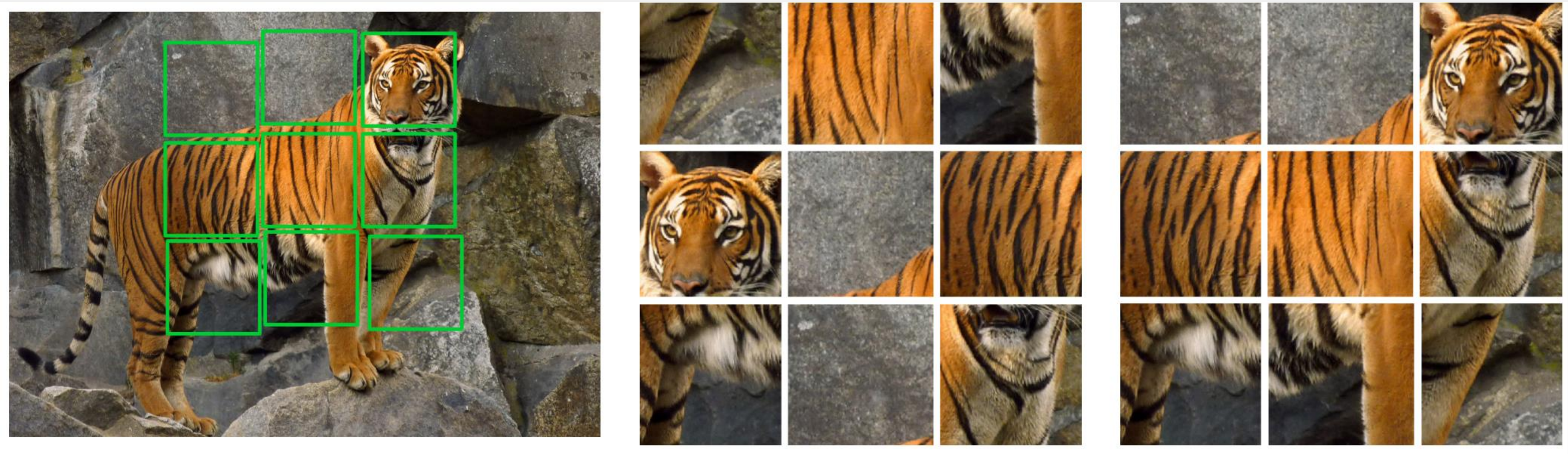
B

Predictive

Self-supervised Learning
Predictive

Predictive

- "Change" and "recovery" image without pixel generation
  - High-level representation generation based on pixel is a hard task
  - Context prediction



(a) (b) (c)

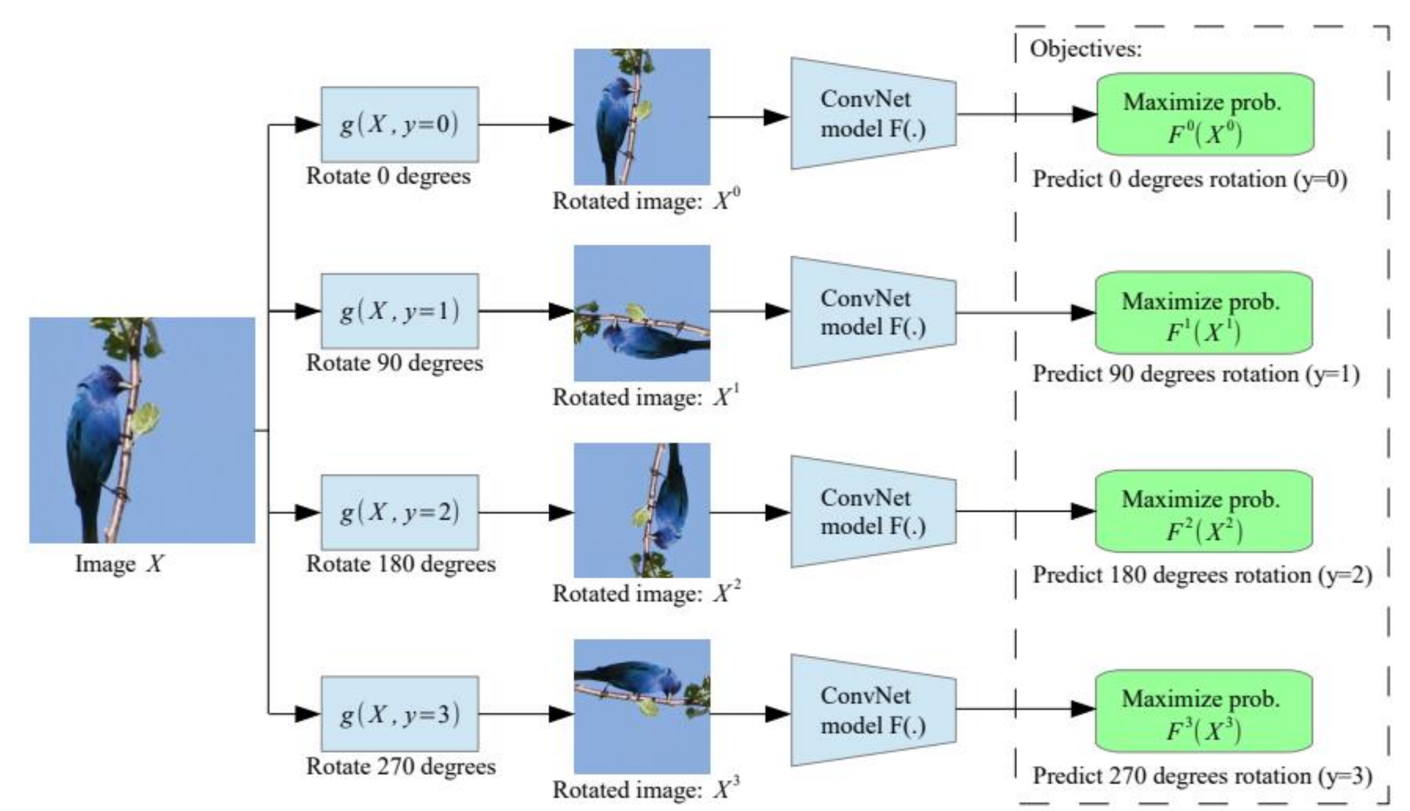
[Reference](#)

Self-supervised Learning

Predictive

Predictive

- "Change" and "recovery" image without pixel generation
  - High-level representation generation based on pixel is a hard task
  - Context prediction



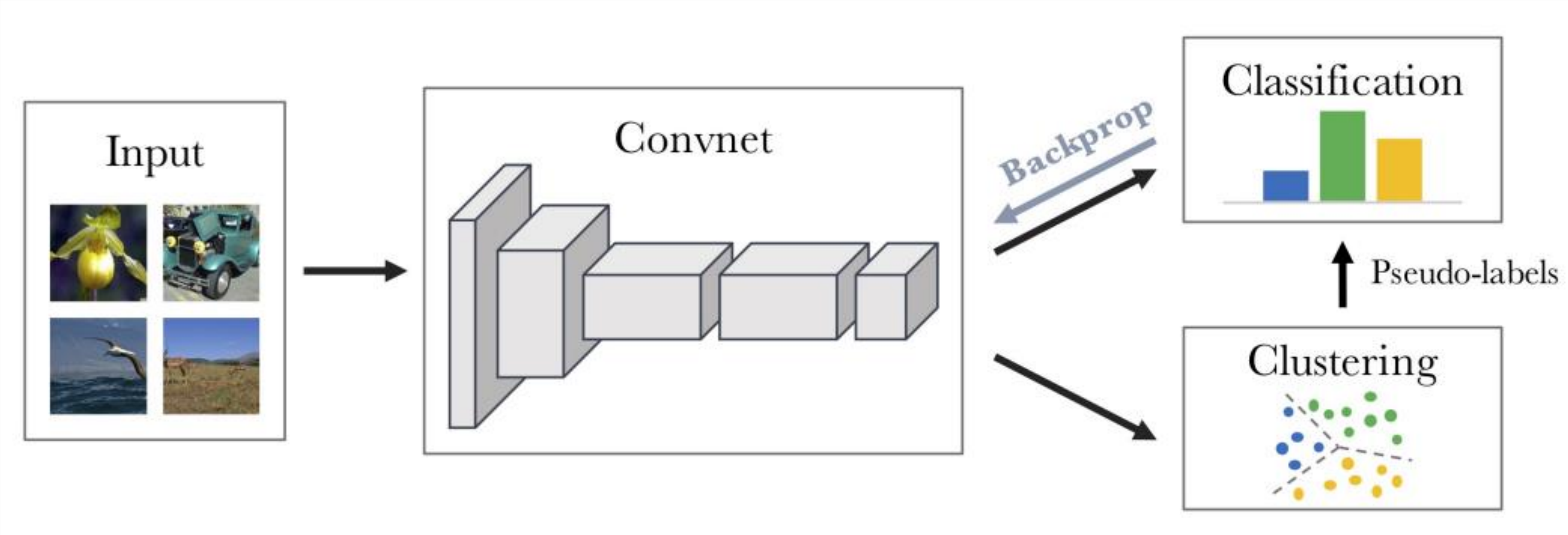
Reference

Self-supervised Learning

Predictive

Predictive

- "Change" and "recovery" image without pixel generation
  - High-level representation generation based on pixel is a hard task
  - Context prediction



[Reference]

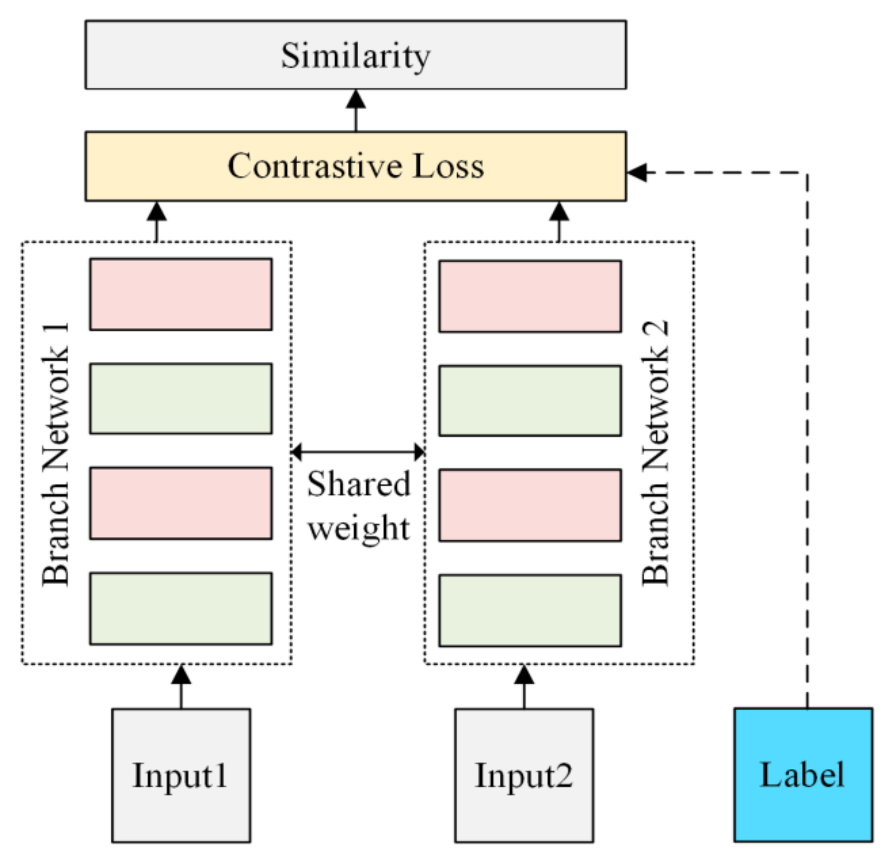
C

Contrastive

Self-supervised Learning
Contrastive

Contrastive

- A widely used approach in SSL
- The higher similarity between images of same class is the better
  - Siamese network



[Reference]

Self-supervised Learning

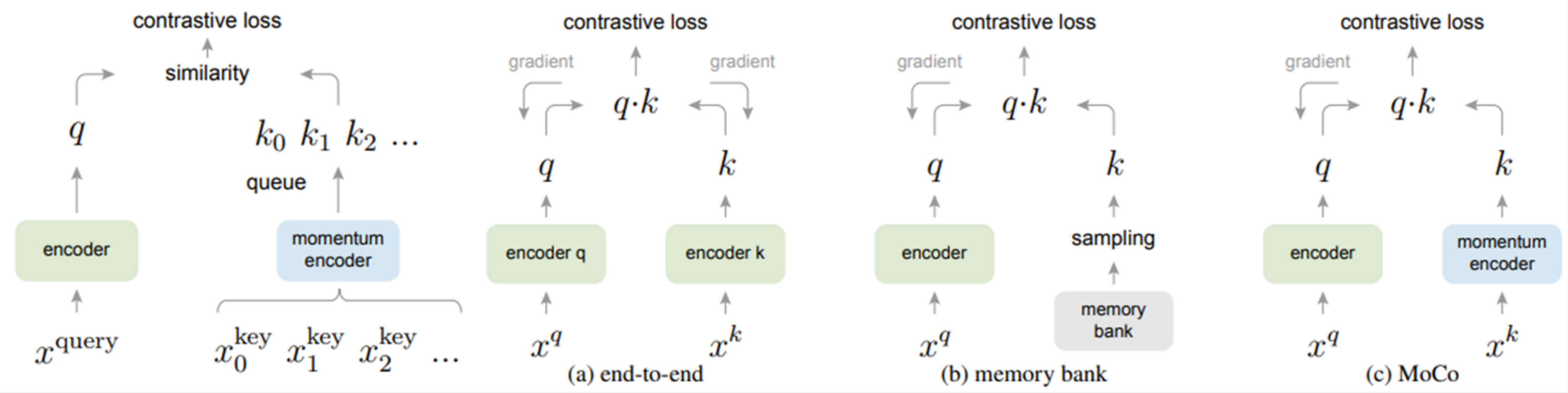
Contrastive

Contrastive

Reference

Contrastive - MoCov1

- Dictionary as a queue
  - Enqueue a batch representation and dequeue the oldest representation
- Momentum encoder
  - Keep queue dictionary data consistent
- Shuffling BN
  - shuffle the data order before training and recovery the order after extracting representation

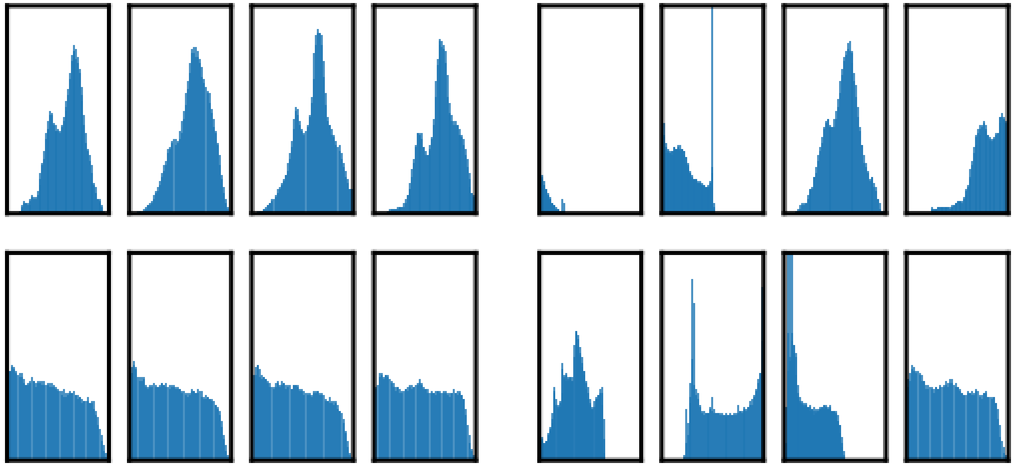


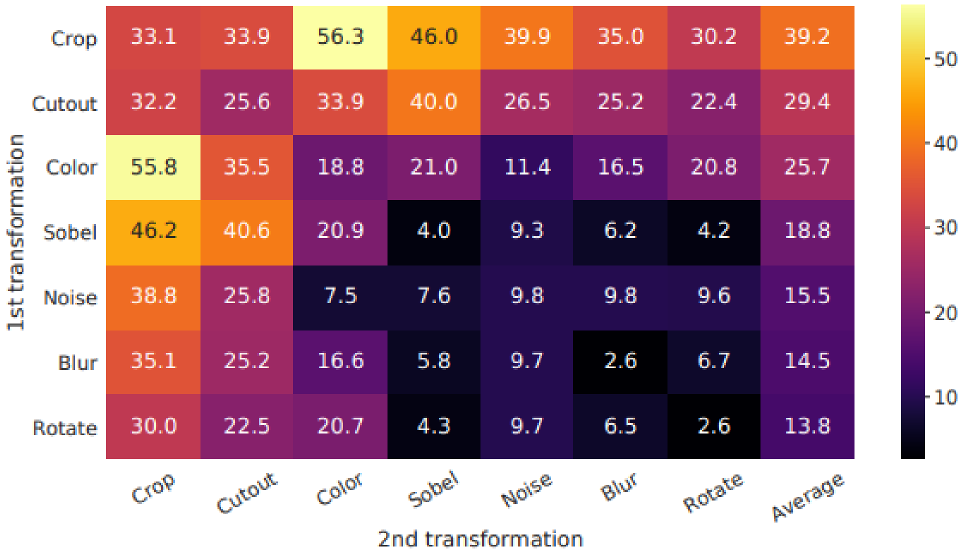
[Reference](#)

Self-supervised Learning
Contrastive

Contrastive - SimCLRv1

- Data augmentation combination
- Projection head
- NT-Xent loss function


(a) Without color distortion. (b) With color distortion.

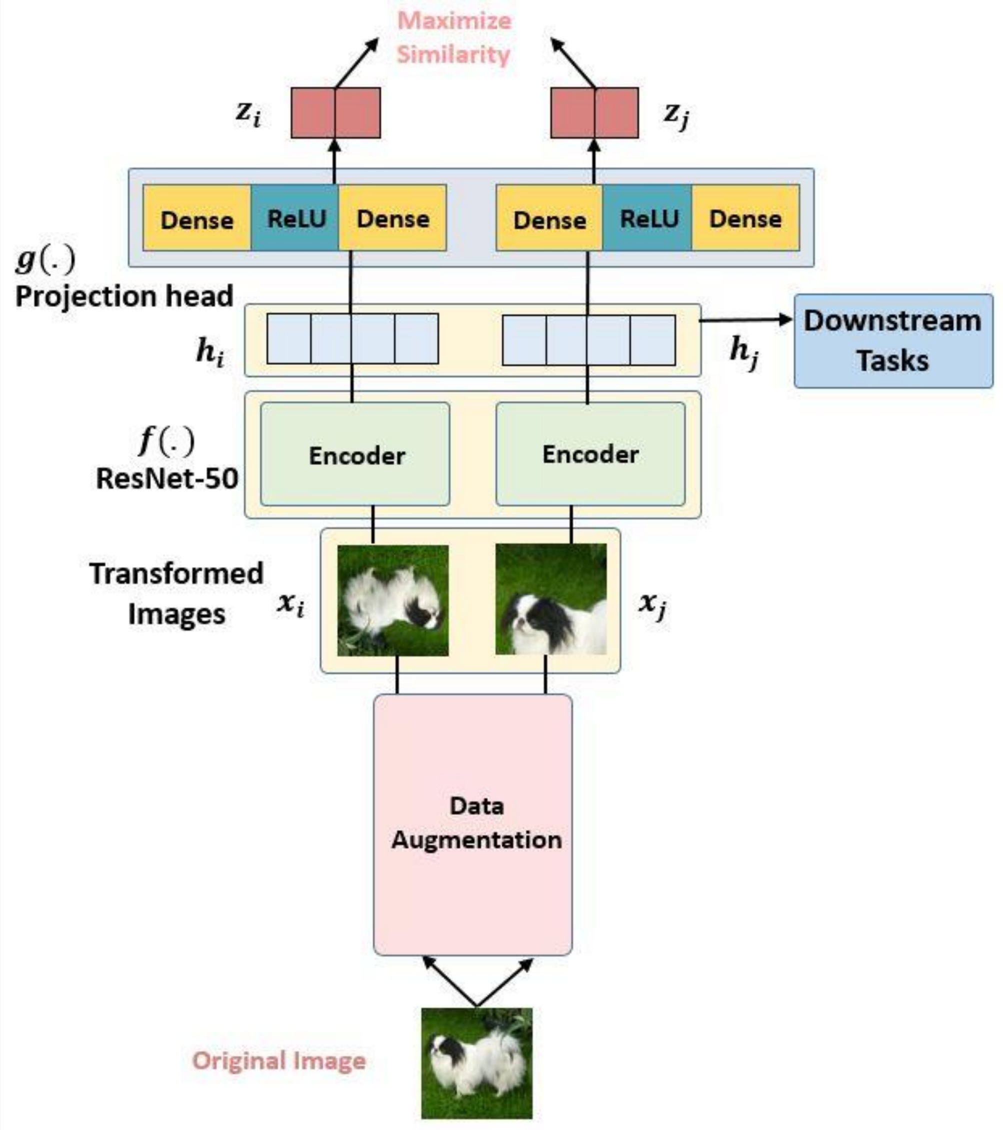


Self-supervised Learning

Contrastive

Contrastive - SimCLRv1

- Projection head



[Reference](#)

Self-supervised Learning

Contrastive

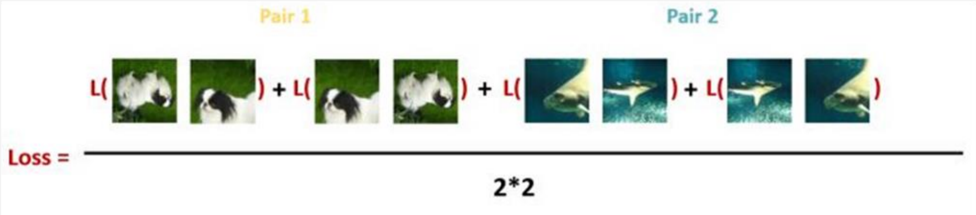
Contrastive - SimCLRv1

➢ NT-Xent loss function

$$s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|) \qquad \text{\# pairwise similarity}$$

$$\ell(i,j) \textbf{ as } \ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$$

$$\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$$



[Reference](#)

D

Bootstrapping

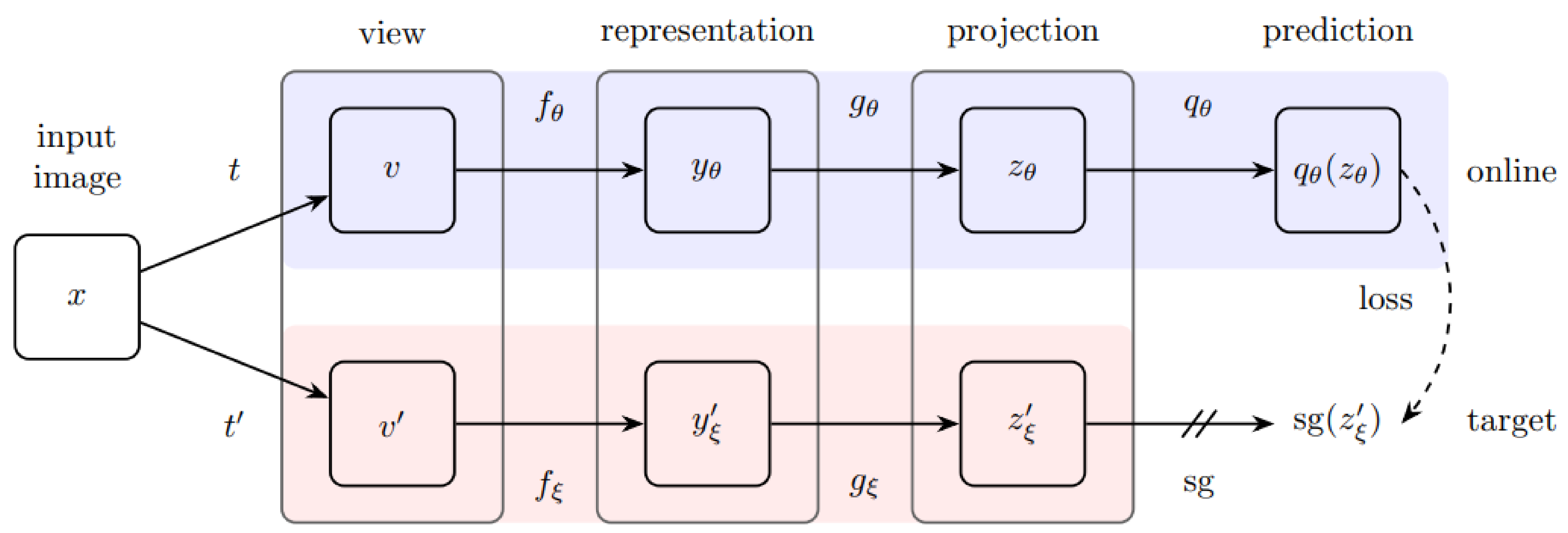
Bootstrapping

- In contrastive methods, the negative samples selection is a hard problem
  - The contribution of negative samples is to avoid model collapse
- How to training without negative samples?
  - BYOL
  - SimSiam

Self-supervised Learning

Bootstrapping

BYOL (Bootstrap your own latent)

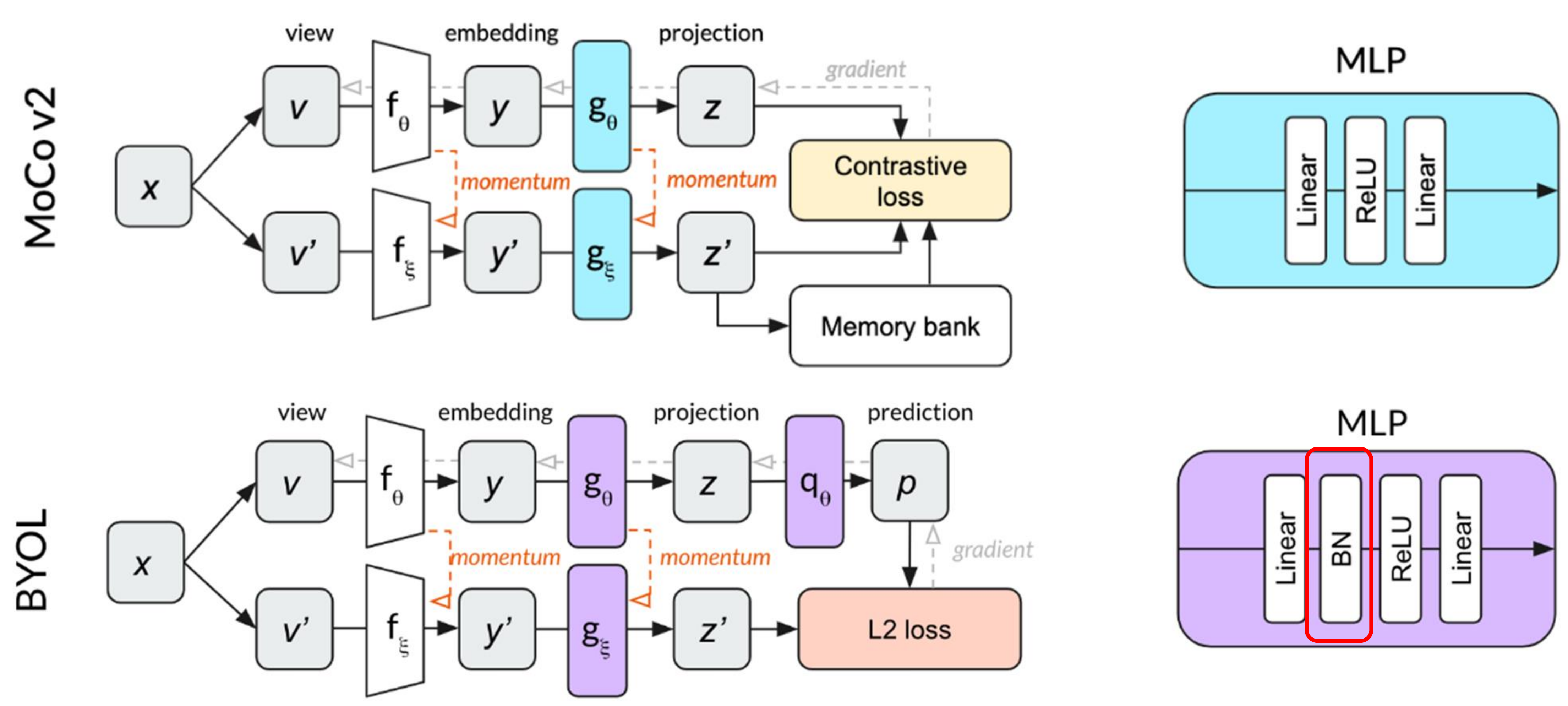


[Reference](#)

Self-supervised Learning

Bootstrapping

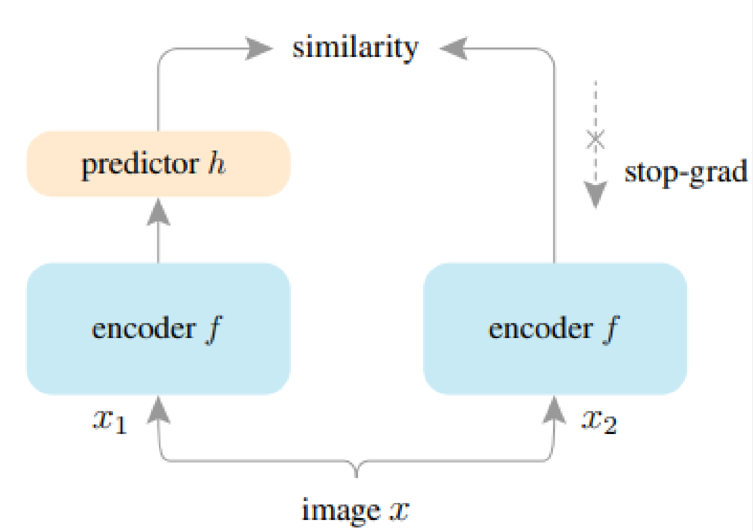
BYOL (Bootstrap your own latent)

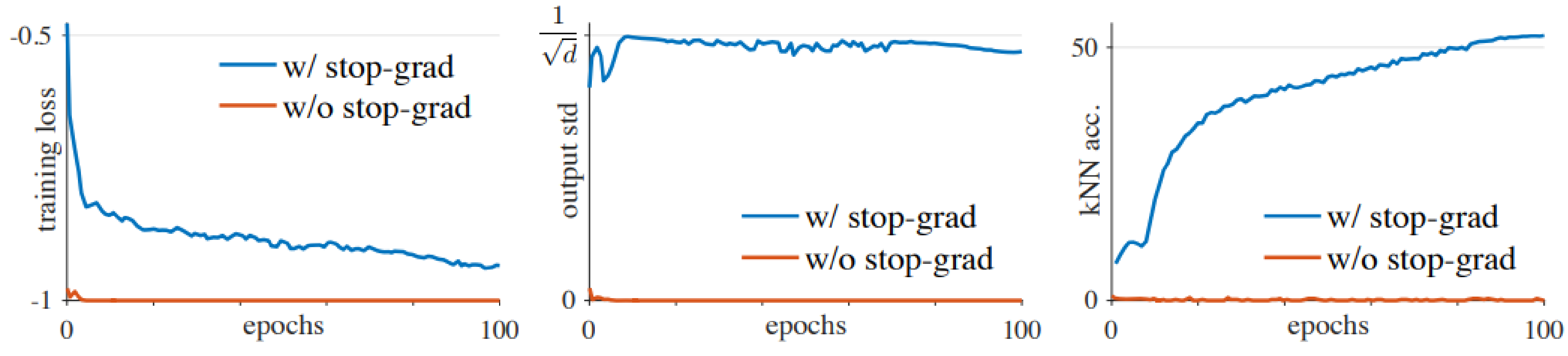


Reference

Self-supervised Learning
Bootstrapping

SimSiam (Simple siamese)





	acc. (%)
w/ stop-grad	67.7±0.1
w/o stop-grad	0.1

Reference

E

Regularization

Self-supervised Learning

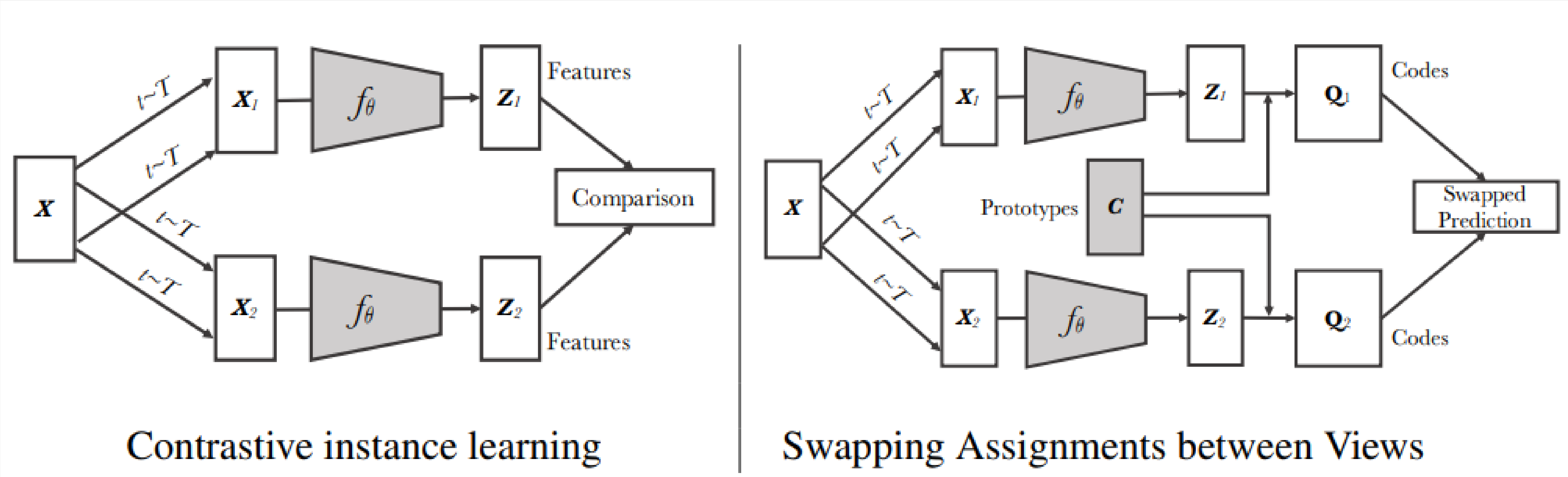
Regularization

Simple extra regularization

- ➢ Training also without negative samples
- ➢ Representation mining with regularization while training
  - SwAV
  - Barlow twins

Self-supervised Learning
Regularization

SwAV (Swapping assignments between views)

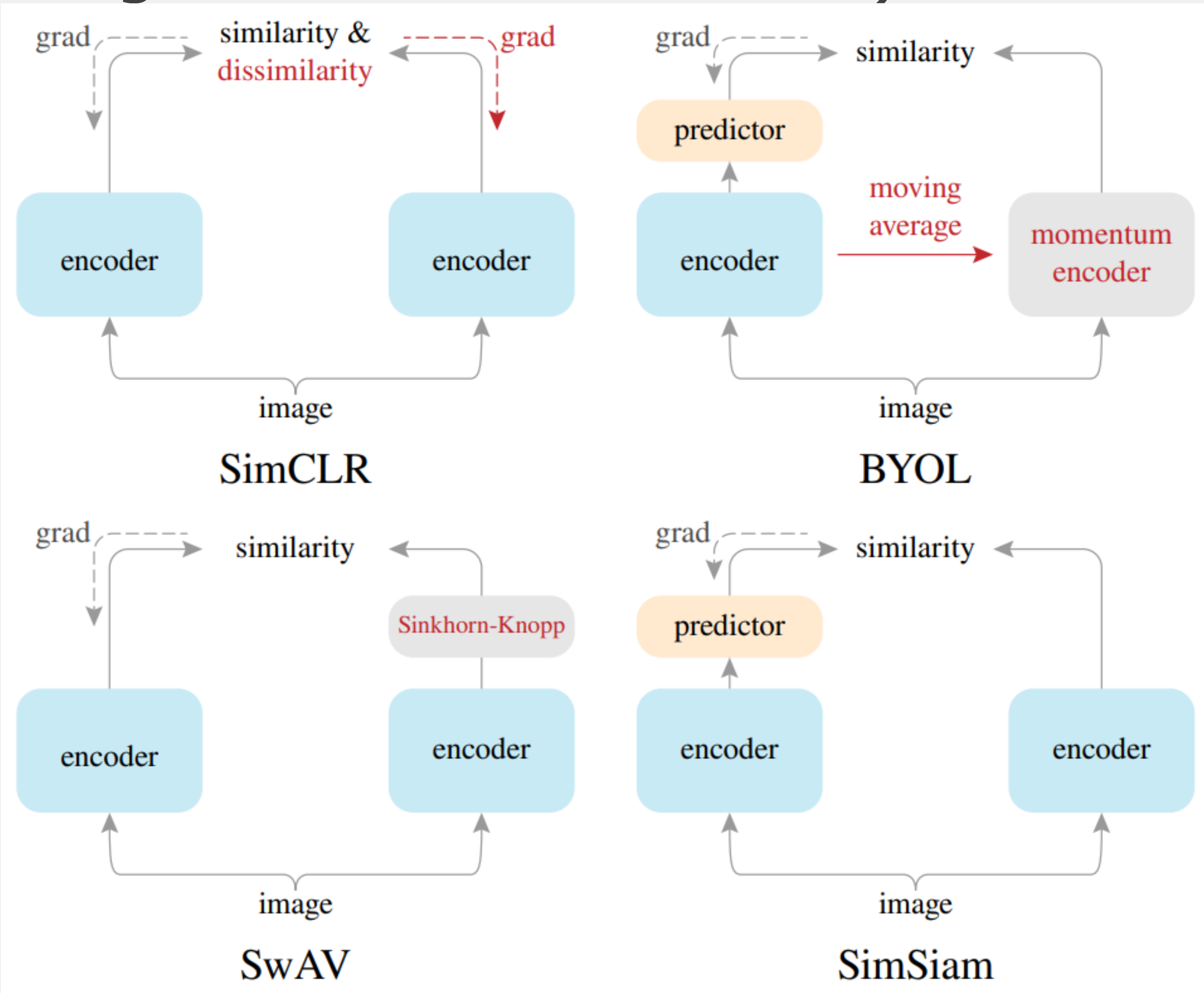


[Reference]

Self-supervised Learning

Regularization

SwAV (Swapping assignments between views)

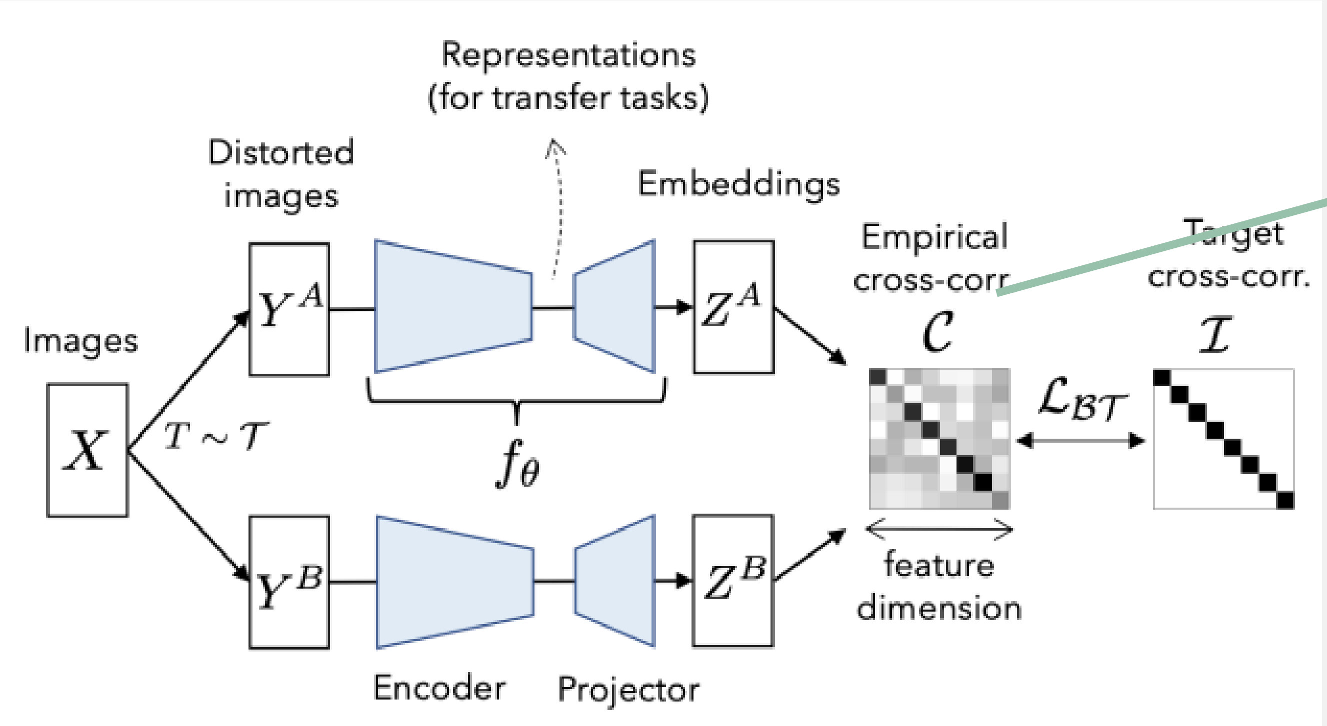


[Reference](#)

Self-supervised Learning

Regularization

Barlow Twins



$$\mathcal{C}_{ij} \triangleq \frac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{\sum_b \left(z^A_{b,i}\right)^2}\sqrt{\sum_b \left(z^B_{b,j}\right)^2}}$$

$$\mathcal{L}_{\mathcal{BT}} \triangleq \underbrace{\sum_i (1-\mathcal{C}_{ii})^2}_{\text{invariance term}} + \lambda \underbrace{\sum_i \sum_{j \neq i} {\mathcal{C}_{ij}}^2}_{\text{redundancy reduction term}}$$

[Reference](#)

05

Examples

Examples

Supervised Learning

Link

Unsupervised Learning

Link

Semi-supervised Learning

Link