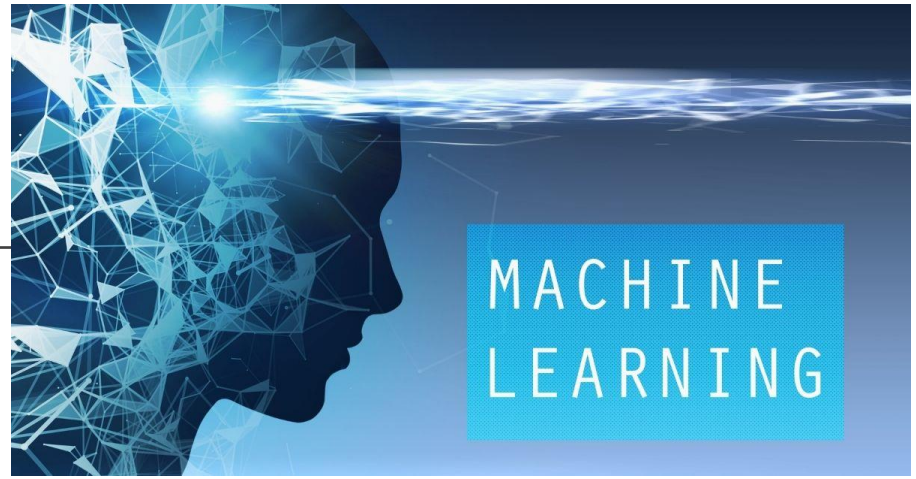
What is Machine Learning?



DEEP LEARNING COURSE – 2023

E. FATEMIZADEH

Definitions

- Artificial Intelligence (AI)
- Machine Learning (ML)
- Deep Learning (DL)

Artificial Intelligence (AI)

- AI research has been defined as the field of study of *intelligent agents*, which refers to any system that perceives its *environment* and takes actions that *maximize* its chance of achieving its *goals*.
- Previous and rejected definition: Machines that mimic and display "human" cognitive skills that are associated with the human mind, such as "learning" and "problem-solving".
- Rejected by major AI researchers who now describe AI in terms of *rationality* and acting *rationally*, which does not limit how intelligence can be articulated.

Machine Learning (ML)

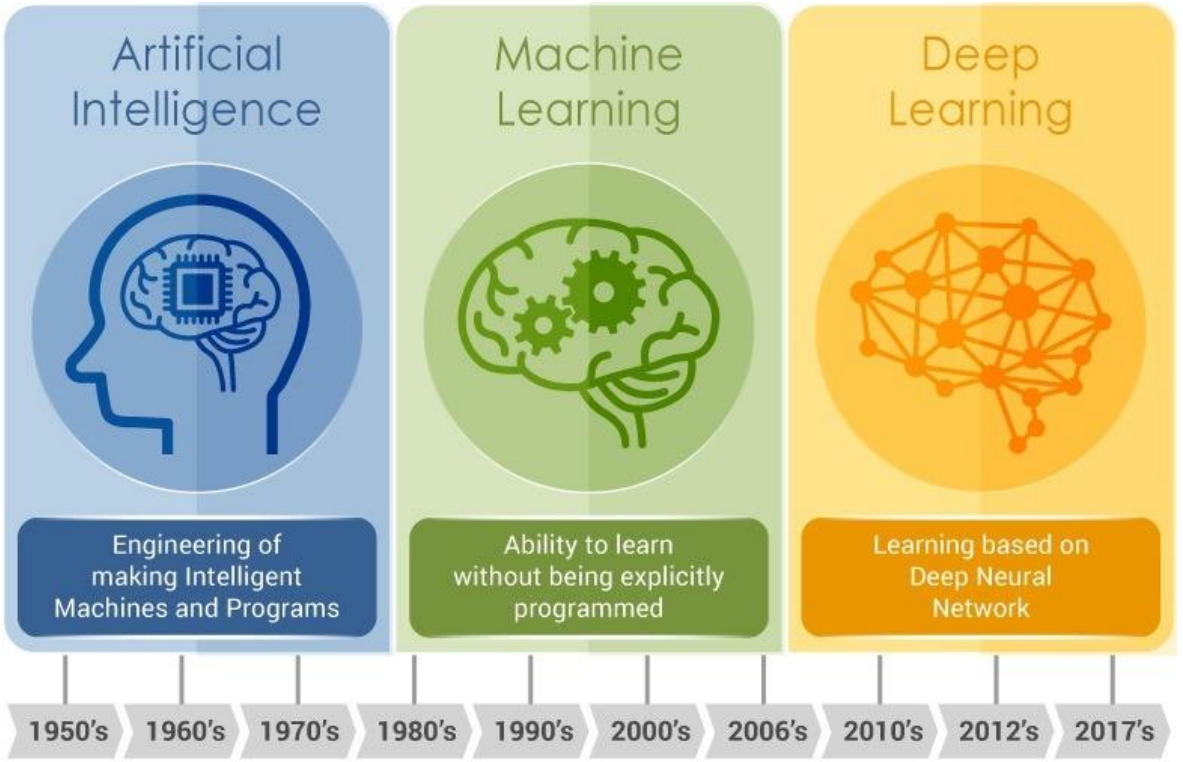
- Machine learning (ML) is the scientific study of *algorithms* and *statistical models* that computer systems use to perform a specific task without using explicit instructions, relying on *patterns* and *inference* instead. Learn from *data*.
- How?
  - Build a *mathematical model* based on sample data, known as "*training data*",

Deep Learning (DL)

- Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning.
- Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains.

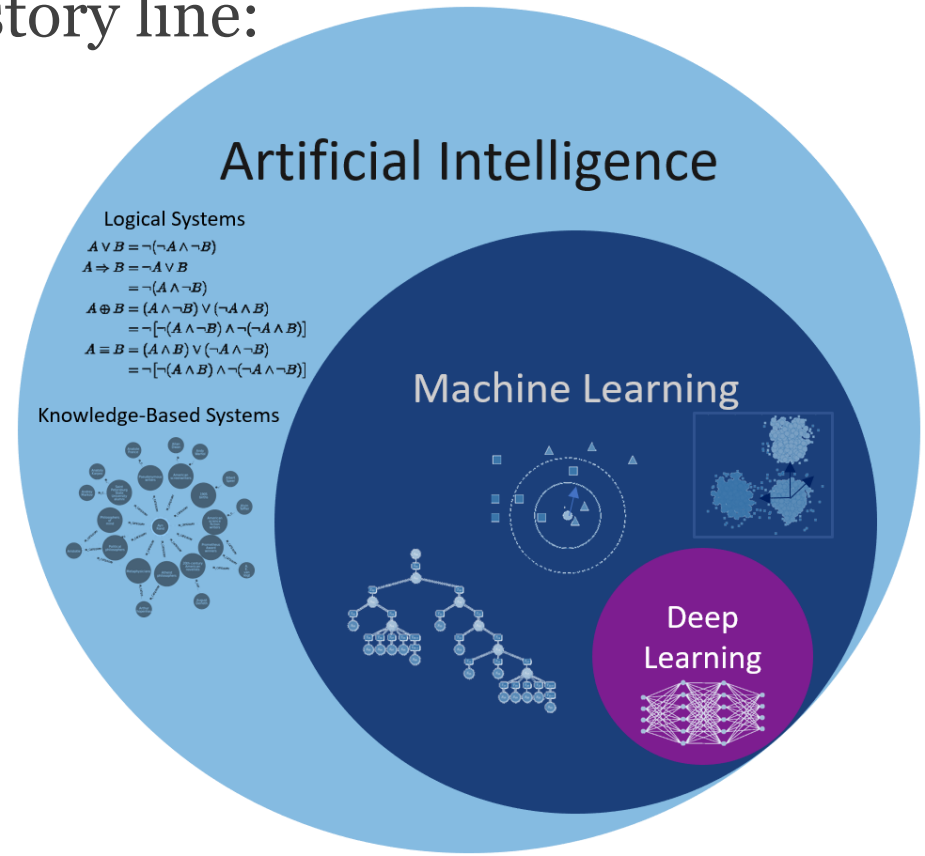
How Related?

- AI, ML, and DL in one frame:



Historical Evolving

- AI, ML, and DL history line:

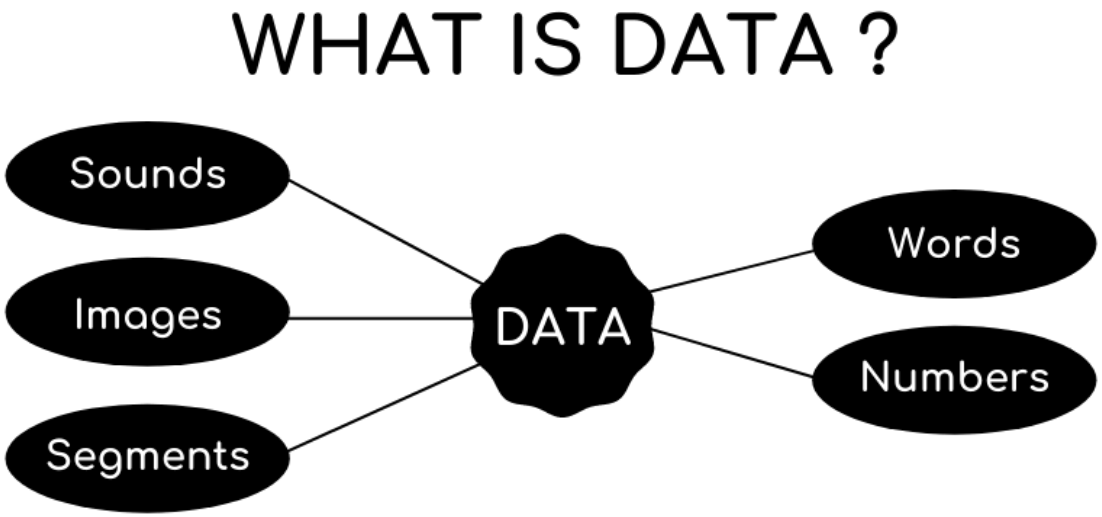


Related Area

- Data Mining,
- Pattern Recognition,
- Computational Intelligence,
- Computational Statistics,
- Statistical Learning,
- Expert System,
- Fuzzy Inference,
- …

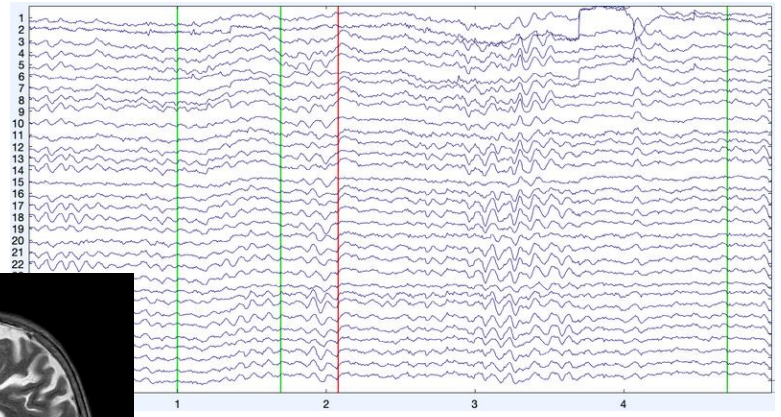
Data

- *Data* is *information* that has been translated into a form that is efficient for processing.



Data

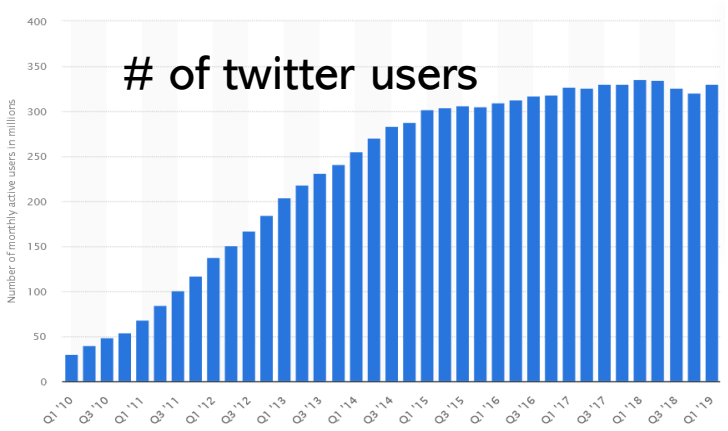
- Examples:

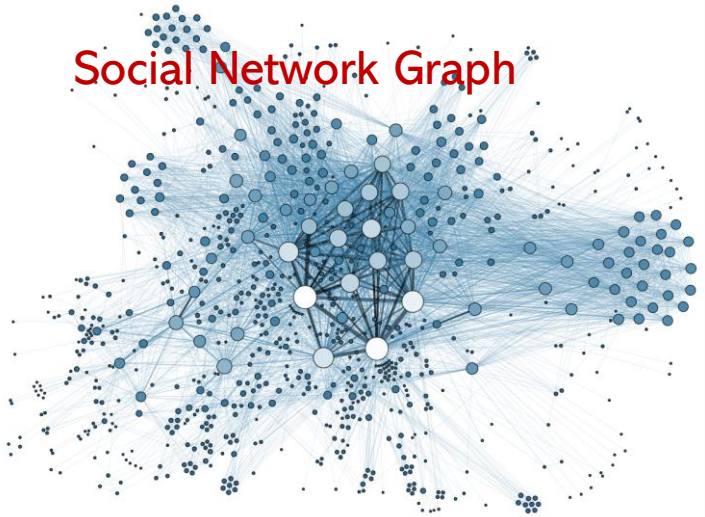


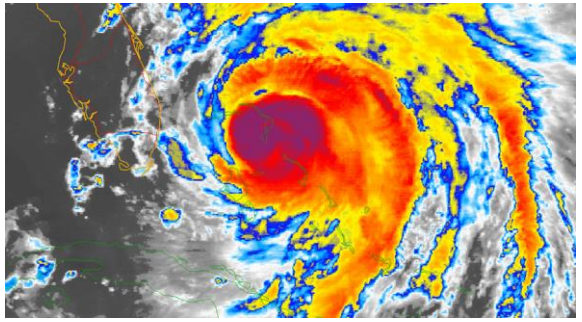
Text as a means of communication evolved to store and convey information over long distances. It is the written representation of any speech communication.

Data

- Examples:

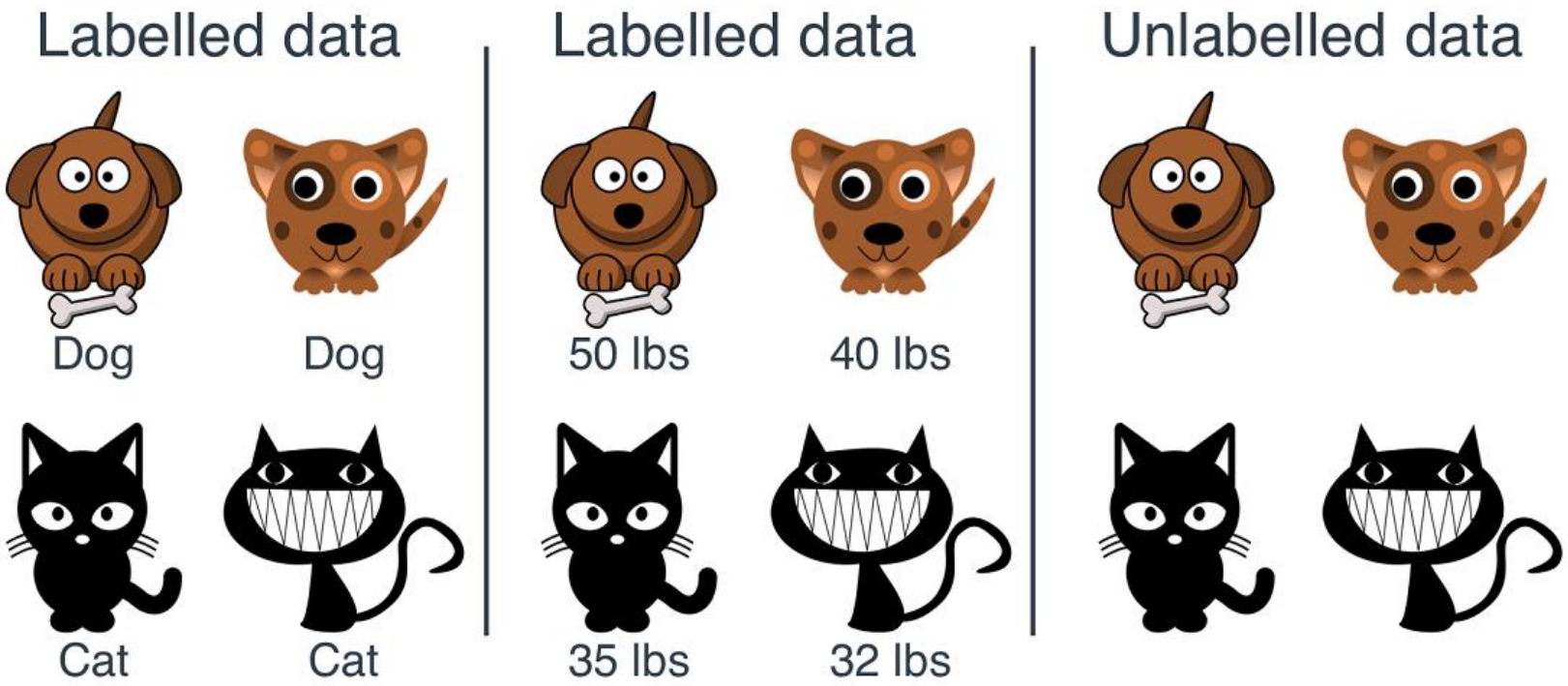






Data

- Labelled (single or multiple) and unlabeled data:

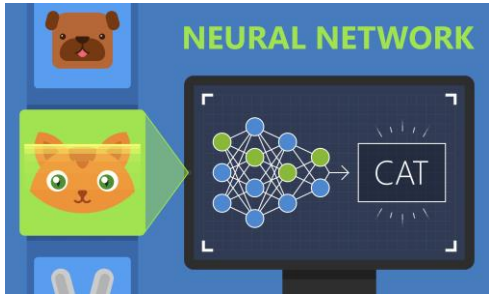


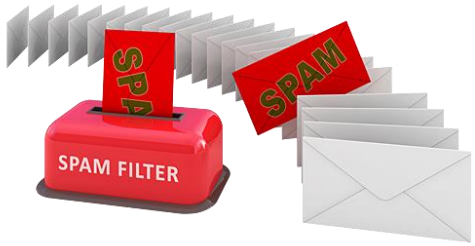
Major Tasks

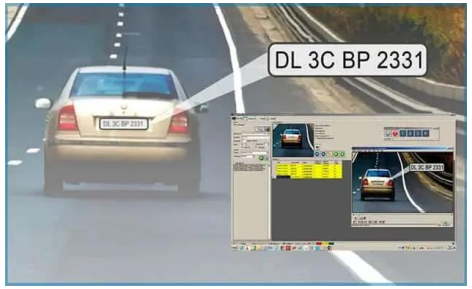
- Supervised Learning,
- Unsupervised Learning,
- Reinforcement learning,
- Semi-Supervised Learning
- Others (Active learning, Vector Quantization, …)

Supervised Learning

- Learn model from both input(s) and output(s) data
- Input(s): Object Properties
- Output(s): Discrete or “Limited Set” (e.g. Labels) or Continuous

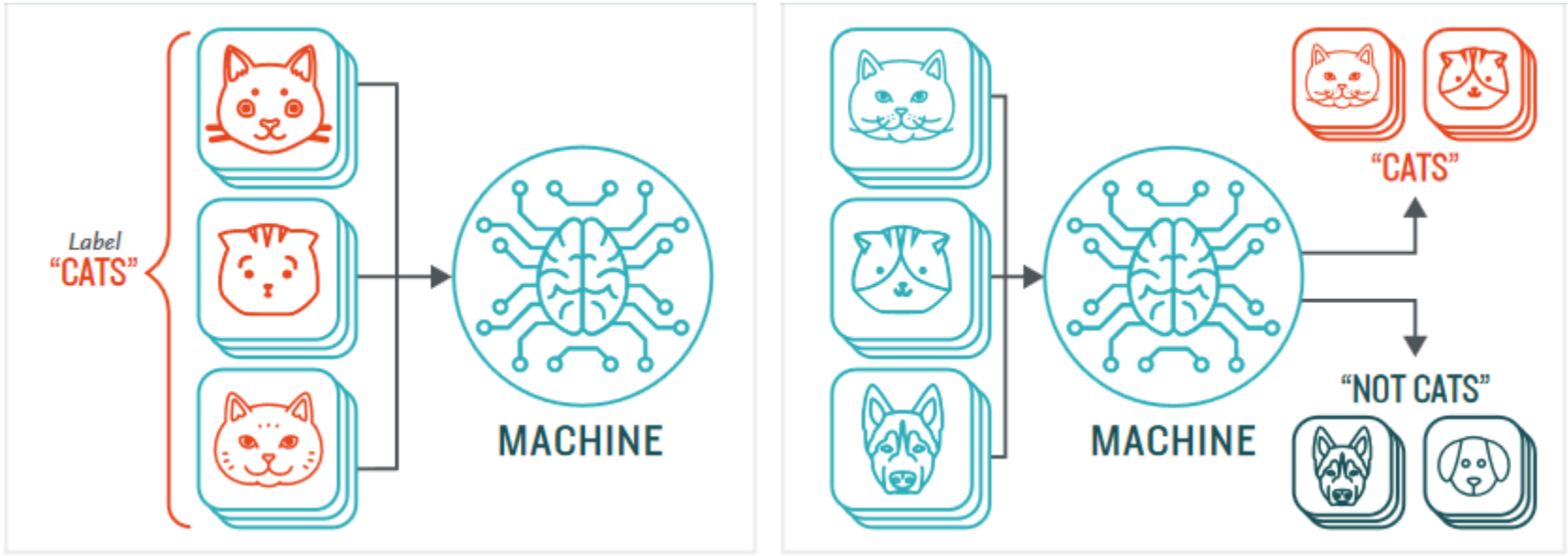






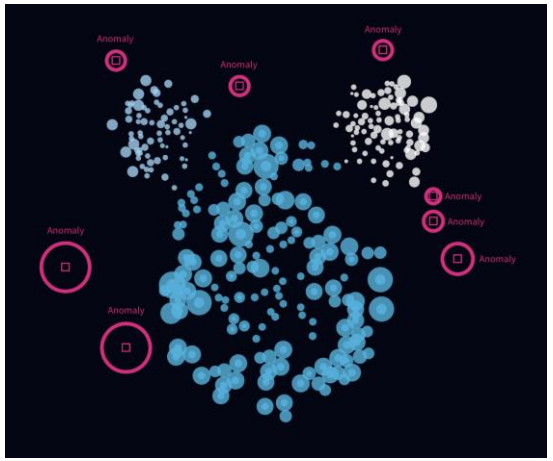
Supervised Learning

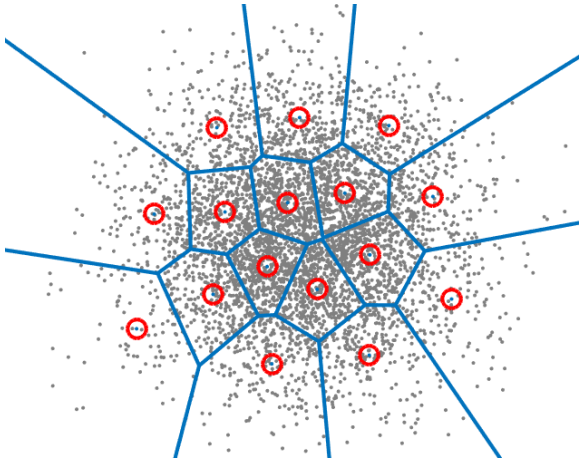
- *Tune* (left) machine and *use* (right) it



Unsupervised Learning

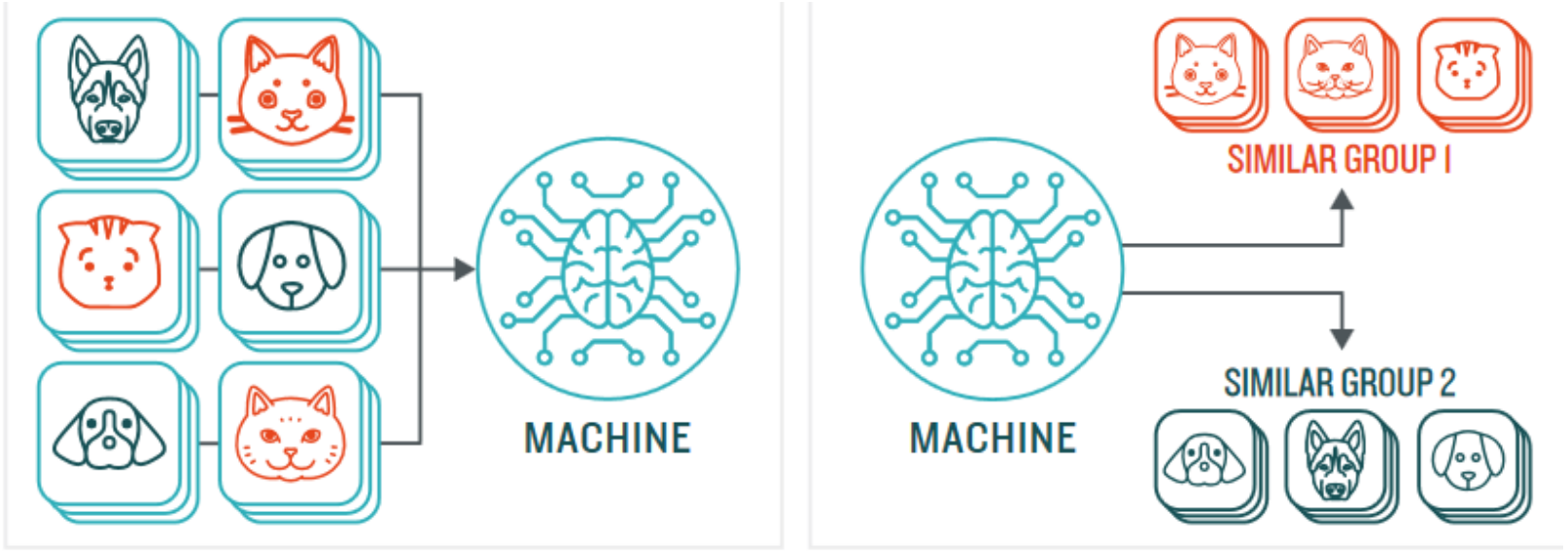
- No desired output!
- Find structure of data/Clustering/Grouping/…
- Applications: Segmentation, anomaly detection, data reduction, and …





Unsupervised Learning

- Group similar data

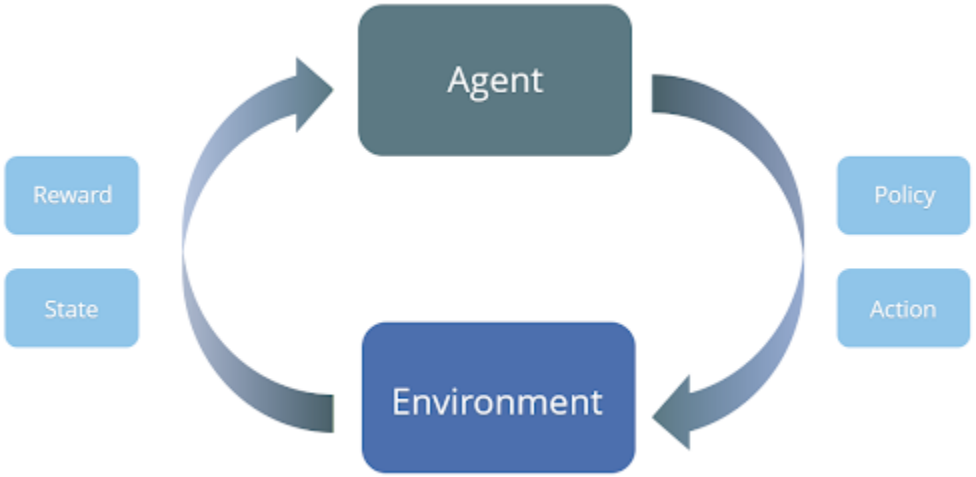


Semi-Supervised Learning

- Semi-supervised learning is an approach to machine learning that combines a *small* amount of *labeled data* with a large amount of unlabeled data during training.
- Semi-supervised learning falls between unsupervised learning (with no labeled training data) and supervised learning (with only labeled training data). It is a special instance of *weak supervision*.

Reinforcement Learning

- Reinforcement learning is a machine learning training method based on *rewarding* desired behaviors and/or *punishing* undesired ones.
- In general, a reinforcement learning agent is able to perceive and interpret its environment, take actions and learn through trial and error.

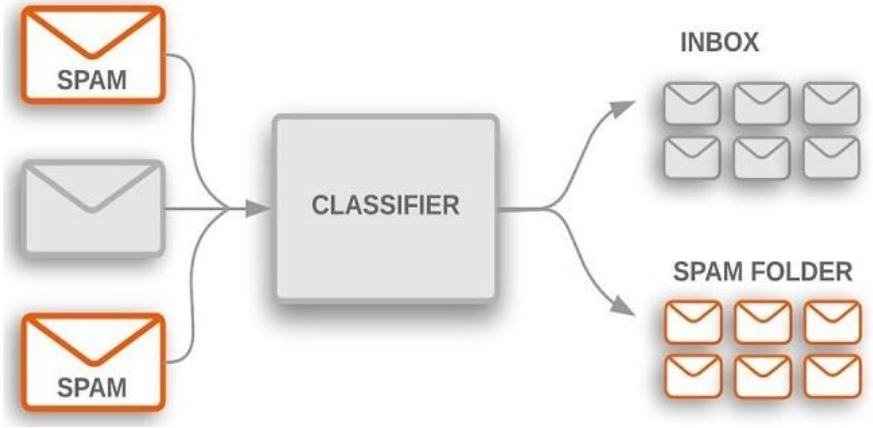


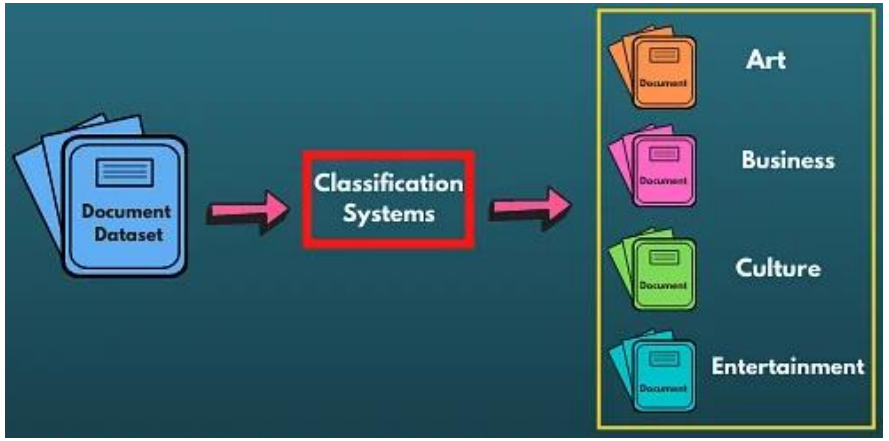
Our Focus

- Supervised Learning:
  - Classification
  - Regression
- Unsupervised Learning:
  - Distribution Estimation and Generation

Classification

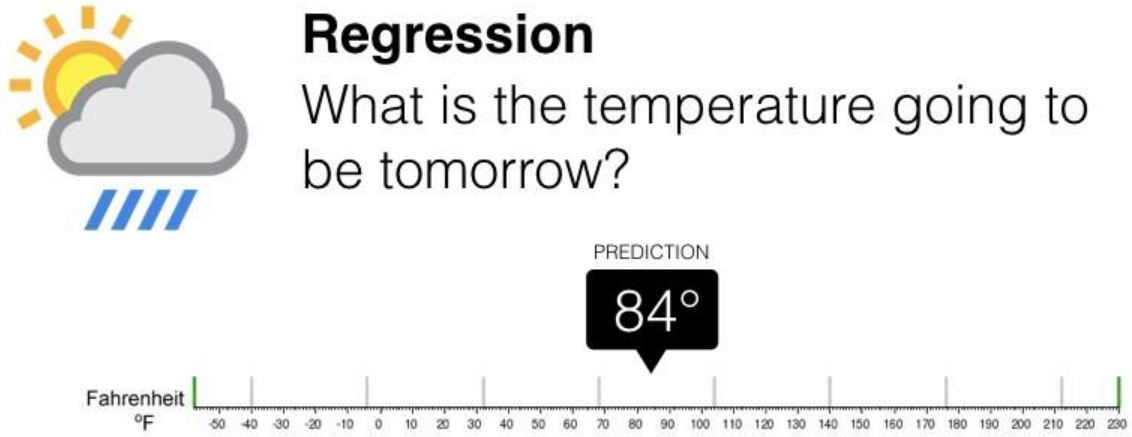
- Input(s) are objects: Electric Signals/Image/Video/Speech/Text/DNA sequences/Email/….
- Output(s) are binary (mostly Labels): “0/1”, “benign”/”malignant” (Tumor), “Primary”, “Social”, “Promotion” (Gmail),





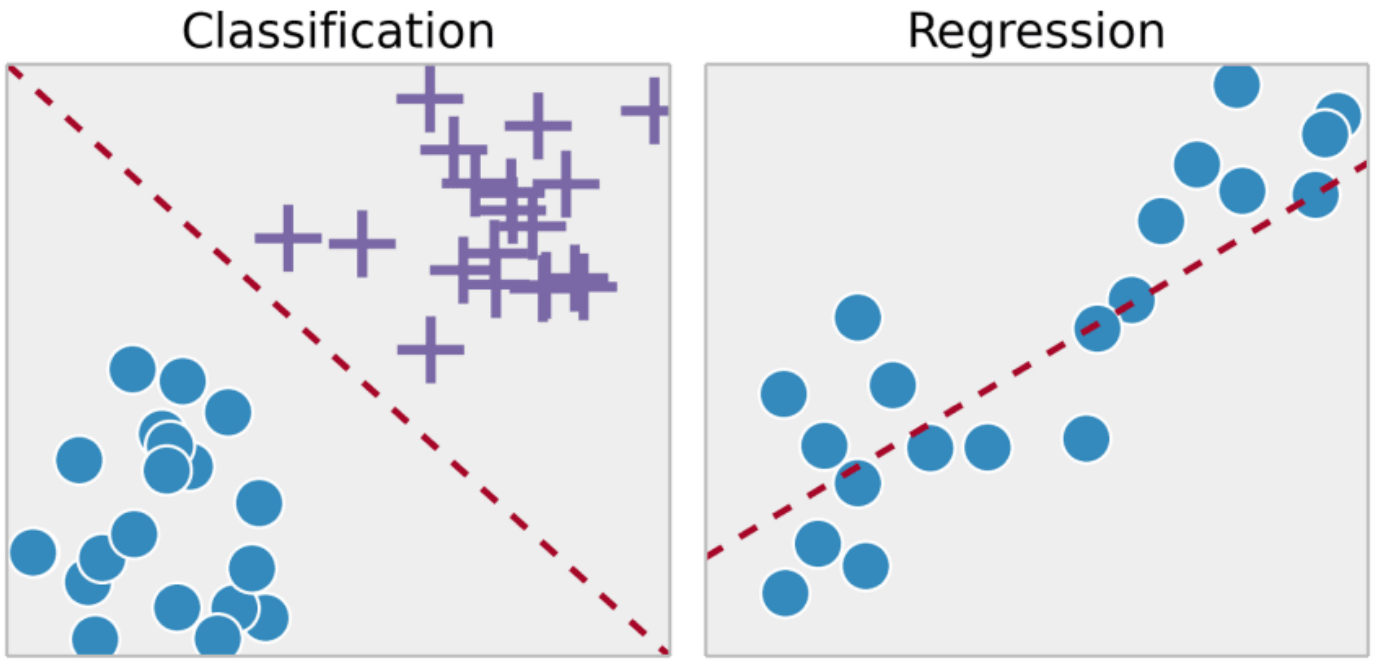
Regression

- Input(s) are objects: Electric Signals/Image/Video/Speech/Text/DNA sequences/Email/….
- Output(s) are Continuous: Temperature, Score, Fuel Consumption, …



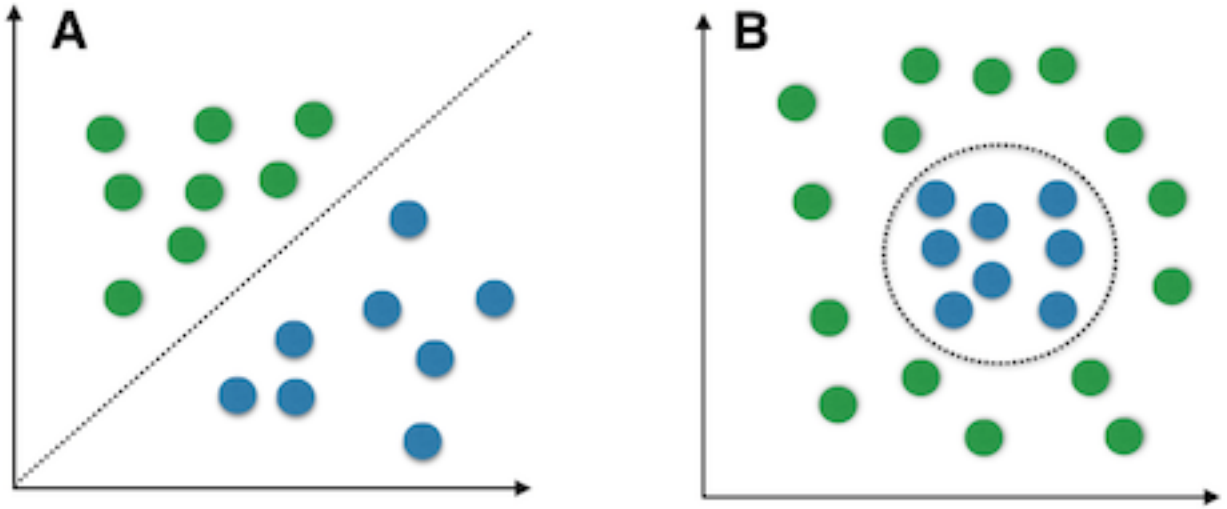
Classification vs Regression

- Mathematically:



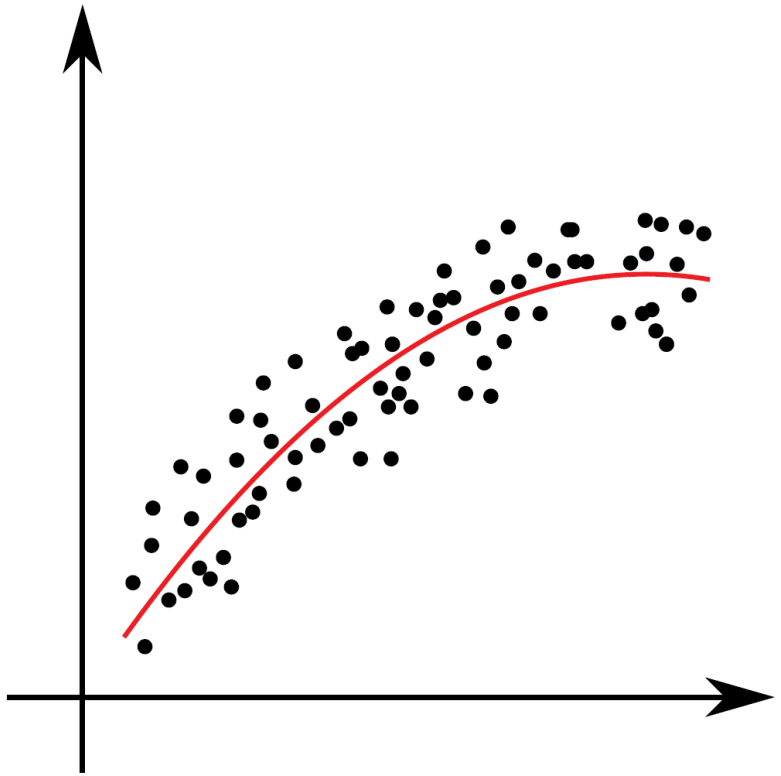
Two Types of Classifier

- Linear vs Non-Linear”



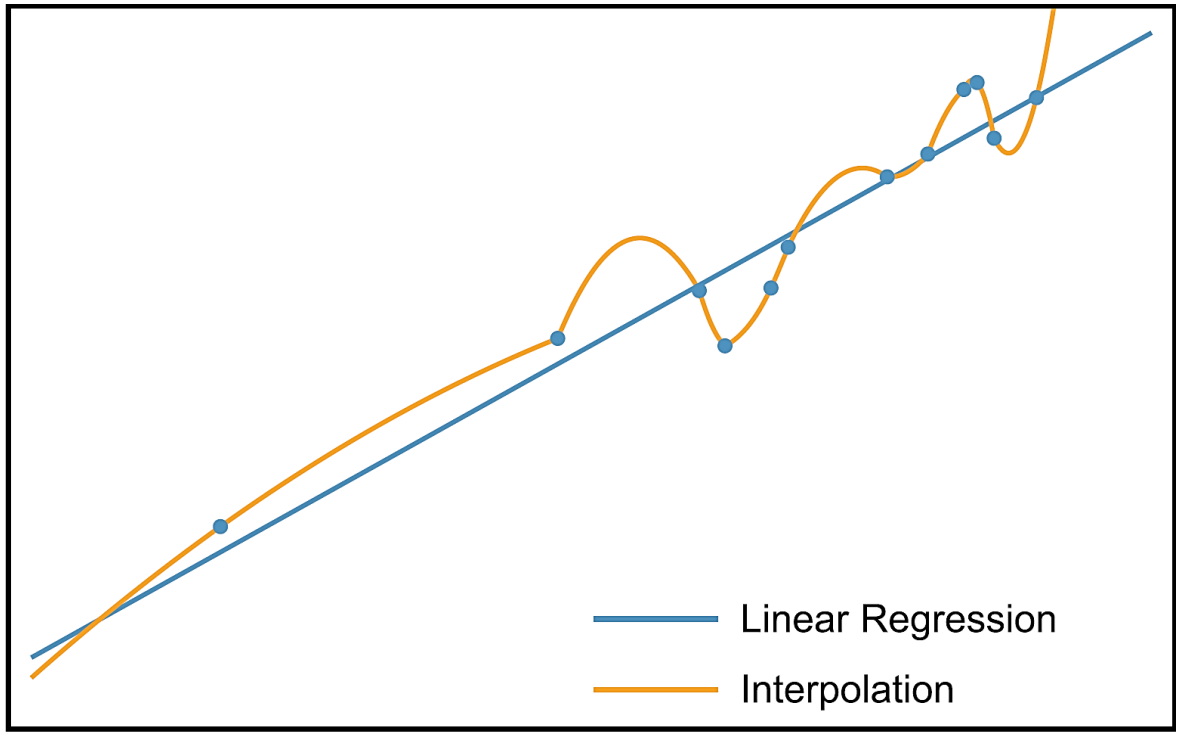
Regression

- Illustration



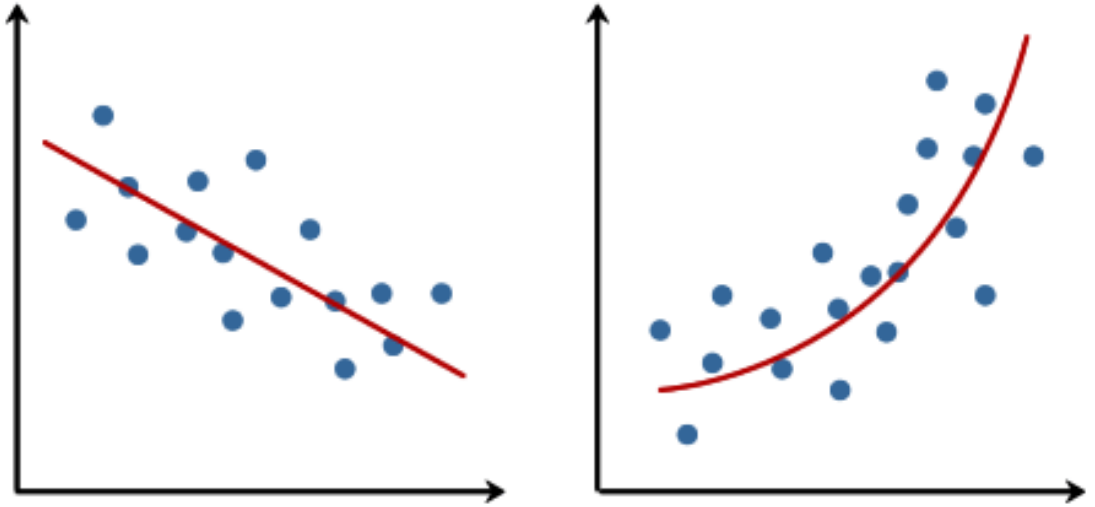
Regression vs Interpolation

- Illustration



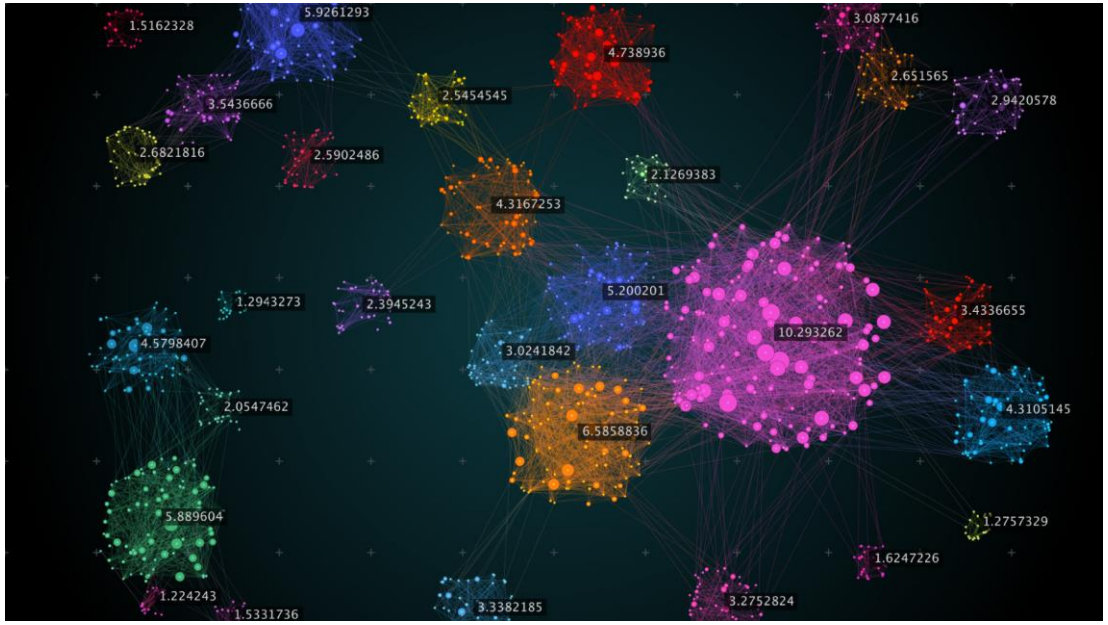
Two Types of Regression

- Linear vs Non-Linear”



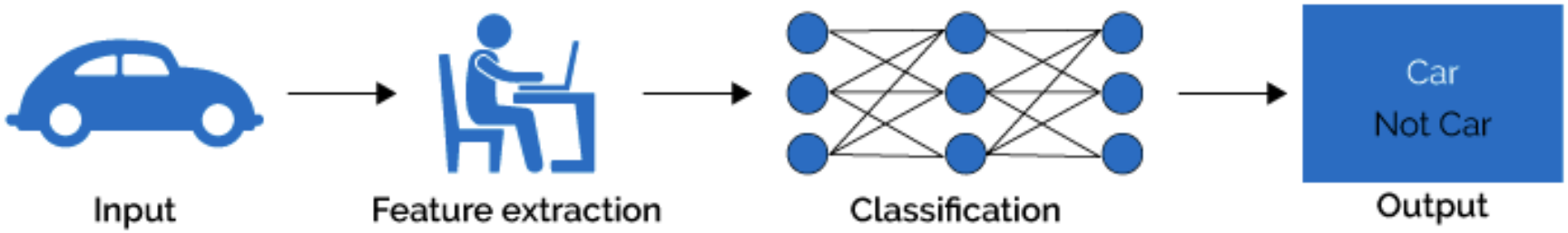
Unsupervised Learning

- Input(s) are objects:
  - Electric Signals/Image/Speech/DNA sequences/Email/….
- No Output data!



Classical Machine Learning

- A general block diagram:



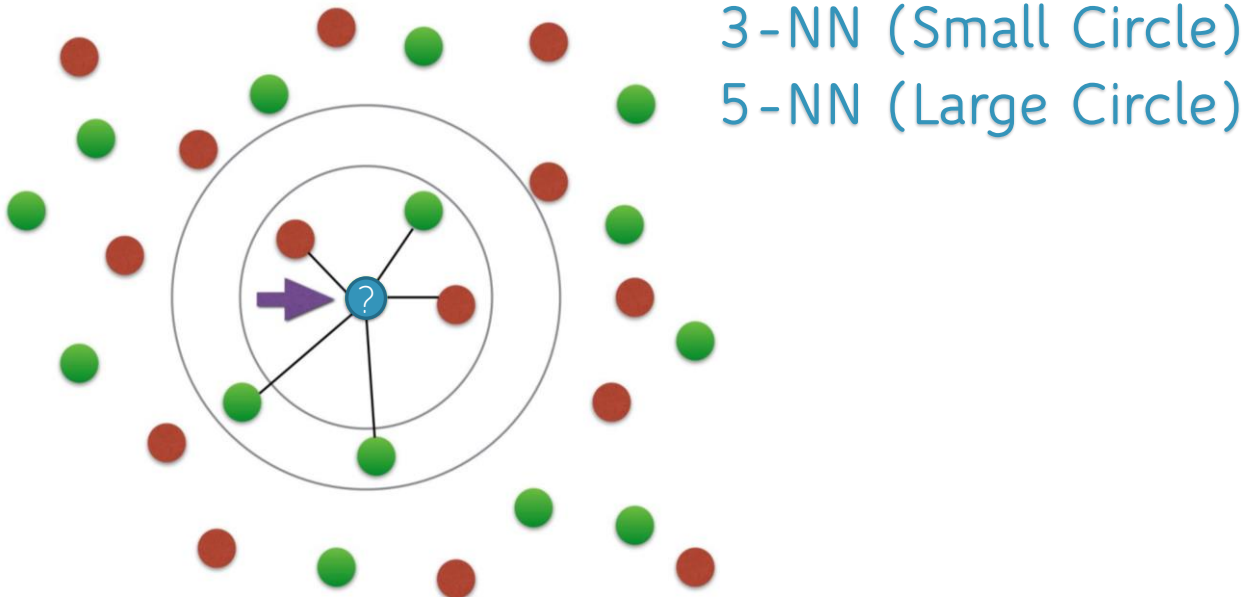
- Feature Generation/Extraction: $objects \rightsquigarrow x \in \mathbb{R}^D$
- Feature Reduction: $x \in \mathbb{R}^D \Longrightarrow y \in \mathbb{R}^d, \;\; d \ll D$

Supervised Learning Models

- Most Known Models:
  - K-Nearest Neighbor (K-NN)
  - Bayes and Naïve Bayes Classifier
  - Logistic Discrimination (Regression)
  - Decision Tree and Random Forest
  - Support Vector Machine (SVM)
  - Artificial Neural Networks (Shallow or Deep)

K-Nearest Neighbor (K-NN)

- Non-Parametric Classifier/Regressor
- An object is classified by a *plurality vote* of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.
- Needs:
  - K: a positive integer,
  - A distance meter.
- Both classifier and regressor



Bayes Classifier (Optimal)

- For two-Classes $\{\Omega_1 \; and \; \Omega_2\}$ with available posterior probabilities:

$$\begin{cases} \boldsymbol{x} \in \Omega_1, & p(\Omega_1|\boldsymbol{x}) > p(\Omega_2|\boldsymbol{x}) \\ \boldsymbol{x} \in \Omega_2, & p(\Omega_1|\boldsymbol{x}) < p(\Omega_2|\boldsymbol{x}) \end{cases}$$

- Using Bayes theorem:

$$P(\Omega_j \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \Omega_j)P(\Omega_j)}{p(\boldsymbol{x})}$$

- We have:

$$\begin{cases} \boldsymbol{x} \in \Omega_1, & P(\Omega_1)p(\boldsymbol{x}|\Omega_1) > P(\Omega_2)p(\boldsymbol{x}|\Omega_2) \\ \boldsymbol{x} \in \Omega_2, & P(\Omega_1)p(\boldsymbol{x}|\Omega_1) < P(\Omega_2)p(\boldsymbol{x}|\Omega_2) \end{cases}$$

- Tie situation is possible: $p(\Omega_1|\boldsymbol{x}) = p(\Omega_2|\boldsymbol{x})$

Bayes Classifier (Optimal)

- Using Likelihood ratio function

$$\begin{cases} \boldsymbol{x} \in \Omega_1, & ln\left(\frac{p(\Omega_1|\boldsymbol{x})}{p(\Omega_2|\boldsymbol{x})}\right) > 0 \\ \boldsymbol{x} \in \Omega_2, & ln\left(\frac{p(\Omega_1|\boldsymbol{x})}{p(\Omega_2|\boldsymbol{x})}\right) < 0 \end{cases}$$

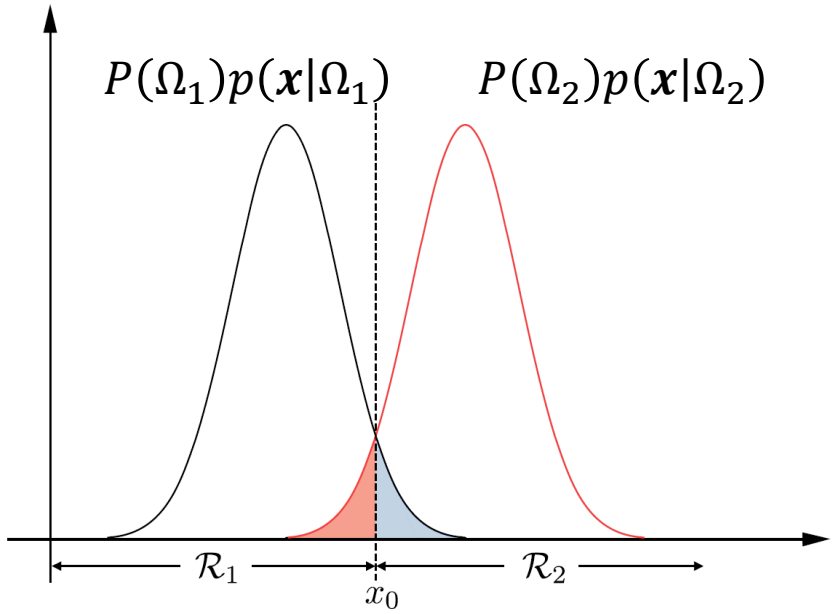
- or:

$$\begin{cases} \boldsymbol{x} \in \Omega_1, & ln\left(\frac{P(\Omega_1)p(\boldsymbol{x}|\Omega_1)}{P(\Omega_2)p(\boldsymbol{x}|\Omega_2)}\right) > 0 \\ \boldsymbol{x} \in \Omega_2, & ln\left(\frac{P(\Omega_1)p(\boldsymbol{x}|\Omega_1)}{P(\Omega_2)p(\boldsymbol{x}|\Omega_2)}\right) < 0 \end{cases}$$

Bayes Classifier (Optimal)

- Bayes classifier minimize total statistical error:

$$\int_{\mathcal{R}_1} P(\Omega_2)p(\boldsymbol{x}|\Omega_2)dx + \int_{\mathcal{R}_2} P(\Omega_1)p(\boldsymbol{x}|\Omega_1)dx$$



Bayes Classifier (Optimal)

- For M-Classes problems:

$$\Omega_i = \arg\max_{\Omega_j} P(\Omega_j)P(\boldsymbol{x}|\Omega_j),\ j = 1,2,\dots,M$$

- Requirements:
  - Class prior probabilities: $\{P(\Omega_j)\}_{j=1}^{M}$, prior knowledge or $P(\Omega_j) \approx \frac{N_j}{N}$
  - Class conditional probability density functions (likelihood): $\{P(\boldsymbol{x}|\Omega_j)\}_{j=1}^{M}$:
    - Parametric (Maximum Likelihood)
    - Non-Parametric (Parzen Window)
  - Curse of dimensionality (# of data for pdf estimation in $\mathbb{R}^D$ is order of N^D)

Naïve Bayes Classifier

- Assume independent feature vector:

$$\boldsymbol{x} = (x_1 \quad x_2 \quad \dots \quad x_n)^T$$

$$p(\boldsymbol{x}|\Omega_i) \cong p(x_1|\Omega_i)p(x_2|\Omega_i) \dots p(x_n|\Omega_i)$$

Logistic Regression (Discrimination)

- Based on Bayes Classifier.
- Model (estimate) the class conditional pdfs such that (M=2):

$$LR(\boldsymbol{x}) = ln\left(\frac{p(\Omega_1|\boldsymbol{x})}{p(\Omega_2|\boldsymbol{x})}\right) = ln\left(\frac{P(\Omega_1)p(\boldsymbol{x}|\Omega_1)}{P(\Omega_2)p(\boldsymbol{x}|\Omega_2)}\right) = \boldsymbol{\omega}^T\boldsymbol{x} + \omega_0$$

- It can be shown (M=2):

$$p(\Omega_1|\boldsymbol{x}) = \frac{e^{(\boldsymbol{\omega}^T\boldsymbol{x}+\omega_0)}}{1+e^{(\boldsymbol{\omega}^T\boldsymbol{x}+\omega_0)}} = \frac{1}{1+e^{-(\boldsymbol{\omega}^T\boldsymbol{x}+\omega_0)}} = \sigma(\boldsymbol{\omega}^T\boldsymbol{x}+\omega_0), \qquad \sigma(s) = \frac{1}{1+e^{-s}}$$

$$p(\Omega_2|\boldsymbol{x}) = \frac{1}{1+e^{(\boldsymbol{\omega}^T\boldsymbol{x}+\omega_0)}} = \frac{e^{-(\boldsymbol{\omega}^T\boldsymbol{x}+\omega_0)}}{1+e^{-(\boldsymbol{\omega}^T\boldsymbol{x}+\omega_0)}} = 1-\sigma(\boldsymbol{\omega}^T\boldsymbol{x}+\omega_0)$$

- $\sigma(s)$ is known as *logistic* function

Logistic Regression (Discrimination)

- Model the class conditional pdfs such that (M>2):

$$LR = ln\left(\frac{p(\Omega_i|\boldsymbol{x})}{p(\Omega_M|\boldsymbol{x})}\right) = ln\left(\frac{P(\Omega_i)p(\boldsymbol{x}|\Omega_i)}{P(\Omega_M)p(\boldsymbol{x}|\Omega_M)}\right) = \boldsymbol{\omega}_i^T\boldsymbol{x} + \omega_{i,0}, \qquad i = 1, 2, \dots, M-1$$

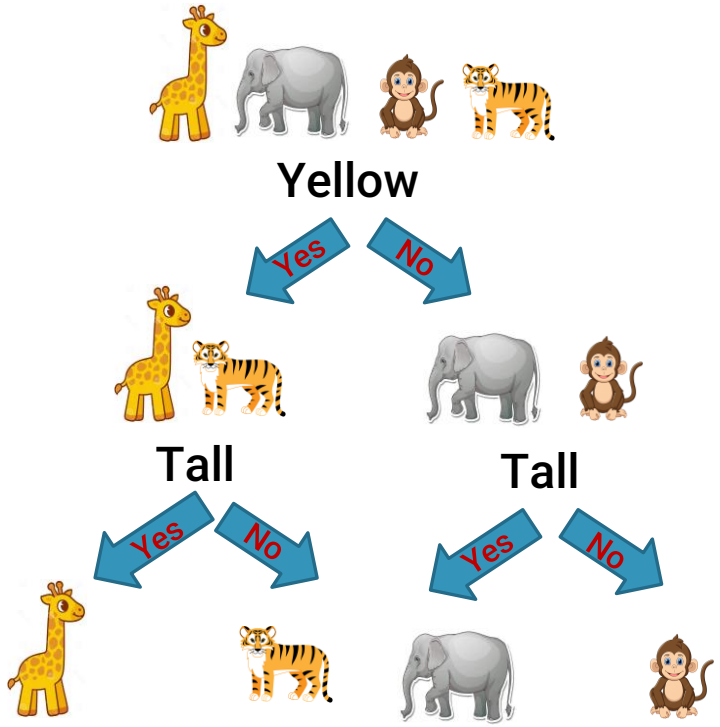
- It can be shown:

$$p(\Omega_i|\boldsymbol{x}) = \frac{e^{(\boldsymbol{\omega}_i^T\boldsymbol{x}+\omega_{i,0})}}{1 + \sum_{i=1}^{M-1} e^{(\boldsymbol{\omega}_i^T\boldsymbol{x}+\omega_{i,0})}}, i = 1,2, \dots, M-1$$

$$p(\Omega_M|\boldsymbol{x}) = \frac{1}{1 + \sum_{i=1}^{M-1} e^{(\boldsymbol{\omega}_i^T\boldsymbol{x}+\omega_{i,0})}}$$

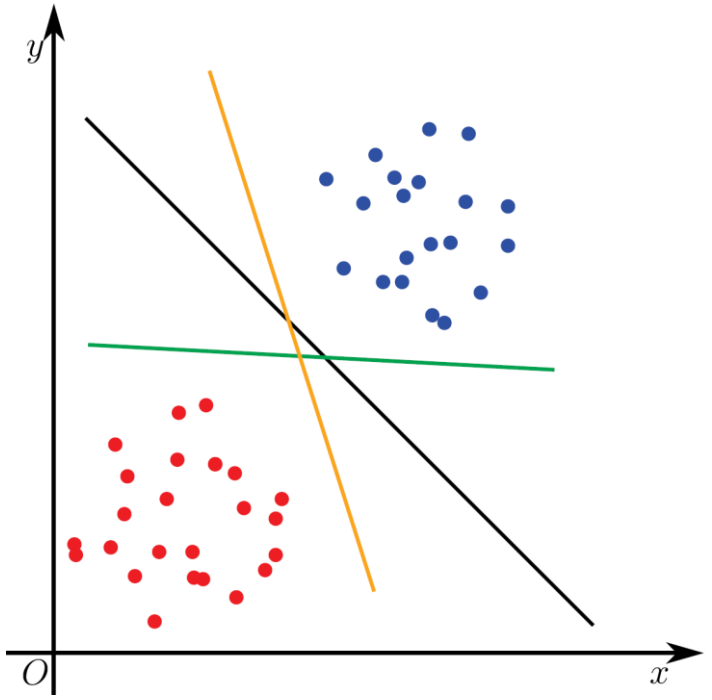
Decision Tree

- A Huge set of if-then rules in tree-like structure, which build systematically!



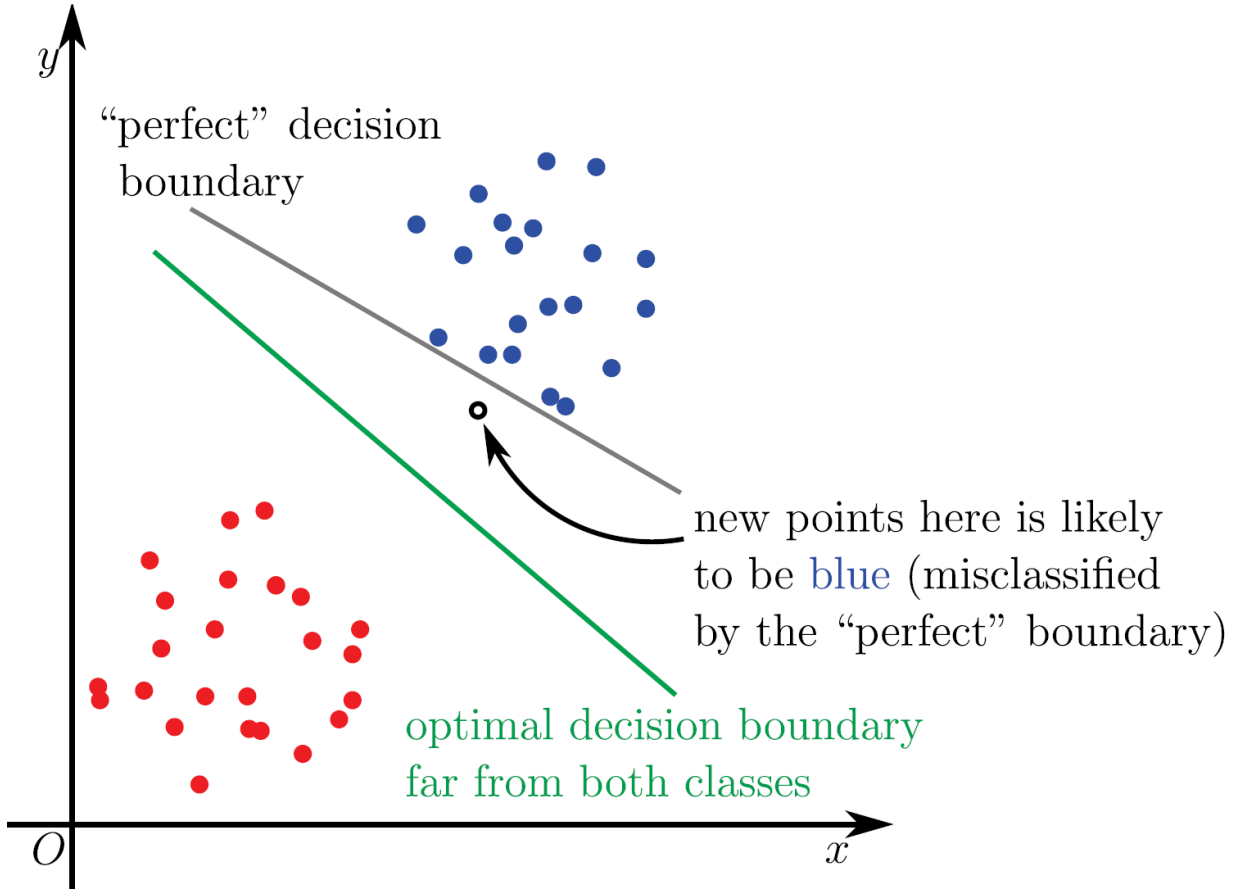
Support Vector Machines (SVM)

- Motivation:
- Linear separable problems has infinite solution



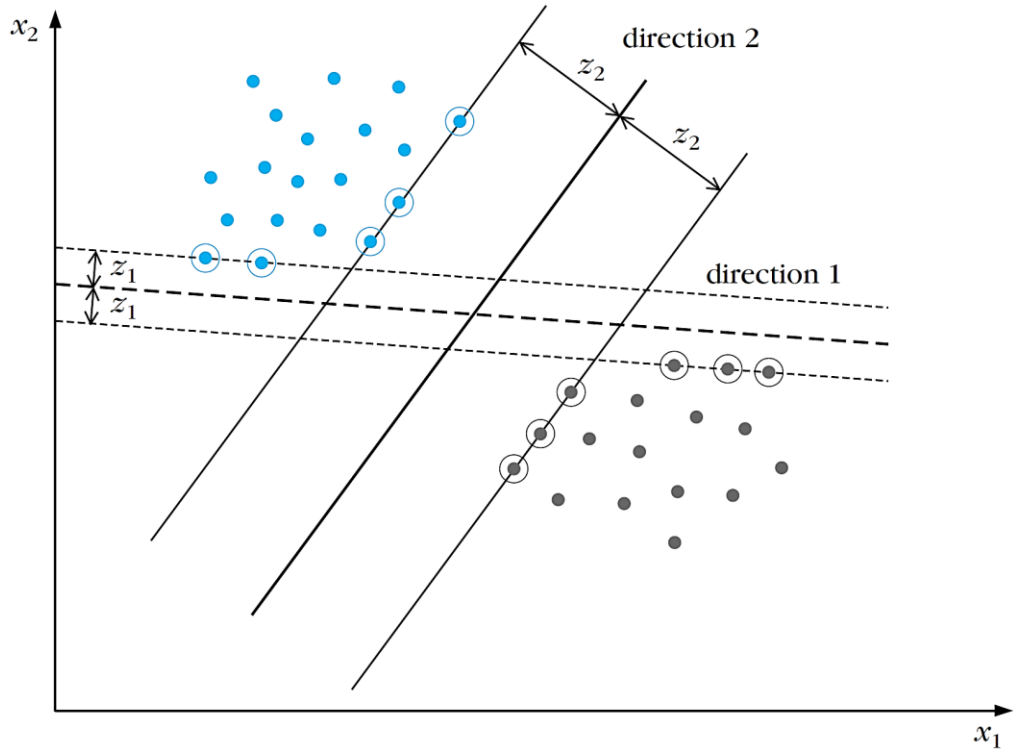
SVM idea

- Motivation:



Hard SVM

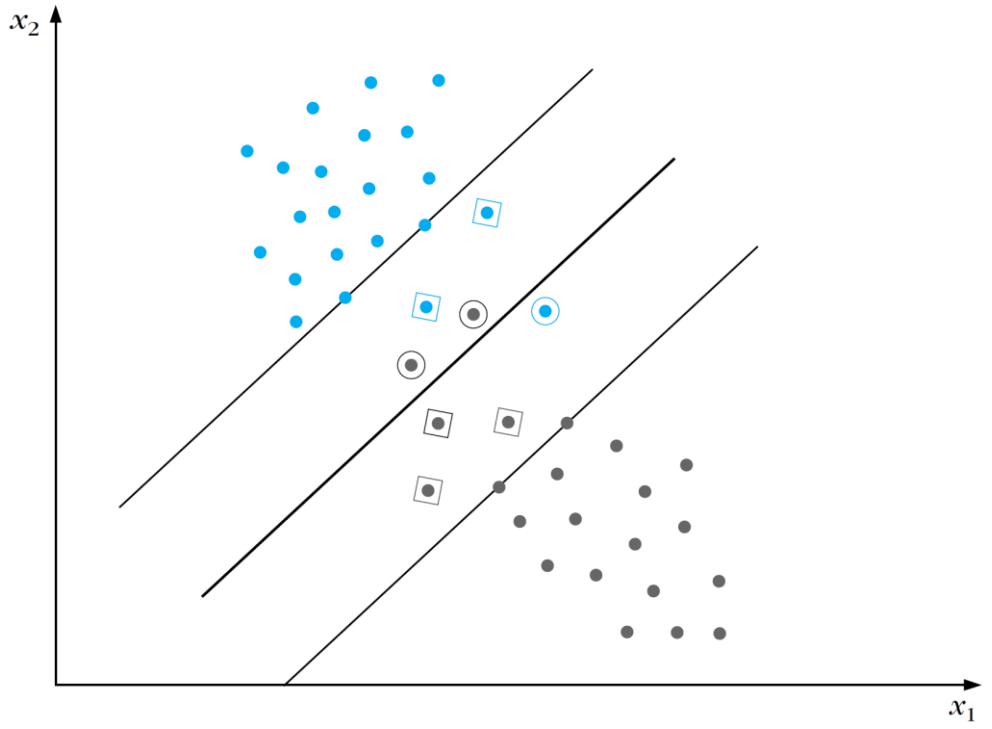
- A maximum margin classifier/regressor:
- This is Hard-SVM (linearly separable classes)



Soft SVM (C-SVM)

- What about non-separable (linearly) problems:

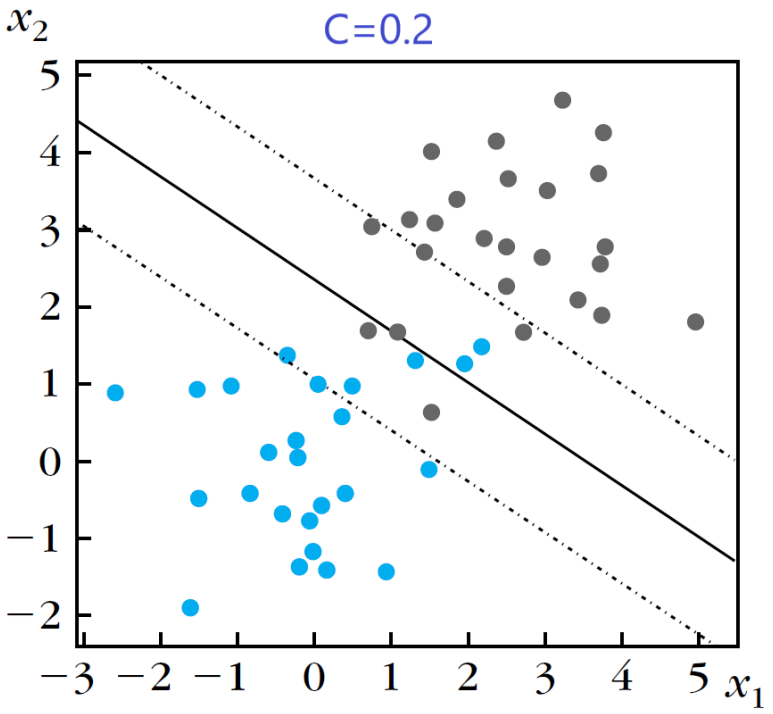
Solution 1: We use Soft-SVM (C-SVM), C is hyperparameter of algorithm.

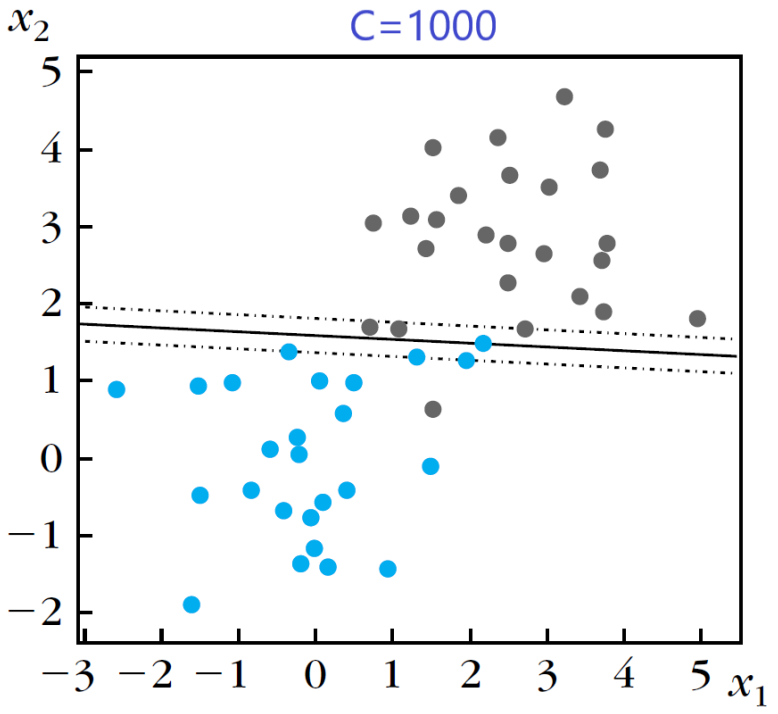


Soft SVM (C-SVM)

- What about non-separable (linearly) problems:

Solution 1: We use Soft-SVM (C-SVM), C is hyperparameter of algorithm.

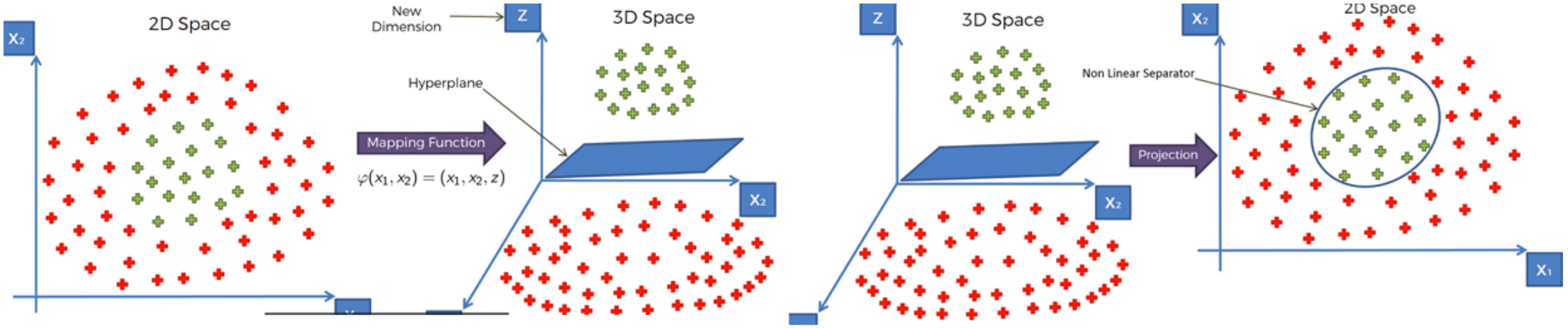




Kernel SVM

- What about non-separable (linearly) problems:

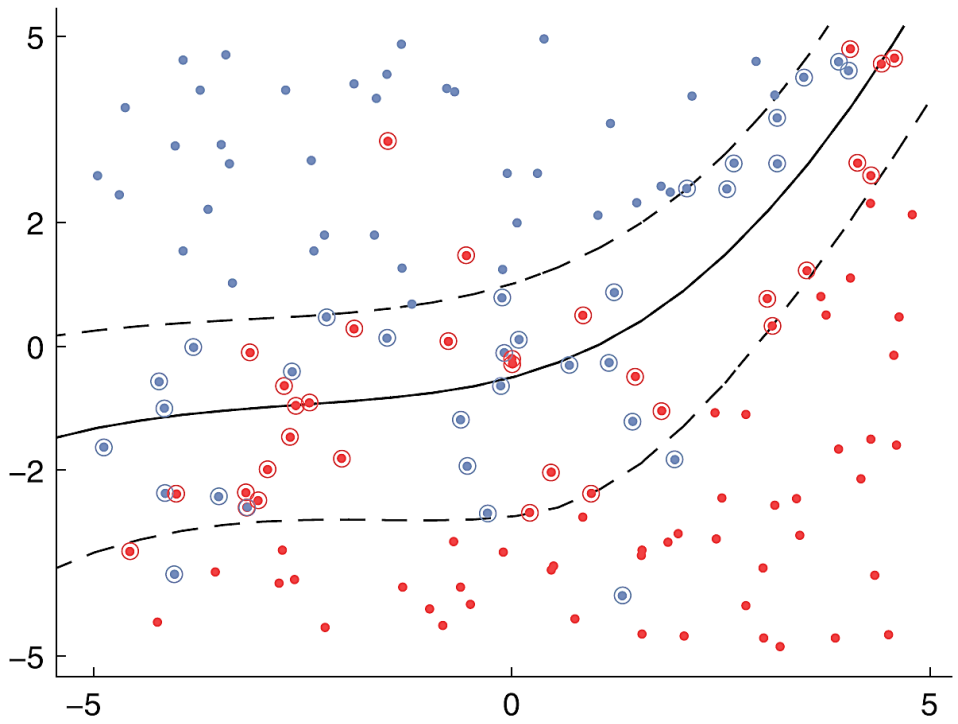
Solution 2: Kernel SVM (+soft SVM) ,

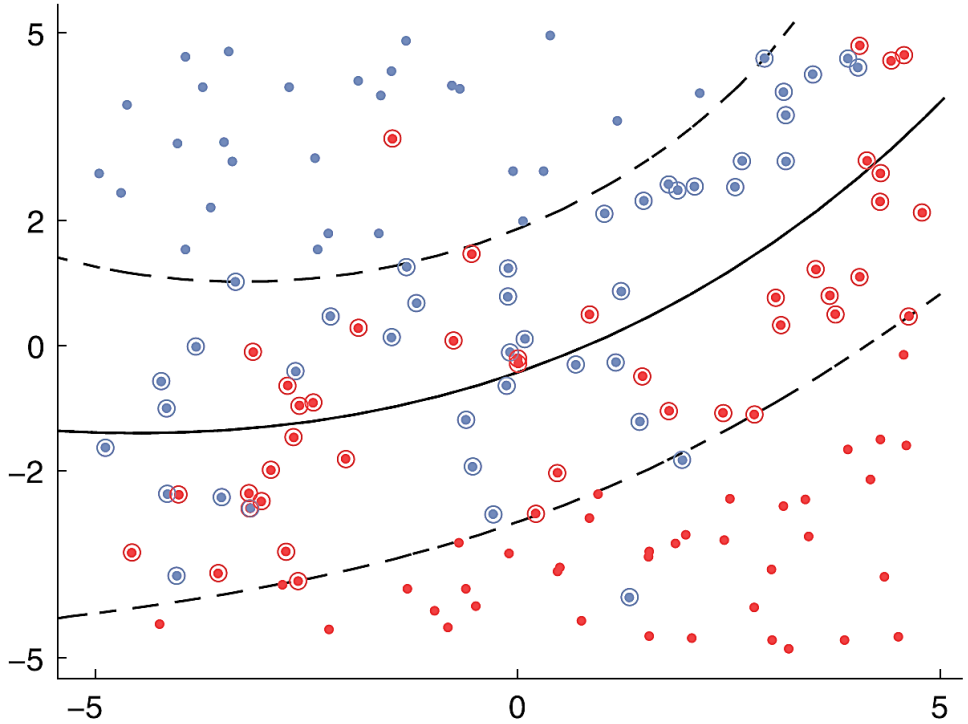


Kernel SVM

- What about non-separable (linearly) problems:

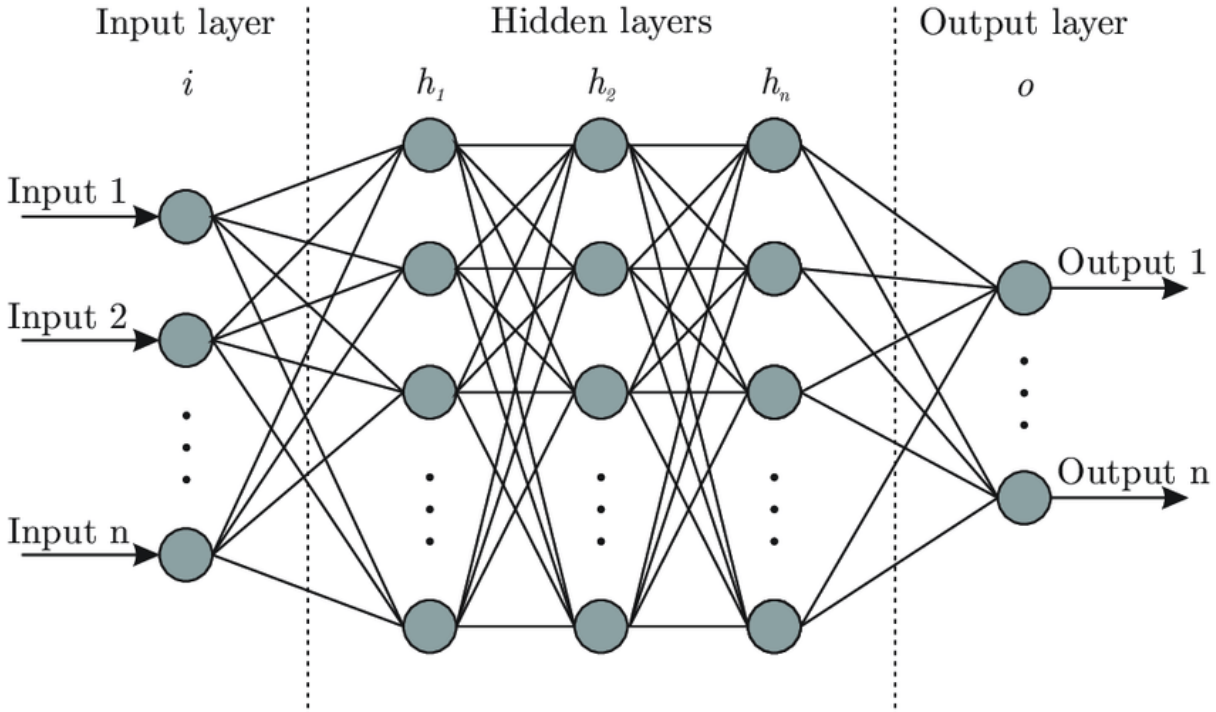
Solution 2: Kernel SVM (+soft SVM), C=20 (Left), C=1 (Right)





Artificial Neural Networks (ANN)

- Biologically inspired Algorithm
- A Shallow ANN

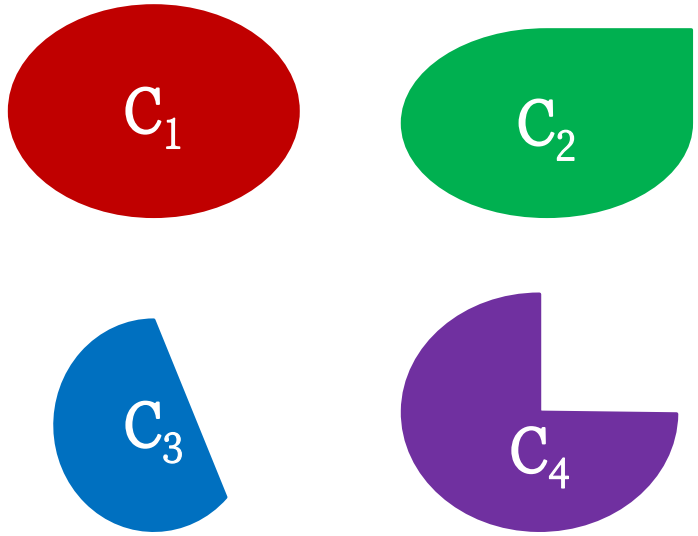


Multiclass Problems

- Use Multiclass classifier (ANN, Bayes, …)
- Use binary classifier (SVM):
  - One-vs-One (One-against-One)
  - One-vs-All/One-vs-Rest (One-against-All/One-against-Rest)

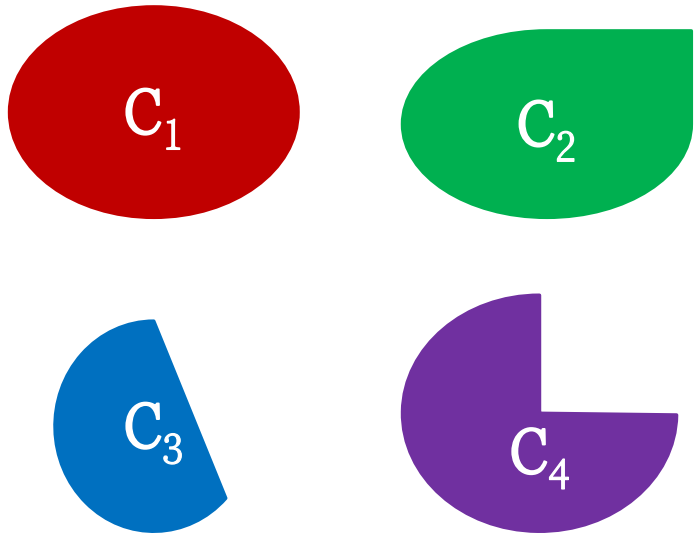
One-vs-One (One-against-One)

- The One-vs-One strategy perform a multi-class classification by $\frac{M(M-1)}{2}$ binary classification problem for each pair of classes.
- For M=4 class problems we need 6 binary classifier:
  - $C_1 \leftrightsquigarrow C_2$
  - $C_1 \leftrightsquigarrow C_3$
  - $C_1 \leftrightsquigarrow C_4$
  - $C_2 \leftrightsquigarrow C_3$
  - $C_2 \leftrightsquigarrow C_4$
  - $C_3 \leftrightsquigarrow C_4$



One-vs-Rest (One-against-Rest)

- The One-vs-Rest strategy perform a multi-class classification by M binary classification problem for each class.
- For M=4 class problems we need 4 binary classifier:
  - $C_1 \leftrightsquigarrow \{C_2, C_3, C_4\}$
  - $C_2 \leftrightsquigarrow \{C_1, C_3, C_4\}$
  - $C_3 \leftrightsquigarrow \{C_1, C_2, C_4\}$
  - $C_4 \leftrightsquigarrow \{C_1, C_2, C_3\}$

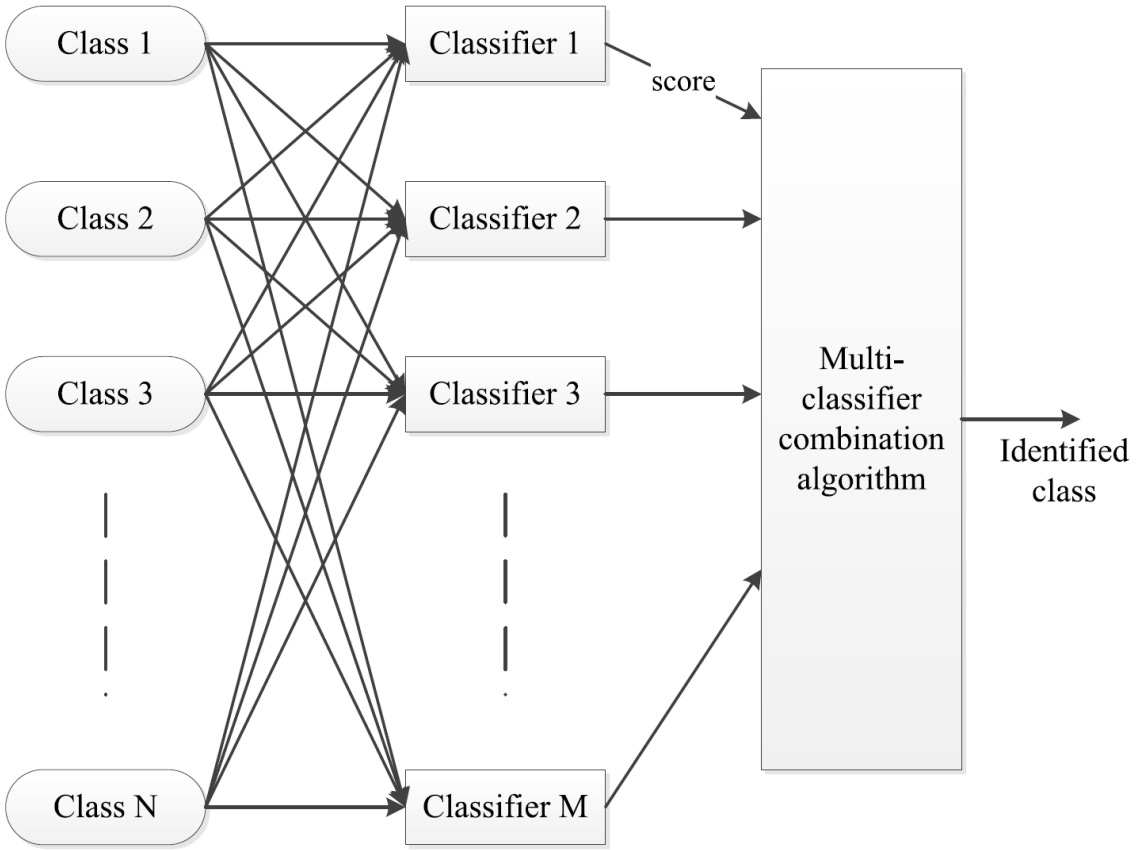


How to Enrich Classification

- Design M classifier
- Combine classifiers
- Motivation

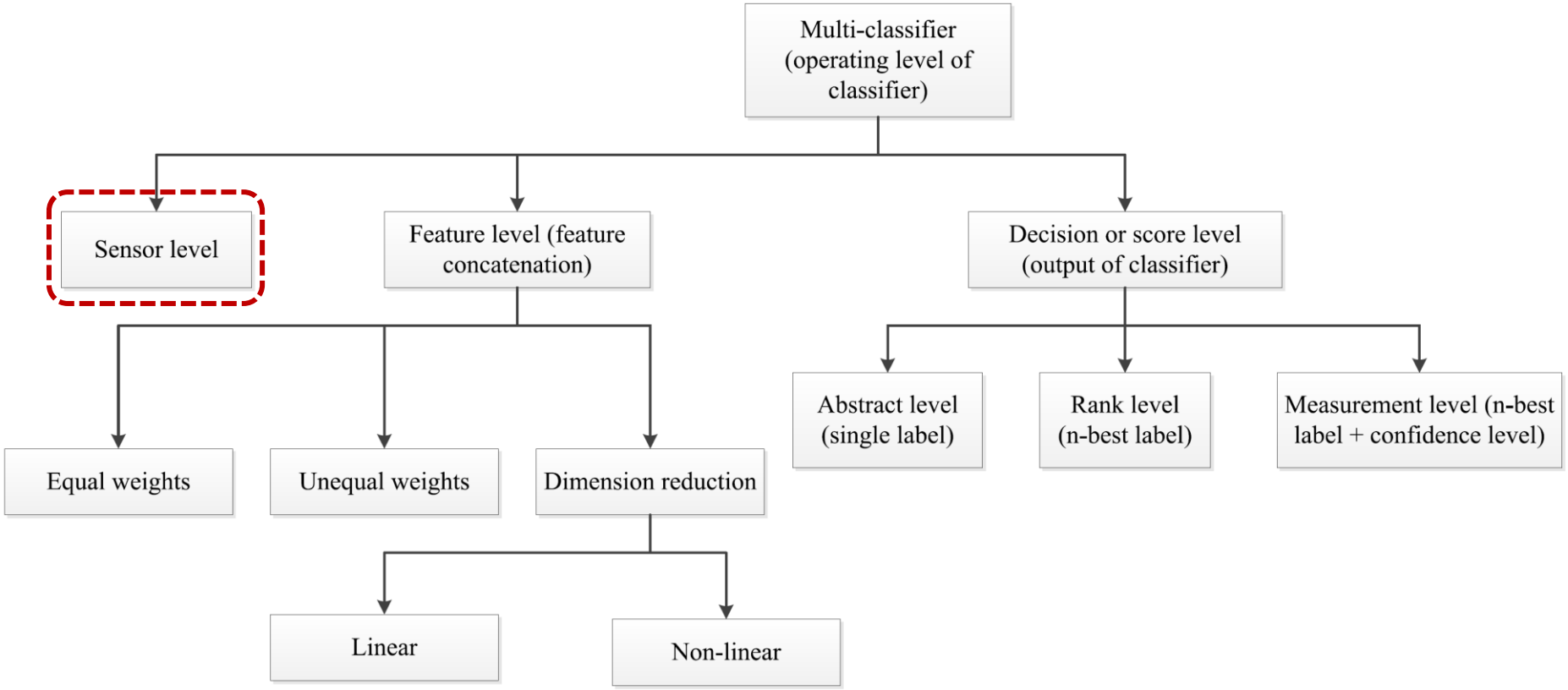
General Framework for Classifier Combination

- Combination:
  - Algorithmic
  - Trainable



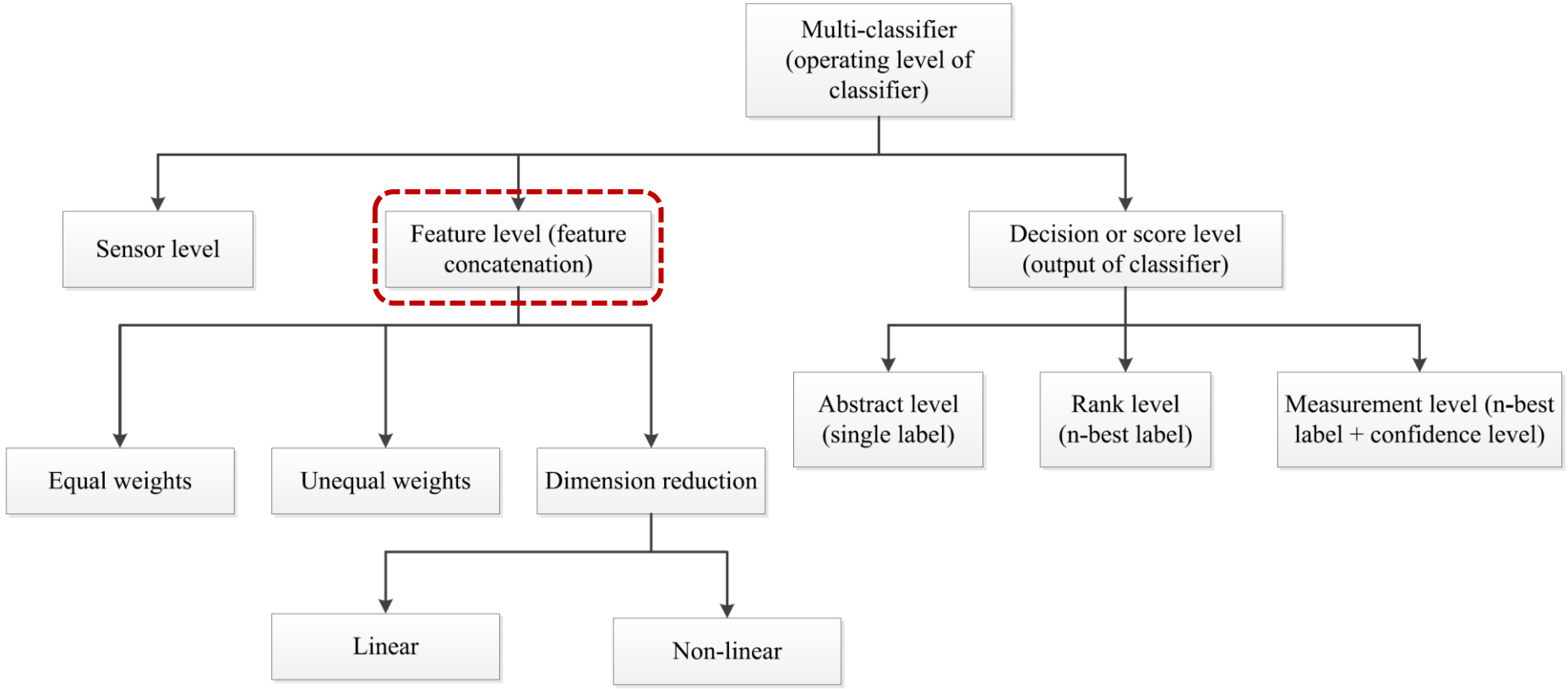
Different levels of classifiers combination

- There are several strategies:



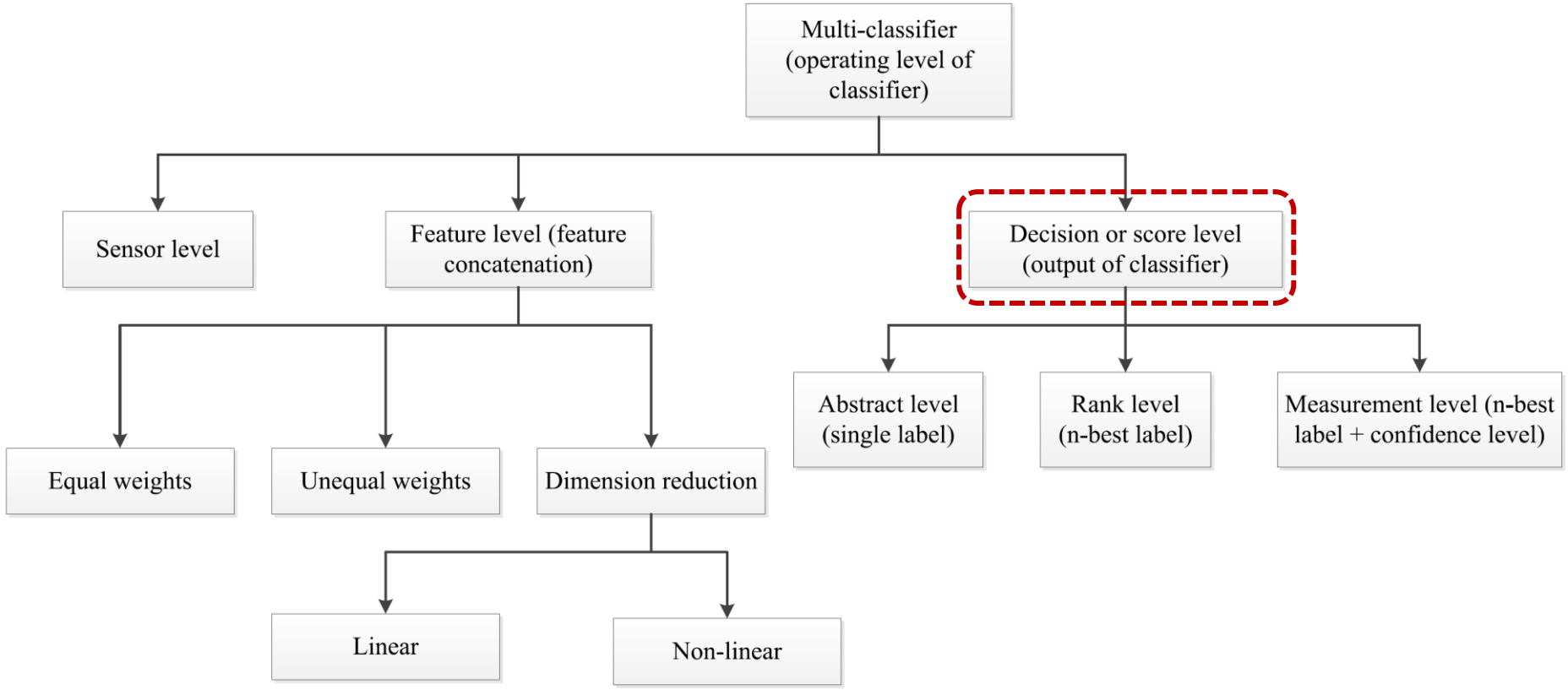
Different levels of classifiers combination

- There are several strategies:



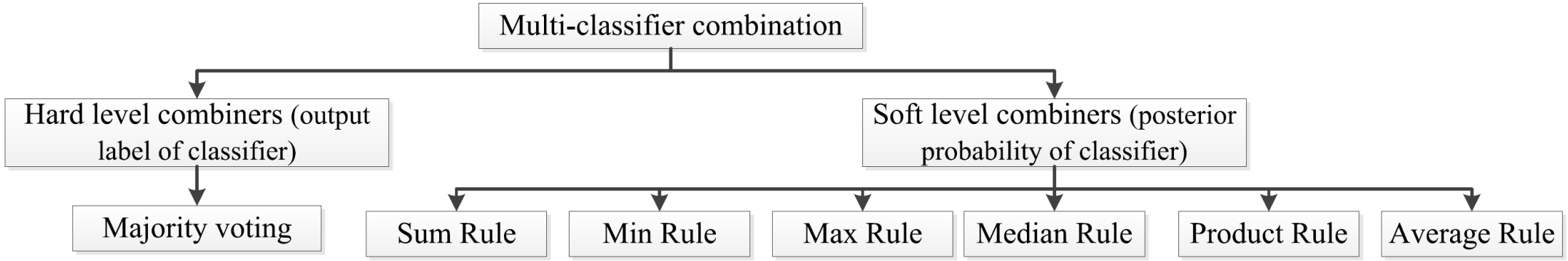
Different levels of classifiers combination

- There are several strategies:



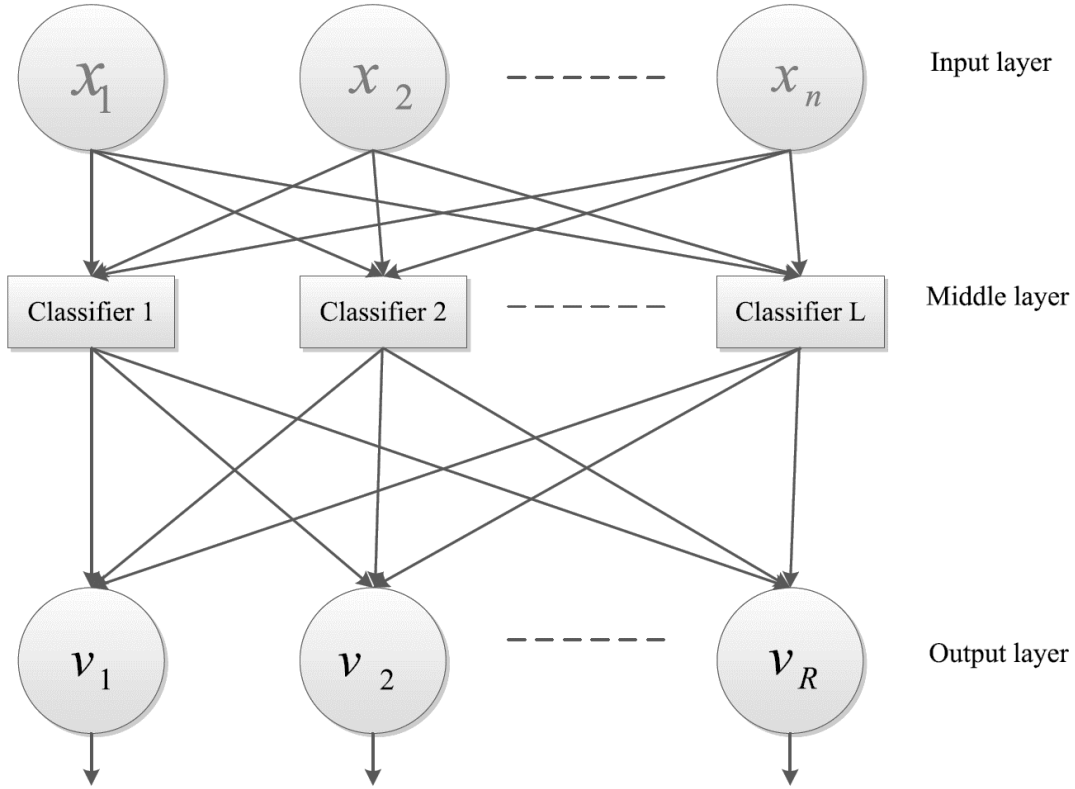
Score-Level Combination -Nonadaptive

- There are several strategies based on classifiers output:
  - Hard: $p_i(\Omega_j|\boldsymbol{x}) \in \{0, 1\},\ i = 1, 2, \dots, C,\ j = 1, 2, \dots, M$
  - Soft: $p_i(\Omega_j|\boldsymbol{x}) \in [0\ 1],\ i = 1, 2, \dots, C,\ j = 1, 2, \dots, M$
  - where:
    - C: number of classifiers
    - M: number of classes



Score-Level Combination -Adaptive

- Add a level of trainable combiner



Heterogeneous Ensemble Combination

- Design several *heterogeneous* classifiers complementing each other using the *whole* training data.
  - Different architectures,
  - Different hyperparameters (identical structure),
  - Different feature sets (identical structure).

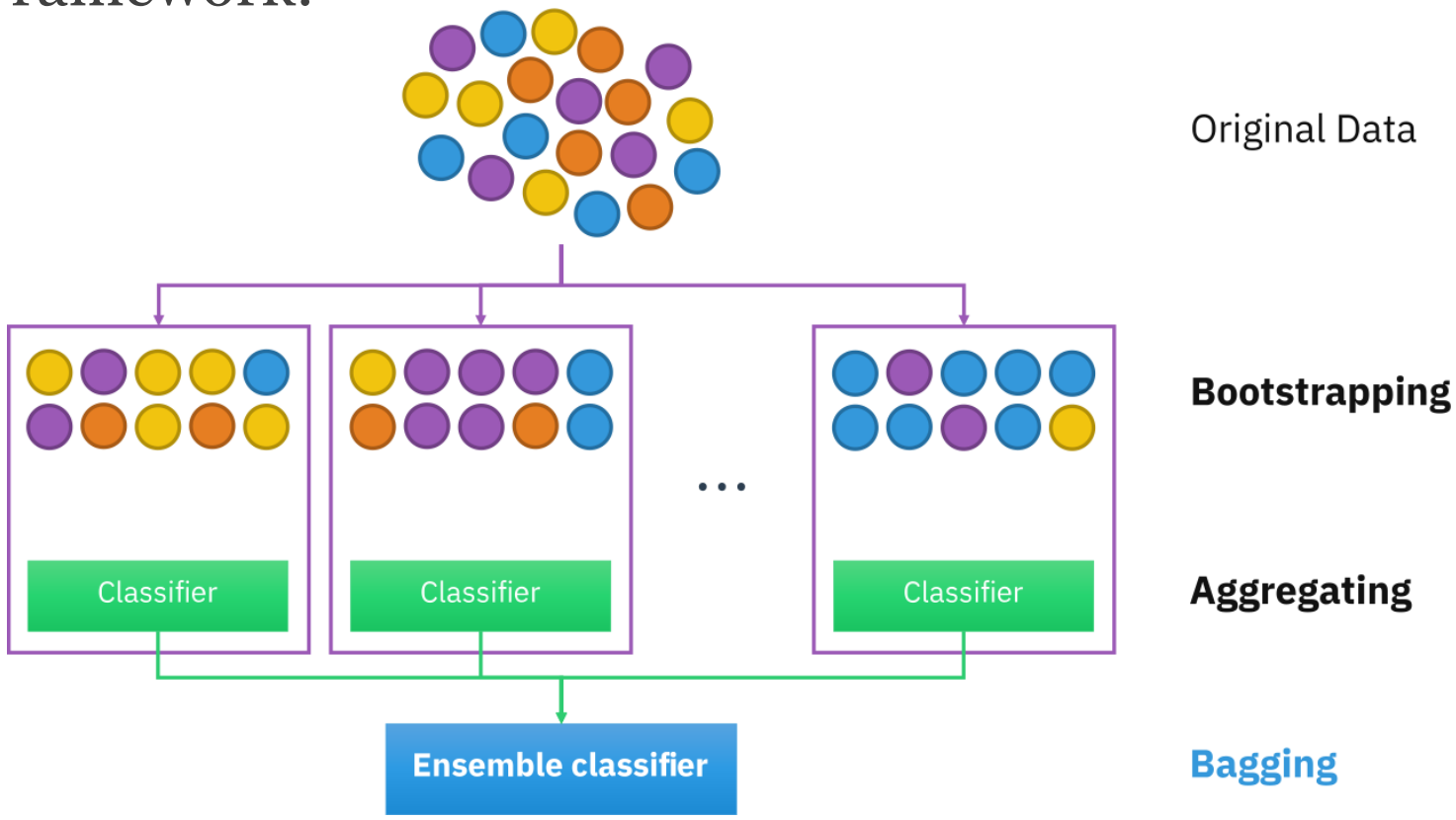
$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \Rightarrow \left\{ \begin{bmatrix} a \\ d \end{bmatrix}, \begin{bmatrix} b \\ d \end{bmatrix}, \begin{bmatrix} a \\ c \end{bmatrix}, \begin{bmatrix} c \\ d \end{bmatrix} \right\}$$

Homogeneous Ensemble Combination

- Design several *Homogeneous* of the same type built upon different data.
  - Bagging : Train individual classifiers using different small-size datasets by bootstrapping the original dataset, all classifier has *same structure*. (Random Forest)
  - Boosting: Train the individual classifiers (weak) *hierarchically* to discriminate more complex regions in the feature space (AdaBoost, Gradient Boosting, and XGBoost)
  - Bagging and Boosting may be preform in feature domain not data domain.

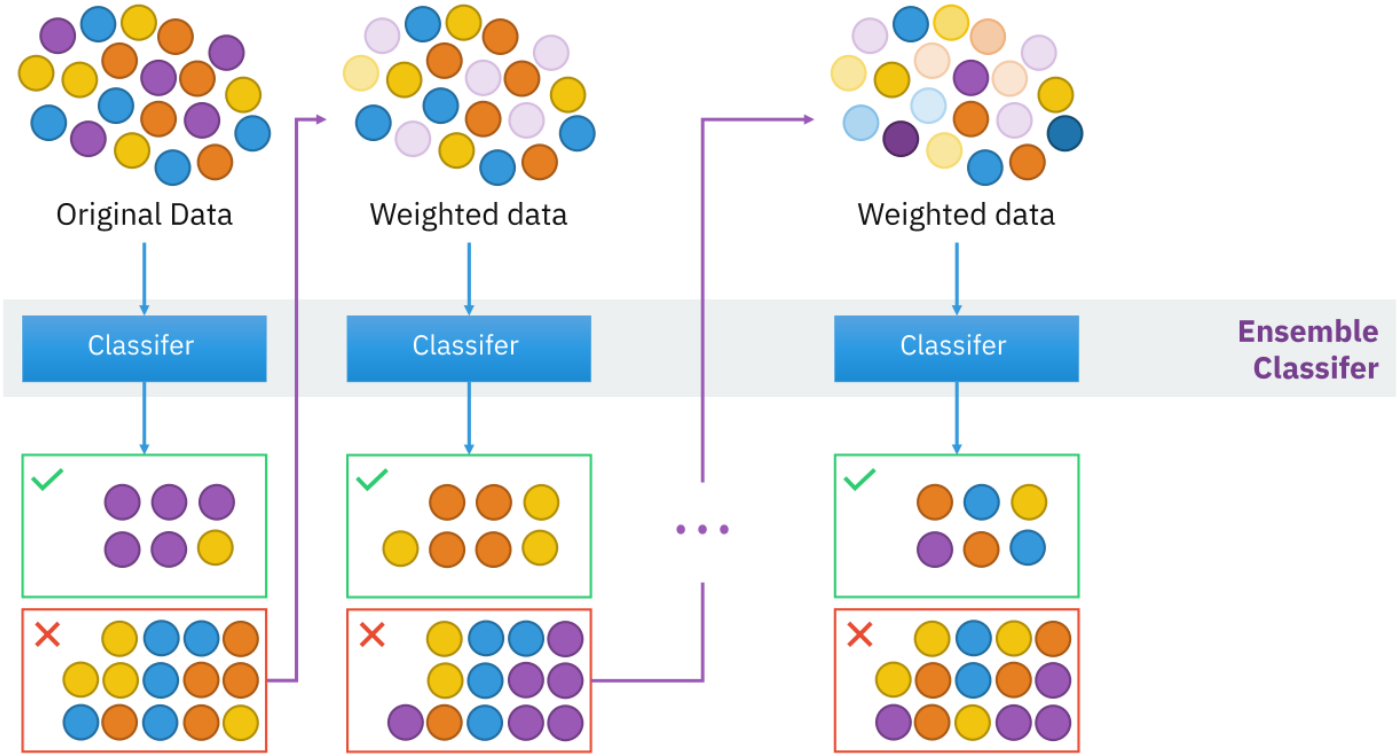
Bagging

- General Framework:



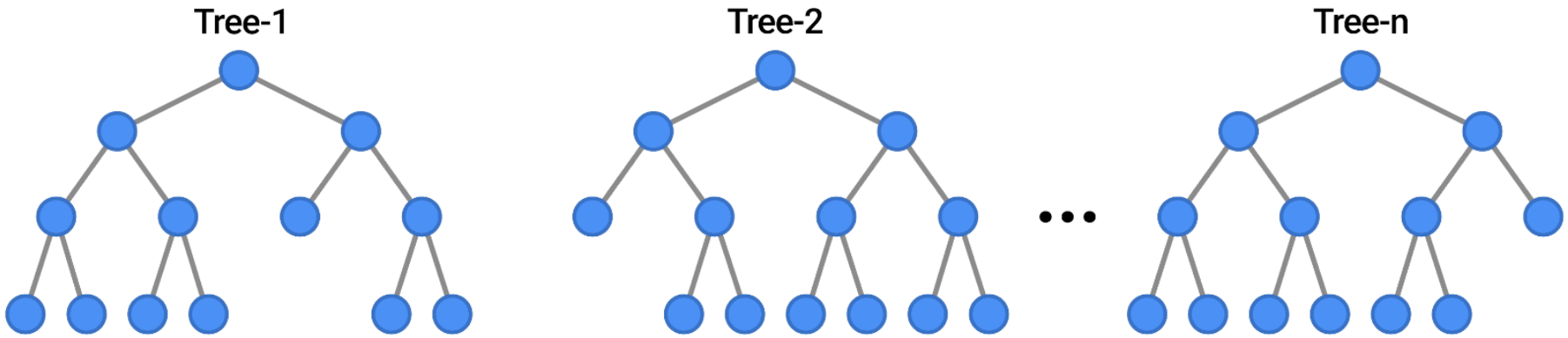
Boosting

- General Framework:



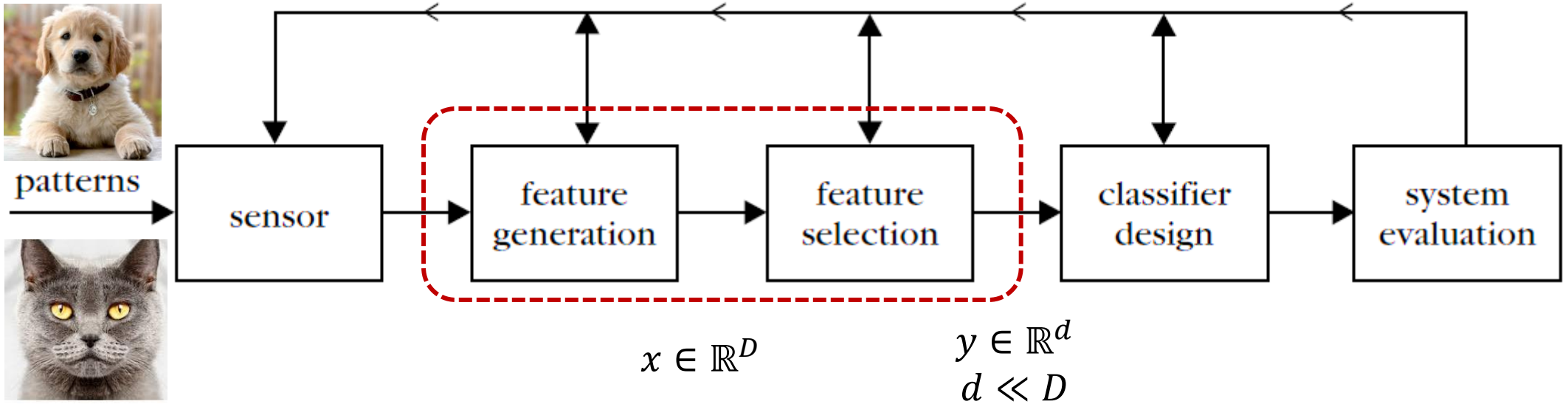
Random Forest

- Build several Decision Tree and combine their results



Classification System Anatomy

- General Framework:



- Feature Selection/Reduction: $y = \{Ax, f(x)\}, A \in \mathbb{R}^{d \times D}, f(\cdot): \mathbb{R}^D \rightarrow \mathbb{R}^d$

Model Variables

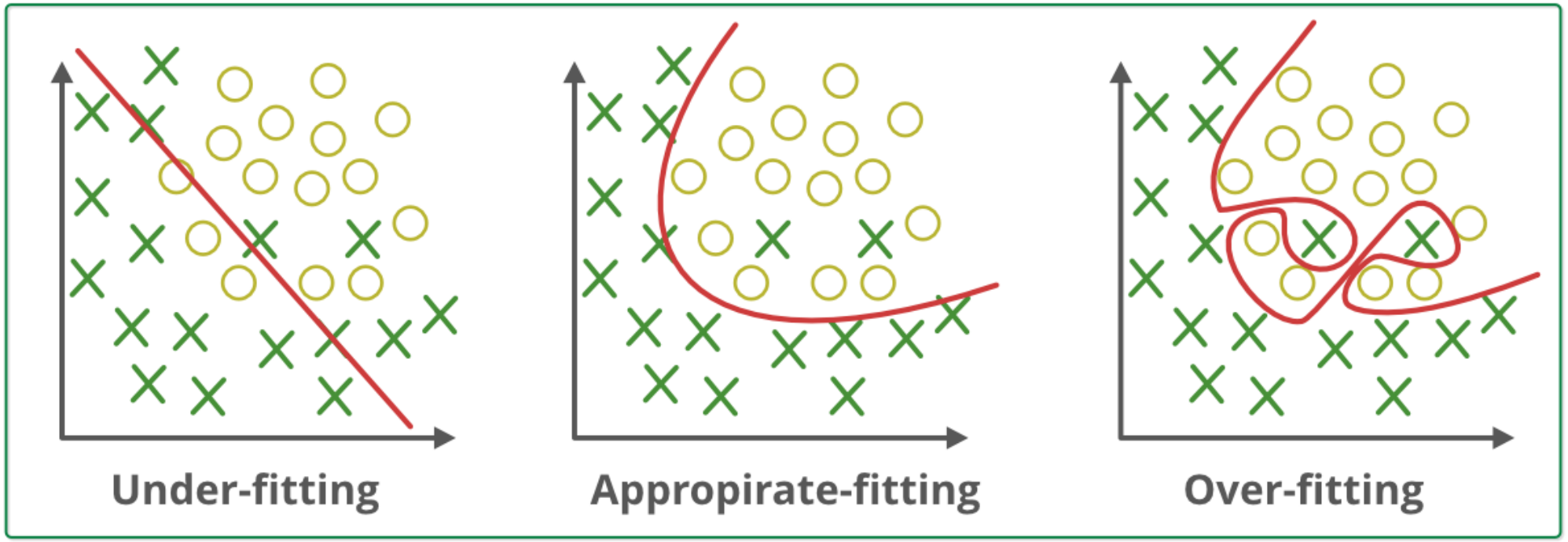
- Model Parameters: A model *parameter* is a configuration variable that is *internal* to the model and whose value can be estimated from training data during learning phase.
  - Coefficients in SVM (hyperplane), logistic regression, ANN, and etc.
- Model Hyperparameters: A model *hyperparameter* is a configuration variables that is *external* to the model and whose value cannot be estimated from training data.
  - K in K-NN, number of layers in MLP, priors in Bayes, and etc.
- Example: $y = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0$, m is hyperparameters. while $\{a_k\}_{k=0}^{m}$ are parameters.

Model Fitness

- Underfitted Model: Model performs *poorly* on the *training* data:
  - May be too simple
- Overfitted Model: Model performs *well* on the *training* data but *poorly* on test data.
  - May be too complex
- Proper Model: Model performs *well* on both *training* and test data.

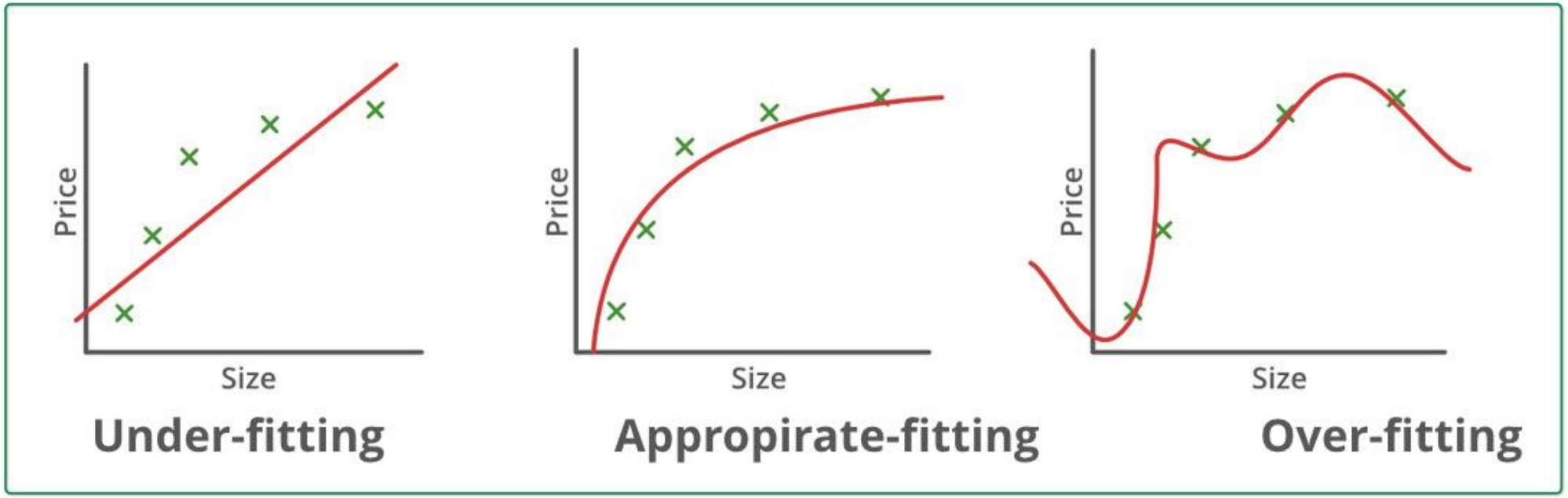
Model Fitness

- Classification



Model Fitness

- Regression

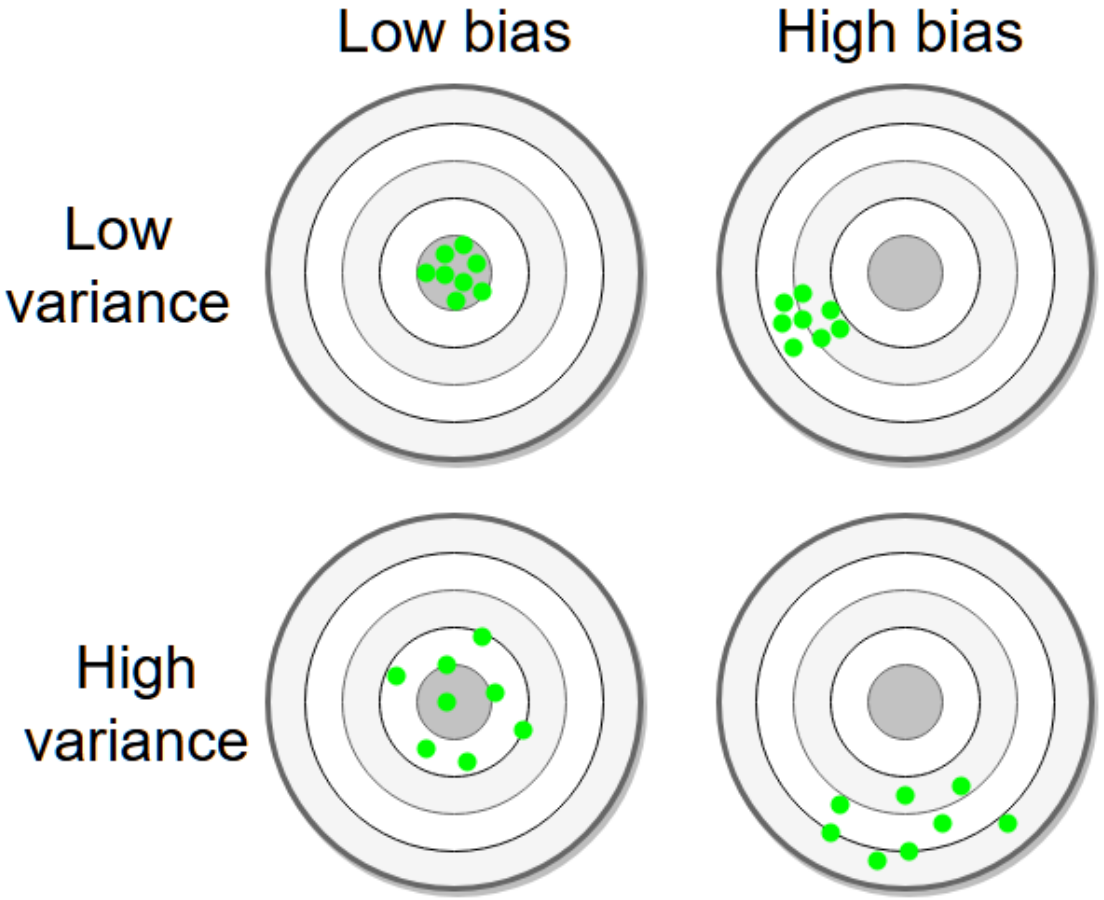


Bias-Variance Dilemma

- *Bias Error*: The error due to *bias* is taken as the difference between the *expected* (or *average*) prediction of our model and the correct value which we are trying to predict. Bias measures how far off in general these models' predictions are from the correct value.
- *Variance Error*: The error due to variance is taken as the *variability* of a model prediction for a given data point. The variance is how much the predictions for a given point vary between different realizations of the model.
- For expectation for a single model assume you could repeat the whole model building process more than once (data bootstrapping)

Bias-Variance Dilemma

- Too Simple Model:
  - High Bias-Low Variance
- Too Complex Model:
  - Low Bias-High Variance



Bias-Variance Dilemma Mathematics

- True model and measurements:

$$y = f(x) + \varepsilon, \qquad \{x_i, y_i\}_{i=1}^{N}, \varepsilon \sim N(0, \sigma_e^2)$$

- We estimate a model:

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

- Estimation Error:

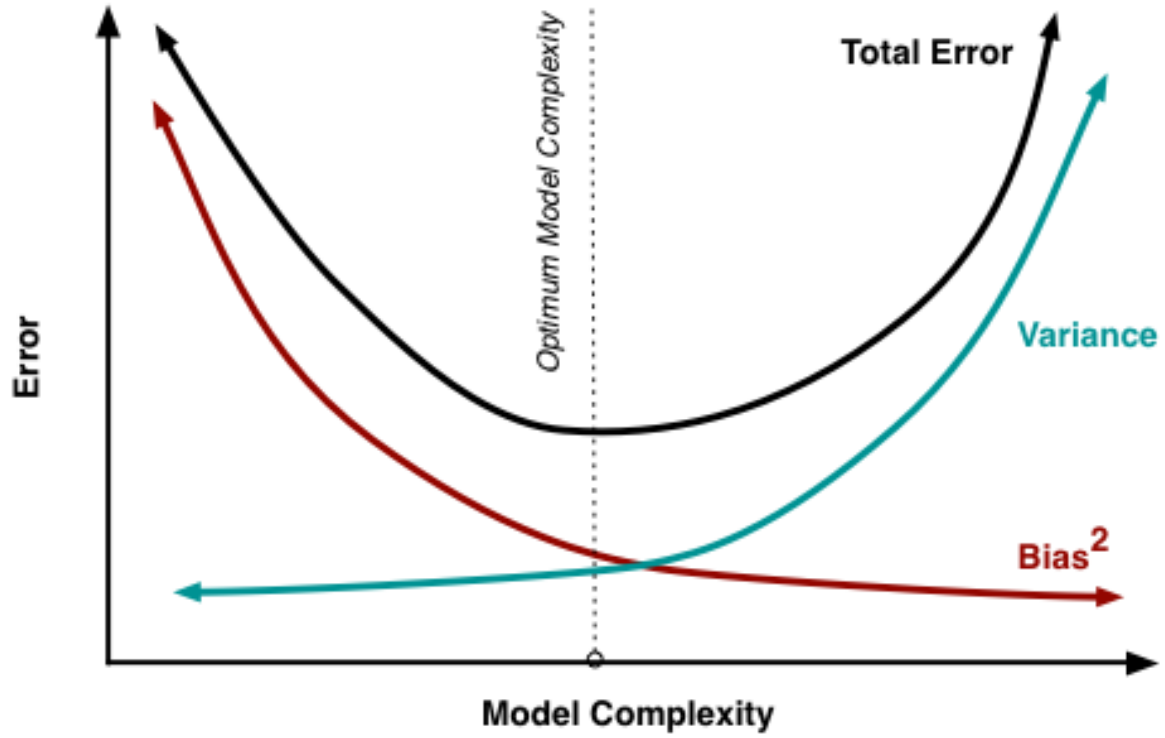
$$Error(x) = E\left\{\left|y - \hat{f}(x)\right|^2\right\}$$

- It can be shown:

$$Error(x) = \underbrace{\left(E\{\hat{f}(x)\} - f(x)\right)^2}_{Bias^2} + \underbrace{E\left\{\left(\hat{f}(x) - E\{\hat{f}(x)\}\right)^2\right\}}_{Variance} + \underbrace{\sigma_e^2}_{Irreducible\ Error}$$

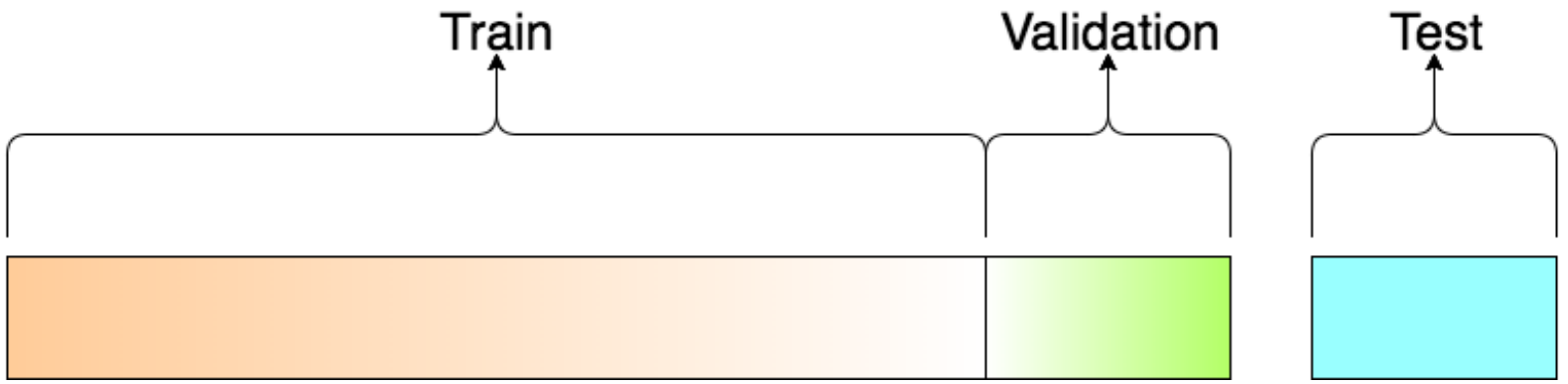
Bias-Variance Dilemma

- Understanding Over- and Under-Fitting



How to Use Labeled Data

- Training Set
- Validation/Development Set
- Test Set

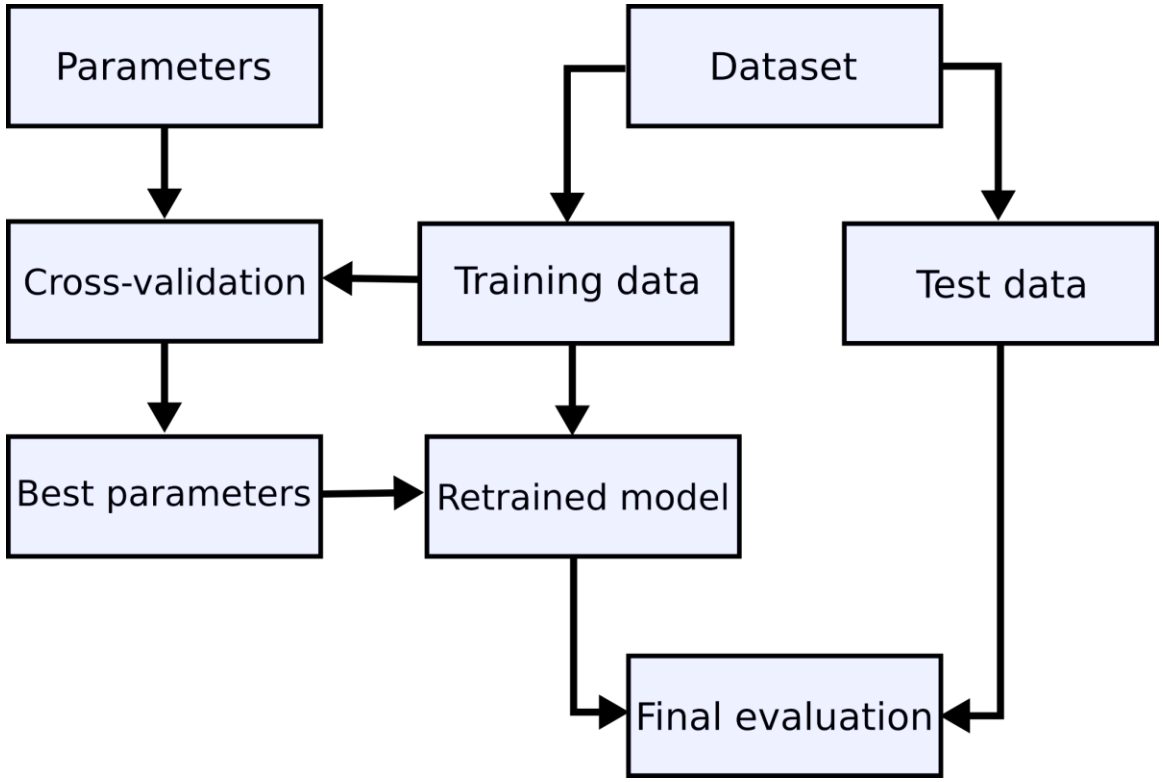


How to Use Labeled Data

- Training Set:
  - The actual dataset that we use to *train* the model, find parameters. The model sees and learns from this data.
- Validation/Development Set:
  - The sample of data used to provide an unbiased and fair evaluation of a model fit on the training dataset while tuning model hyperparameters.
  - The model occasionally sees this data, but never does it "Learn" from this.
- Test Set:
  - The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

How to Use Labeled Data

- Model Tuning

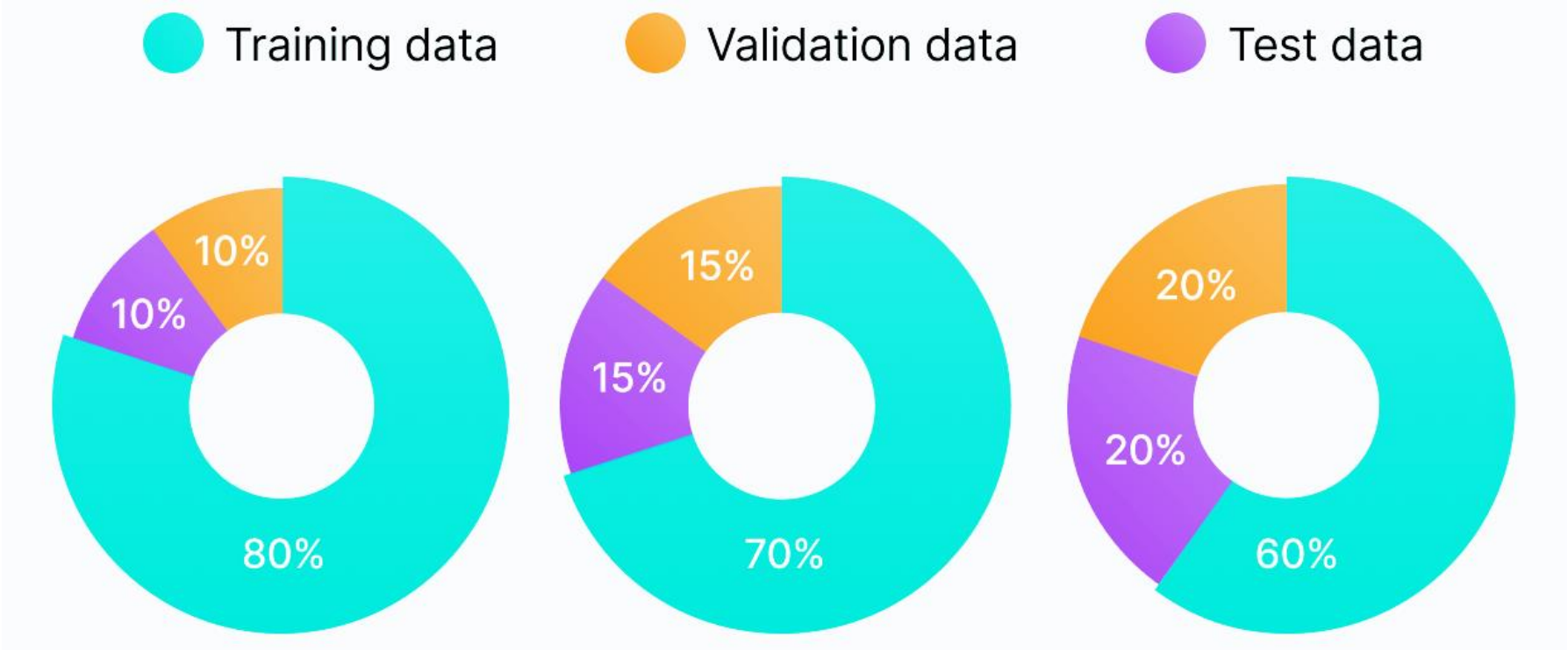


How to split Labeled Data

- The dataset split ratio depends on the number of samples present in the dataset and the model.
- Too many hyperparameters need large validation set.
- If there is less training data, the machine learning model will show high variance in training.
- With less testing data/validation data, your model evaluation/model performance statistic will have greater variance.

How to split Labeled Data

- There is no optimal split percentage.

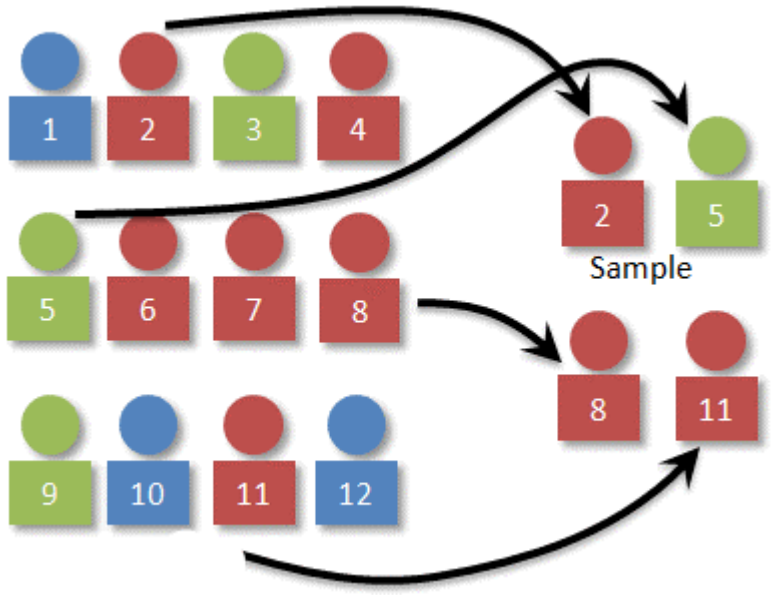


Splitting Strategy

- Most popular methods are:
  - Random: The simplest way to create a subset (train, validation, test) is to pick required instances randomly (equal chance) from a large dataset. It is good for *large* and balanced *database*.
  - Stratified: Split the dataset into subsets in a way that *preserves* the same *proportions* of examples in each class as observed in the *original* dataset.

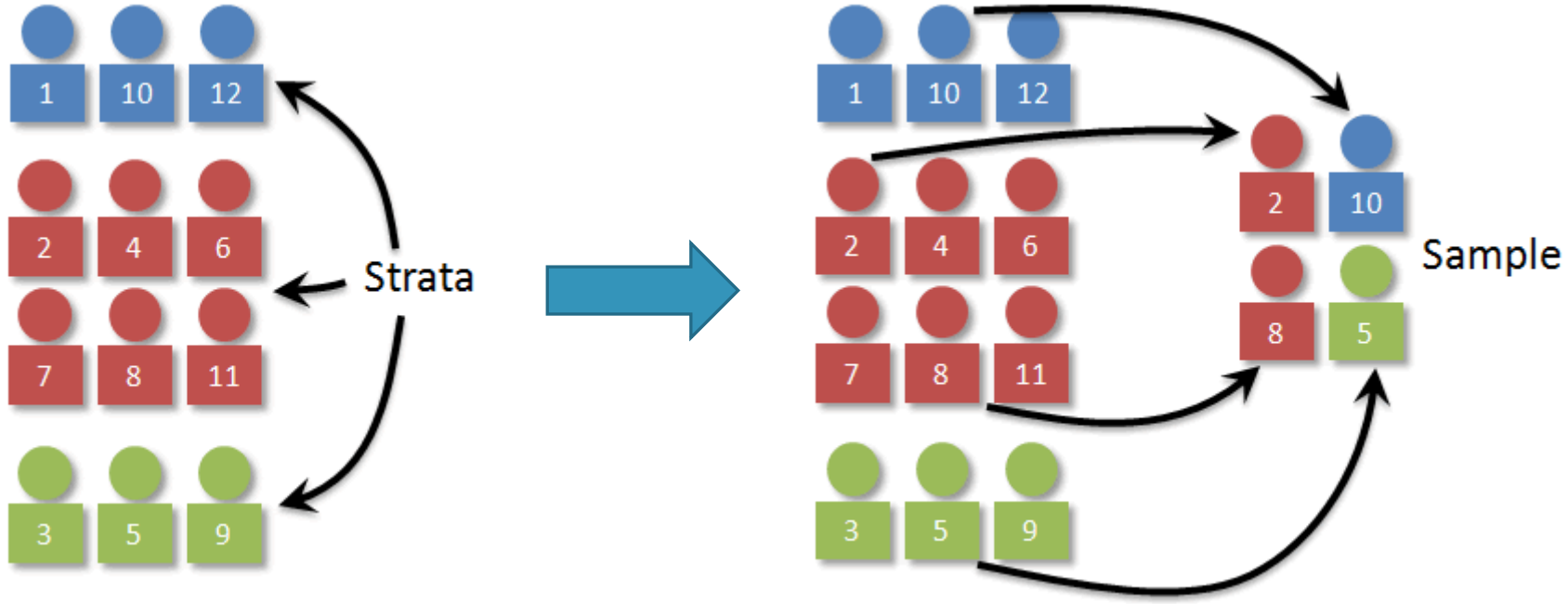
Random Sampling

- It is not guaranteed that the proportion of samples is preserved (small database)



Stratified Sampling

- It is guaranteed that the proportion of samples is preserved



K-Fold Cross Validation

- A statistical method used to estimate the skill, repeatability, generalization of machine learning models, and hyperparameter tuning.
  - Data set is partitioned into, K (Typically 10), roughly equal-sized parts
  - Perform training K times
  - In each training: One Part for evaluation (test), (K-1) Parts for Training
  - Combine results (mean and std)
  - Select best model (best parameters) with highest *average* performance

K-Fold Cross Validation

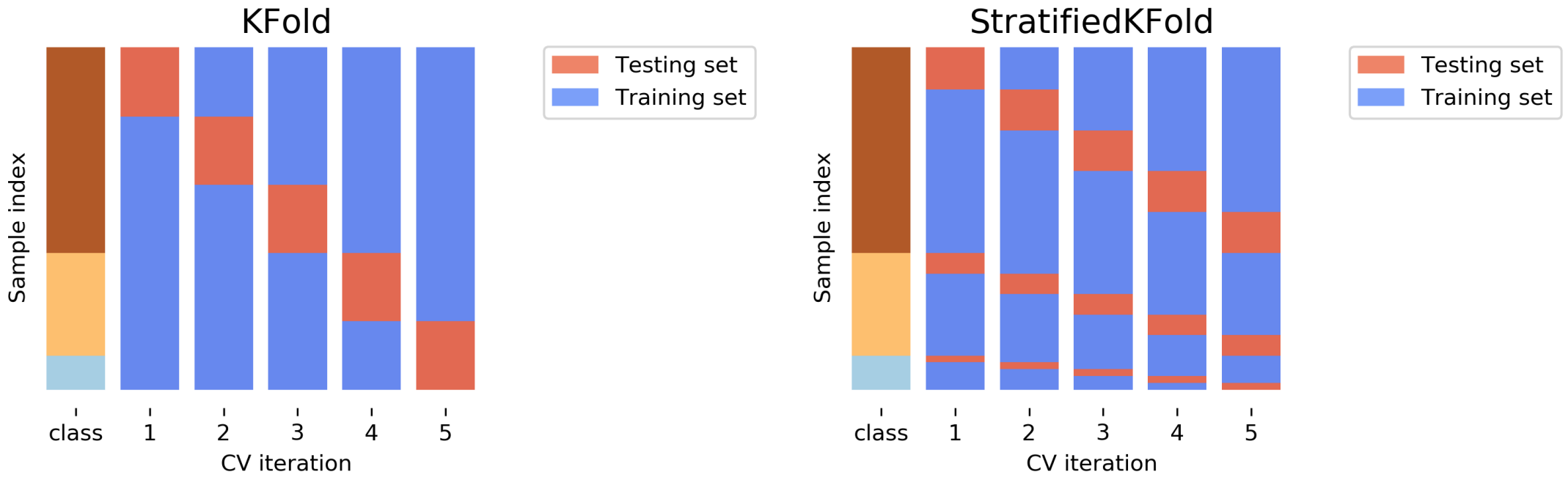
- Strategy for K=5:

	Estimation (Train+Validation)					Test
Fold #1	Validation	Train	Train	Train	Train	
Fold #2	Train	Validation	Train	Train	Train	
Fold #3	Train	Train	Validation	Train	Train	
Fold #4	Train	Train	Train	Validation	Train	
Fold #5	Train	Train	Train	Train	Validation	

- $K = N$: Leave-one-Out (LOO) Cross Validation
- Bootstrap: Drawing random sample of size ($\sim N/K$), K times!

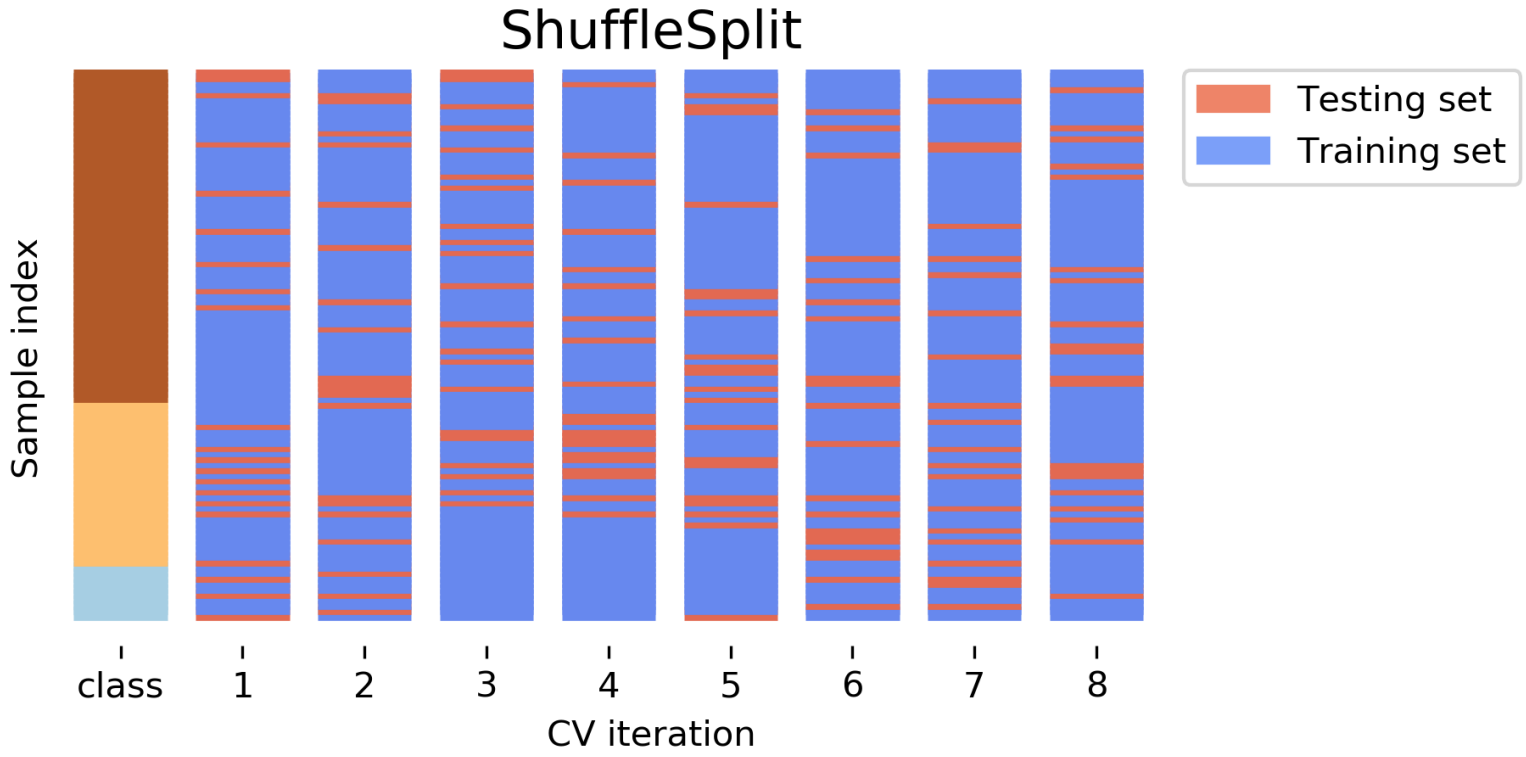
K-Fold Variations

- K-Fold vs Stratified K-Fold



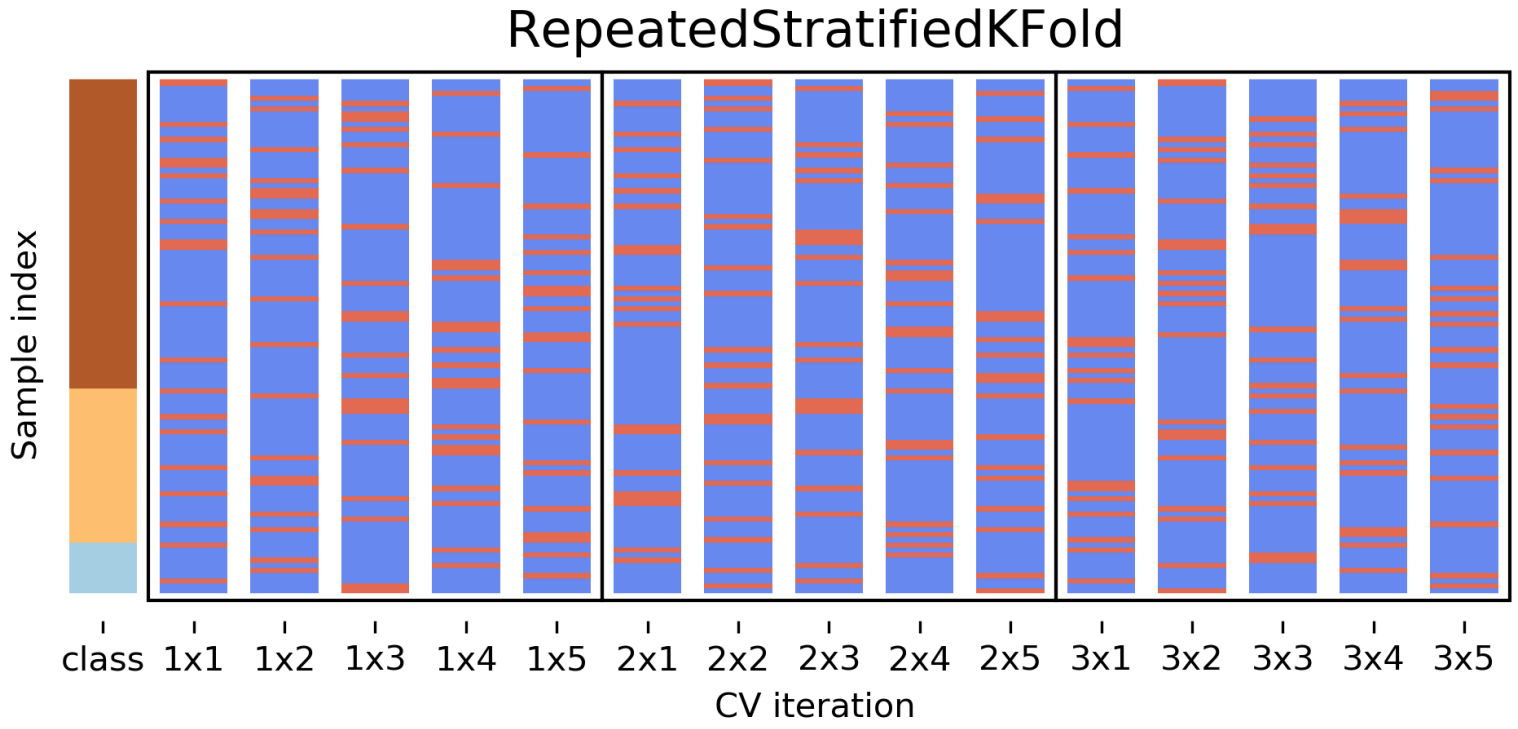
K-Fold Variations

- Bootstrap (Shuffle Split)



K-Fold Variations

- Repeated Stratified (Shuffle Split)

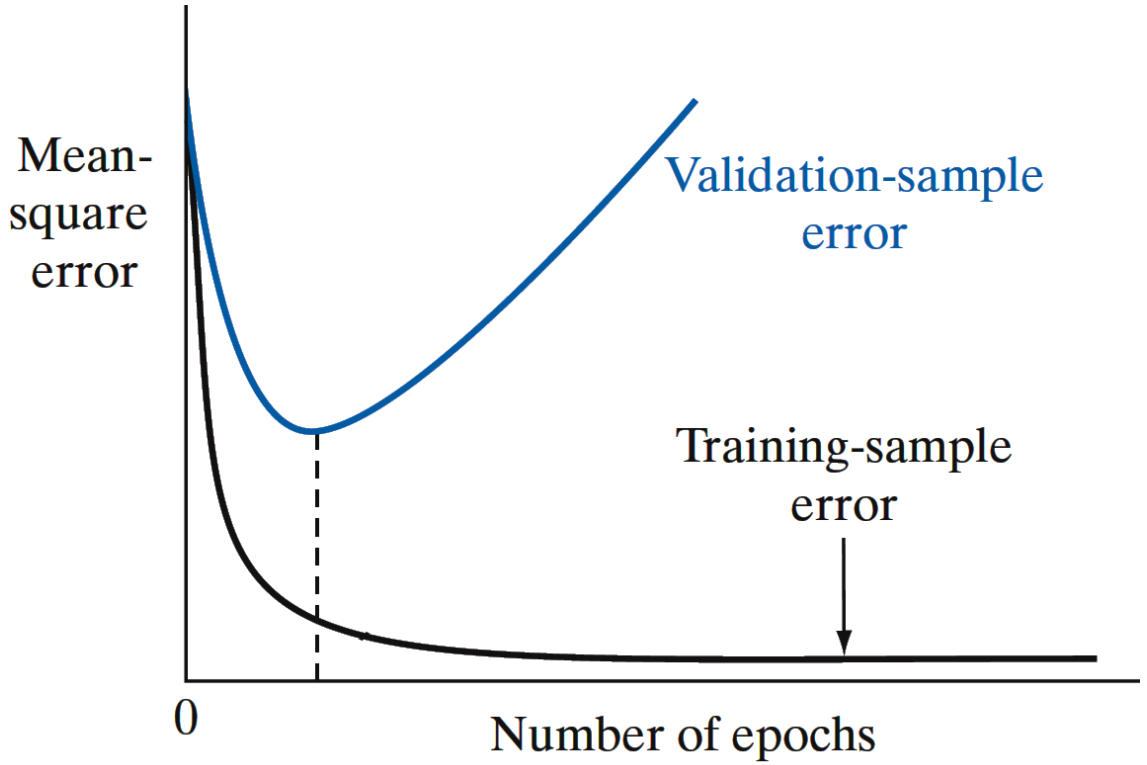


K-Fold Variations

- See: https://amueller.github.io/aml/04-model-evaluation/1-data-splitting-strategies.html

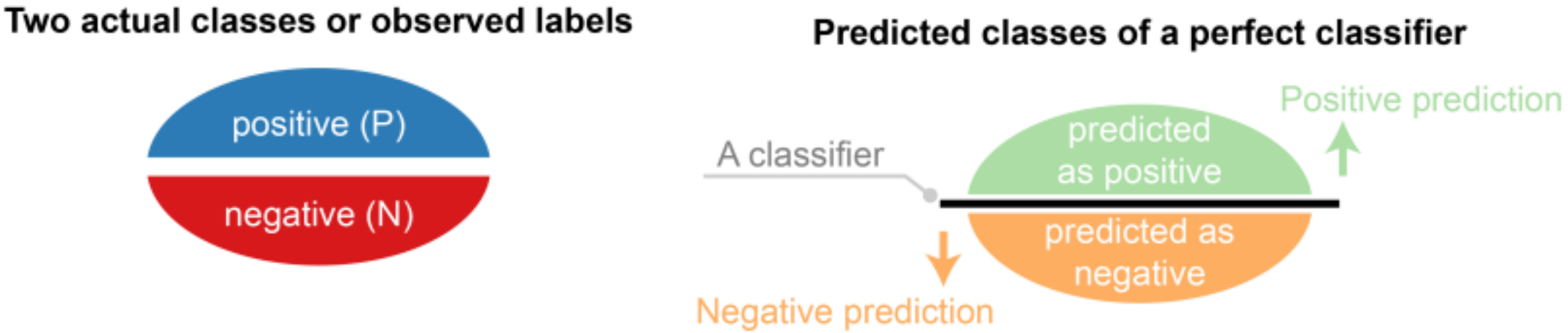
Machine Learning Evaluation

- Typical performance curve:



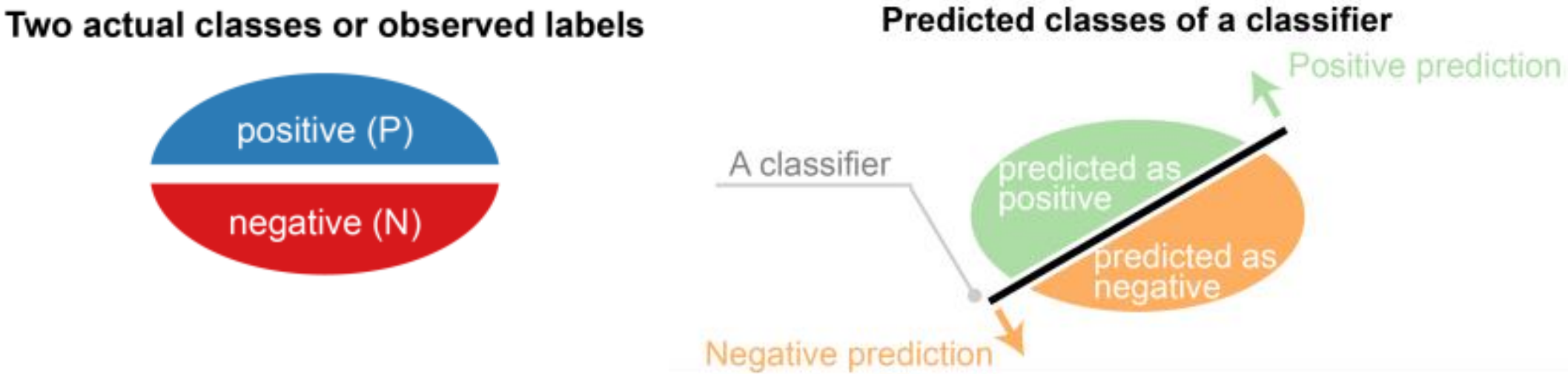
Performance Evaluation

- Consider two-class binary classification (Positive vs Negative)
- We have actual and predicted labels.
- Perfect Classifier:



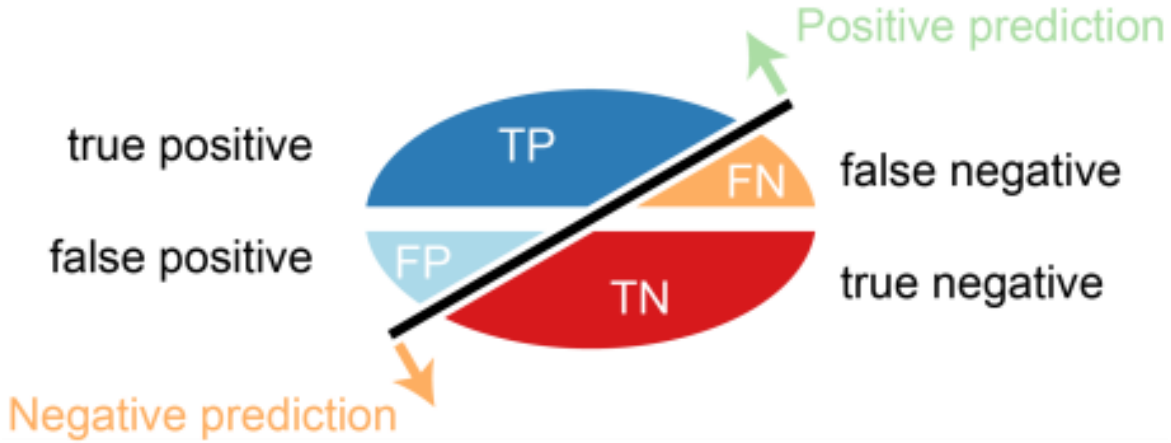
Performance Evaluation

- Real world Classifier:



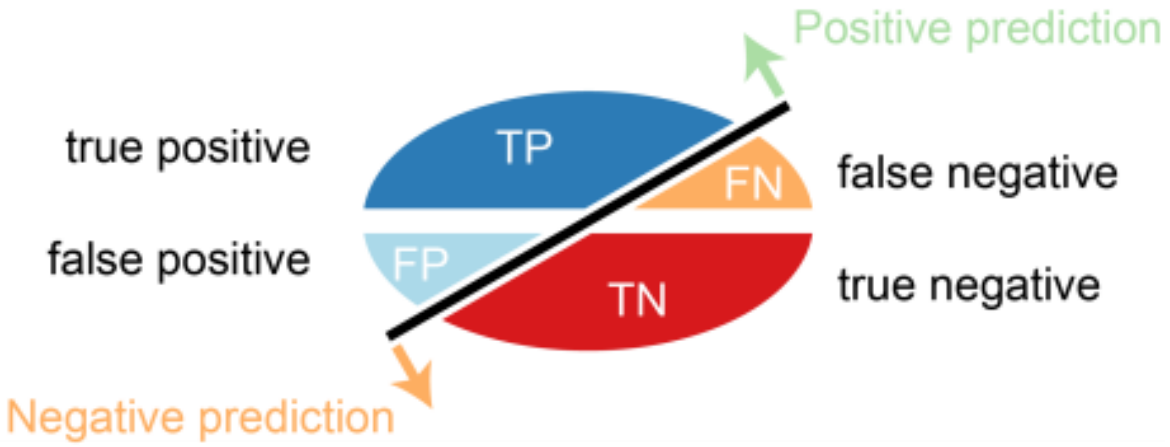
Performance Evaluation

- Consider two-class binary classification
- We have 4 situations:
  - True – Positive ✓
  - True – Negative ✓
  - False – Positive ✗
  - False – Negative ✗



Confusion Matrix

- Tabulate all decision (correct and wrong)



		True Label	
		Positive	Negative
Predicted Label	Positive	TP	FP
	Negative	FN	TN

Accuracy Measure

- Accuracy:

$$Accuracy = \frac{TN + TP}{FN + FP + TN + TP}$$

- Class balance effect and large TN.

Error-Rate Measure

- Error Rate:

$$ERR = \frac{FN + FP}{FN + FP + TN + TP} = 1 - Accuracy$$

- Class balance effect and large TN.

Recall Measure

- Sensitivity or Recall:

$$Recall = \frac{TP}{TP + FN}$$

Specificity Measure

- Specificity:

$$Specificity = \frac{TN}{TN + FP}$$

Precision Measure

- Precision (PPV - Positive Predictive Value):

$$Precision = \frac{TP}{TP + FP}$$

- NPV - Negative Predictive Value:

$$Precision = \frac{TN}{TN + FN}$$

Dice & Jaccard Measure

- Dice Index:

$$Dice = \frac{2TP}{2TP + FP + FN} = \frac{2|P \cap A|}{|P| + |A|}$$

- P: Predicted, A: Actual decision
- Jaccard Index:

$$Jaccard = \frac{TP}{TP + FP + FN} = \frac{|P \cap A|}{|P \cup A|}$$

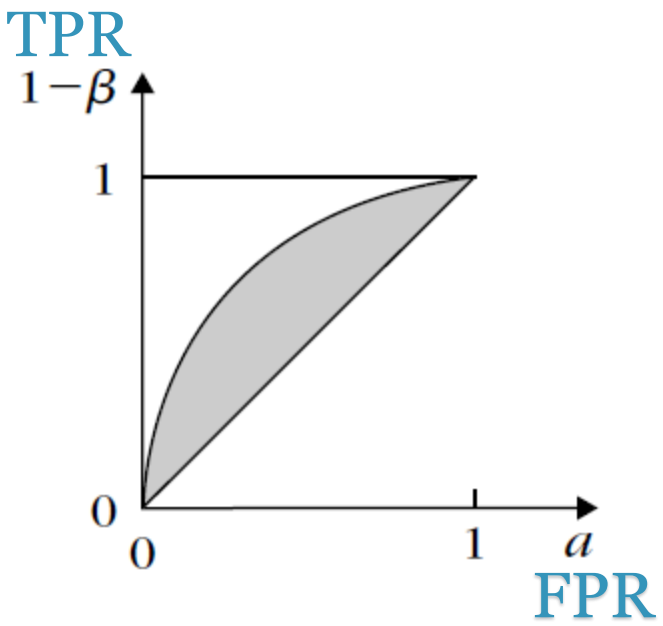
- Index zoo:
  - https://en.wikipedia.org/wiki/Precision_and_recall
- Exercise: Write Dice and Jaccard index for M Class problem.

ROC (Receiver Operating Characteristic)

- Performance Measurement for Two-Class Classification at Various Thresholds.
- Inputs:
  - Target class, {0, 1}
  - Classifier score [0 1]
- Algorithm:
  - Discretize scores with various threshold
  - Find TPR and FPR for each threshold
  - Plot TPR-FPR curve
- See https://github.com/brian-lau/MatlabAUC for practical implementation.

ROC (Receiver Operating Characteristic)

- ROC is TPR against FPR



- AUC: Area Under Curve

Machine Learning