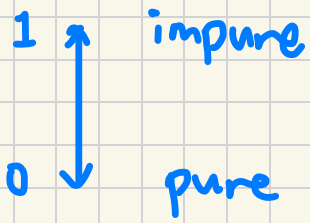
Variations for Decision Tree

- Importance measurement
  - Entropy, Gini index
  - Information gain, gain ratio
- Algorithms
  - ID3
  - C4.5, C5.0
  - CART (Classification And Regression Tree)
  - …



Gini impurity [edit]

Gini impurity, **Gini's diversity index**,[23] or **Gini-Simpson Index** in biodiversity research, is named after Italian mathematician Corrado Gini and used by the CART (classification and regression tree) algorithm for classification trees. Gini impurity measures how often a randomly chosen element of a set would be incorrectly labeled if it were labeled randomly and independently according to the distribution of labels in the set. It reaches its minimum (zero) when all cases in the node fall into a single target category.

For a set of items with J classes and relative frequencies p_i, $i \in \{1, 2, \ldots, J\}$, the probability of choosing an item with label i is p_i, and the probability of miscategorizing that item is $\sum_{k \neq i} p_k = 1 - p_i$. The Gini impurity is computed by summing pairwise products of these probabilities for each class label:

$$\mathrm{I}_G(p) = \sum_{i=1}^{J} \left(p_i \sum_{k \neq i} p_k \right) = \sum_{i=1}^{J} p_i (1 - p_i) = \sum_{i=1}^{J} (p_i - p_i^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} p_i^2 = 1 - \sum_{i=1}^{J} p_i^2 .$$

The Gini impurity is also an information theoretic measure and corresponds to Tsallis Entropy with deformation coefficient $q = 2$, which in physics is associated with the lack of information in out-of-equilibrium, non-extensive, dissipative and quantum systems. For the limit $q \to 1$ one recovers the usual Boltzmann-Gibbs or Shannon entropy. In this sense, the Gini impurity is nothing but a variation of the usual entropy measure for decision trees.

Performance Metrics for Classification Models

- **Confusion matrix**
  - Example: two-class prediction

		Predicted Class	
		yes	***no***
Actual Class	***yes***	true positive (TP)	false negative (FN)
	no	false positive (FP)	true negative (TN)

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

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN}$$

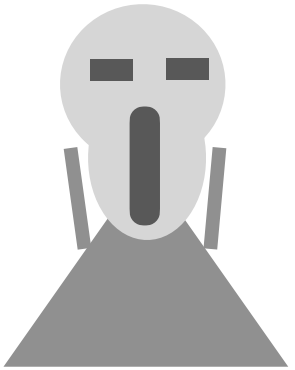
$$\text{Specificity} = \frac{TN}{FP + TN}$$

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

$$F1 - score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

General Hypothesis Space

Exponential Computation



Ensemble Learning

- Select a collection (or ensemble) of hypotheses, and combine their predictions by averaging, voting, or by another level of machine learning
  - Base models: individual hypotheses
  - Ensemble model: their combination

combine

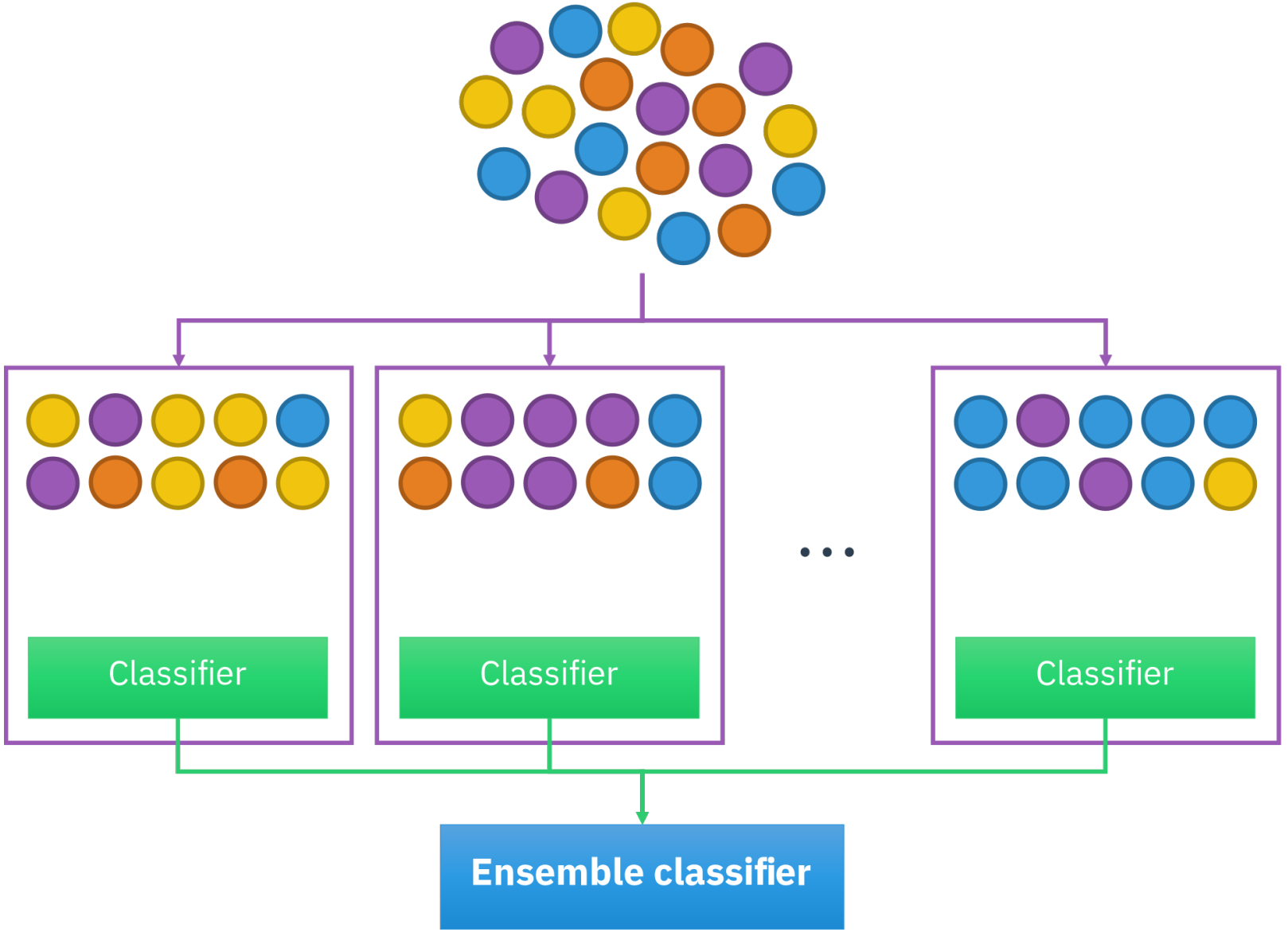
- Pros
  - Reduce bias → better models
  - Reduce variance → prevent models from overfitting
  - Less computation time → train models in parallel

Ensemble Learning

- Bagging

Bagging (**B**ootstrap **Agg**regat**ing**)

- Generate *K* distinct training sets by **sampling with replacement** from the original training set
  - Each training set consists of *N* examples
- For each training set (i.e., *K* rounds):
  - Run a machine learning algorithm on the *N* examples to get a hypothesis
- Predict the value of a new input by **aggregating the predictions** from all *K* hypotheses



Original Data

Bootstrapping

Aggregating

Bagging

Source: https://upload.wikimedia.org/wikipedia/commons/thumb/c/c8/Ensemble_Bagging.svg/2560px-Ensemble_Bagging.svg.png

Bagging (Bootstrap Aggregating) (cont.)

- Aggregation strategies
  - Classification problems
    - Voting (Majority Rule)
  - Regression problems

$$h(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^{K} h_i(\mathbf{x})$$ (Averaging)

Bagging (Bootstrap Aggregating) (cont.)

- Pros
  - **Reduce variance**
  - Can be applied to any class of model
  - Most commonly used with decision trees
    - Decision trees are unstable
      (a different set of examples)
  - Efficient because the hypotheses can be computed in parallel

Issues of Bagging Decision Trees

- *K* trees are highly correlated
  - If there is one attribute with a very high information gain, it is likely to be the root of most of the trees

Random Forests

- Random forest model is a form of decision tree bagging
- Take extra steps to make the ensemble of *K* trees more diverse to reduce variance
- **Key idea: randomly vary the attribute/feature choices**

Random Forests

- At each split point in constructing the tree, select a **random sampling of attributes**, and then compute which of those gives the highest information gain
  - If there are n attributes, a common default choice:
    - Classification problem: randomly picks $\sqrt{n}$ attributes
    - Regression problem: randomly picks 1/n attributes

Ref. T. K. Ho (1995). *Random Decision Forests.*
T. K. Ho *(1998). "The Random Subspace Method for Constructing Decision Forests"*

Out-of-Bag (OOB) Error Estimation in Training

- A reliable estimate of the generalization error
- The mean error on each example, using only the trees whose example set didn't include that particular example

OOB Error = the proportion of error that was incorrectly classified

Ensemble Learning

- Bagging
- Boosting

Boosting

- Idea
  - In general, the first hypothesis will classify some of the training examples correctly and some incorrectly
  - We would like the next hypothesis to do better on the misclassified examples
- Weighted training set
  - Each example has an associated weight $w_j \geq 0$ that describes how much the example should count during training
  - Examples that are misclassified will get increasingly larger weights
  - Examples that are correctly classified will get decreasingly lower weights

Boosting (cont.)

- A greedy algorithm
  - Initial: equal weights $w_j = 1$ for all the examples $\Rightarrow h_1$
  - New weighted training set:
    - Increase the weights of the misclassified examples while decreasing the weights of the correctly classified examples

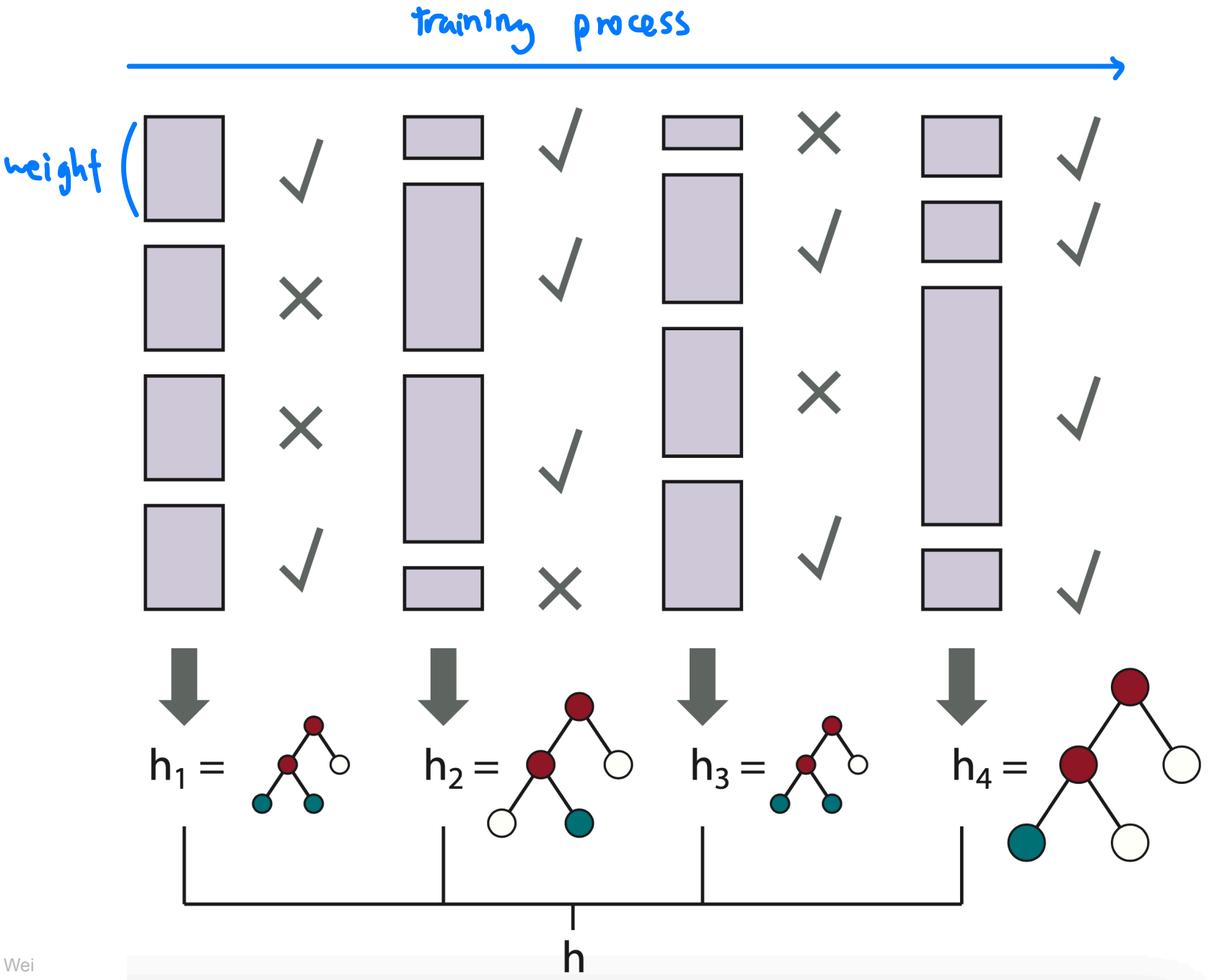
  $\Rightarrow$ generate hypothesis h_2
  - The process continues in this way until we have generated *K* hypotheses

Boosting (cont.)

- Aggregation strategies
  - The hypotheses that did better on their respective weighted training sets are given more voting weight

$$h(\mathbf{x}) = \sum_{i=1}^{K} z_i h_i(\mathbf{x})$$

where z_i is the weight of the ith hypothesis


training process
weight
h1 =
h2 =
h3 =
h4 =
h

Boosting Methods

- Adaboost (Adaptive Boosting)
  - Property
    - If the input learning algorithm *L* is a **weak learning algorithm**, then AdaBoost will return a hypothesis that classifies the training data perfectly for large enough *K*
      - *L* always returns a hypothesis with accuracy on the training set that is slightly better than random guessing (e.g., 50%+ϵ for boolean classification)
  - Start with one hypothesis h_1 and boost it with a sequence of hypotheses that pay special attention to the examples that the previous ones got wrong

AdaBoost

function ADABOOST(*examples*, L, K) **returns** a hypothesis

inputs: *examples*, set of N labeled examples $(x_1, y_1), \ldots, (x_N, y_N)$

L, a learning algorithm

K, the number of hypotheses in the ensemble

local variables: **w**, a vector of N example weights, initially all $1/N$

h, a vector of K hypotheses

z, a vector of K hypothesis weights

Calculate total error

function ADABOOST($examples, L, K$) **returns** a hypothesis

$\epsilon \leftarrow$ a small positive number, used to avoid division by zero

for $k = 1$ **to** K **do** // K hypotheses

$\mathbf{h}[k] \leftarrow L(examples, \mathbf{w})$

$\mathbf{w}$, a vector of N example weights, initially all $1/N$
$\mathbf{h}$, a vector of K hypotheses
$\mathbf{z}$, a vector of K hypothesis weights

$error \leftarrow 0$

for $j = 1$ **to** N **do** // *Compute the total error for* $\mathbf{h}[k]$

if $\mathbf{h}[k](x_j) \neq y_j$ **then** $error \leftarrow error + \mathbf{w}[j]$

if $error > 1/2$ **then break** from loop // breaks early if the current model is worse than random guessing

avoid division by 0

$error \leftarrow \min(error, 1 - \epsilon)$

for $j = 1$ **to** N **do** // *Give more weight to the examples* $\mathbf{h}[k]$ *got wrong*

if $\mathbf{h}[k](x_j) = y_j$ **then** $\mathbf{w}[j] \leftarrow \mathbf{w}[j] \cdot error/(1 - error)$

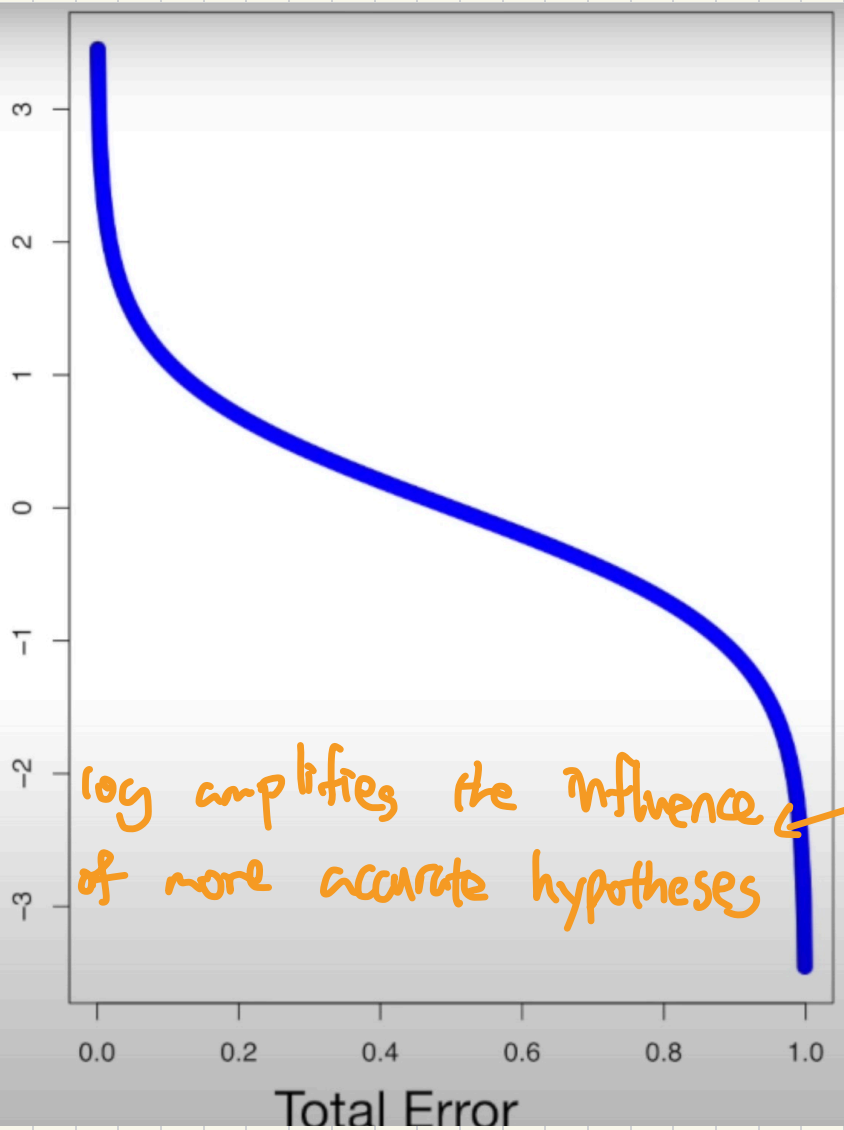
$\mathbf{w} \leftarrow$ NORMALIZE($\mathbf{w}$) // make all the weights add up to 1

$\mathbf{z}[k] \leftarrow \frac{1}{2} \log((1 - error)/error)$ // *Give more weight to accurate* $\mathbf{h}[k]$

return $Function(x) : \sum \mathbf{z}_i \mathbf{h}_i(x)$

See the next 3 slides

Hypothesis Weights, z (Amount of Say)

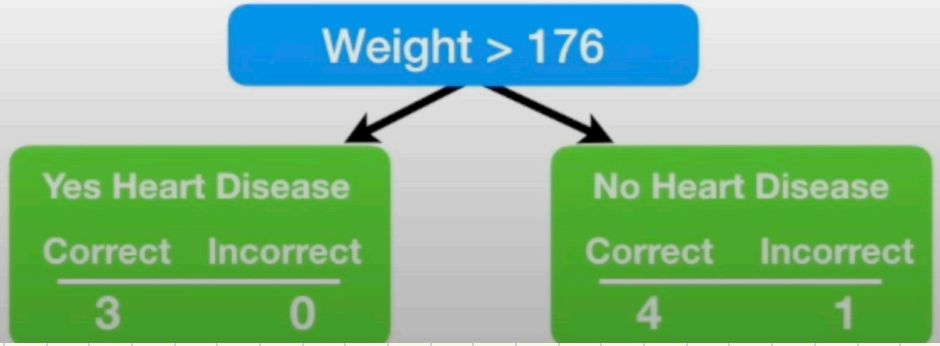


log amplifies the influence of more accurate hypotheses

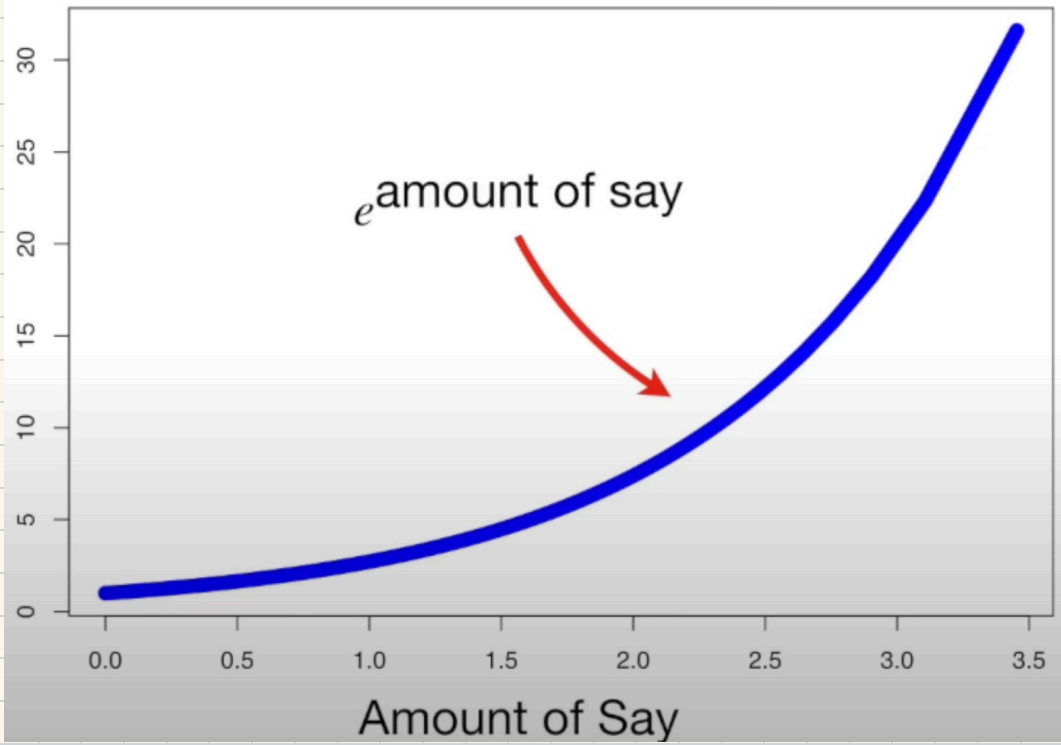
In practice a small error term is added to prevent this from happening.

↳ division by 0

$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$



Updating Sample Weights w (Misclassified Samples)



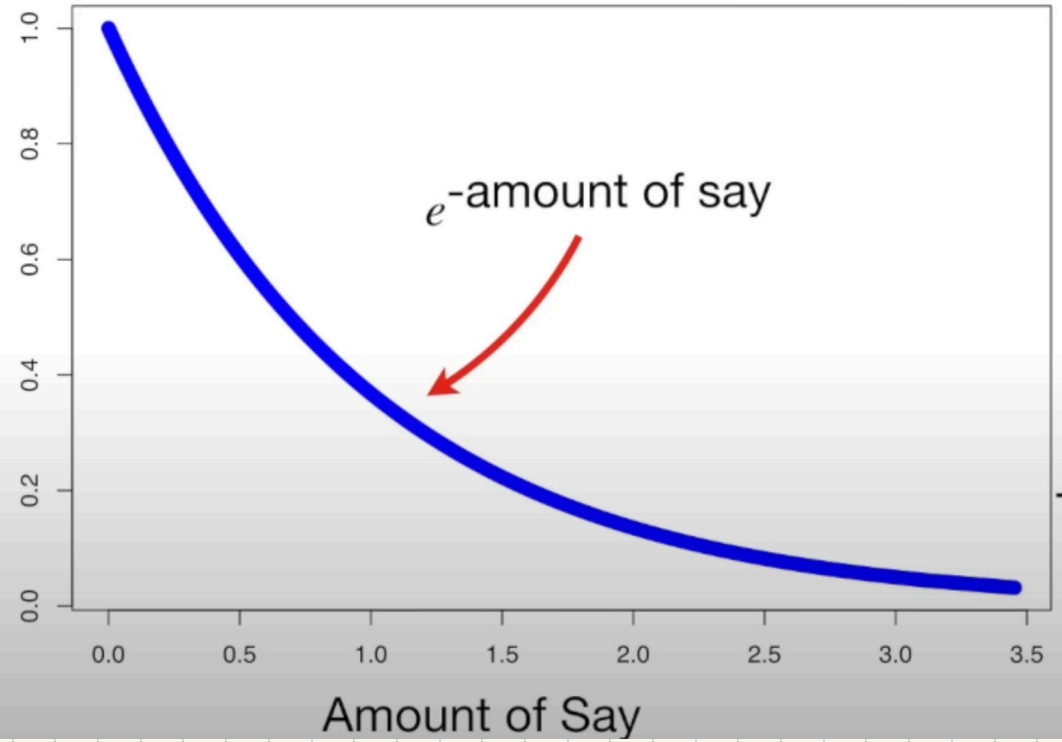
$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

$$= \frac{1}{8} e^{\text{amount of say}}$$

$$= \frac{1}{8} e^{0.97} = \frac{1}{8} \times 2.64 = 0.33$$

That means the new **Sample Weight** is **0.33**, which is *more* than the old one (**1/8** = **0.125**).

Updating Sample Weights w (Correctly Classified Samples)



$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{-amount of say}}$$

$$= \frac{1}{8} e^{\text{-amount of say}}$$

$$= \frac{1}{8} e^{-0.97} = \frac{1}{8} \times 0.38 = 0.05$$

The new **Sample Weight** is **0.05**, which is *less* than the old one (**1/8 = 0.125**).

$$e^{\alpha} \div e^{-\alpha} = e^{2\alpha} = \exp\left(2 \times \frac{1}{2} \log \frac{1-\text{Total Error}}{\text{Total Error}}\right) = \frac{1-\text{Total Error}}{\text{Total Error}}$$

Boosting Methods

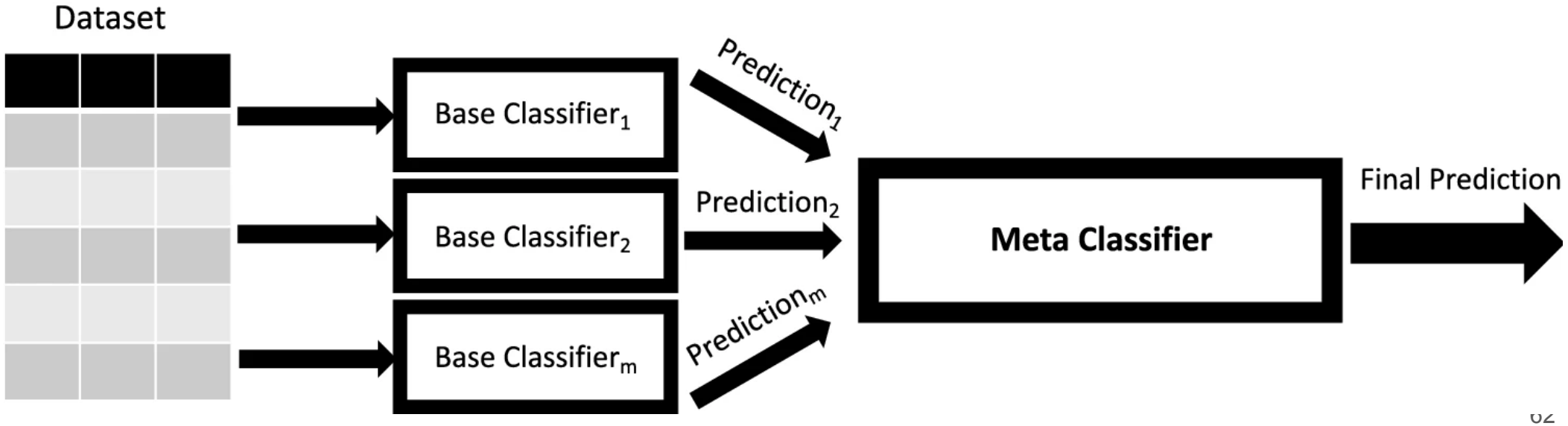
- Adaboost (Adaptive Boosting)
  - Start with one hypothesis h_1 and boost it with a sequence of hypotheses that pay special attention to the examples that the previous ones got wrong
- Gradient boosting (Gradient boosting machines, GBM)
  - Add new boosting hypotheses, which pay attention not to specific examples, but to the gradient between the right answers and the answers given by the previous hypotheses (loss)
  - XGBoost (eXtreme Gradient Boosting) package

Ensemble Learning

- Bagging
- Boosting
- Stacking

Stacking

- **Stacked generalization (stacking)** combines multiple base models from **different model classes** trained on **the same data**
- Pors
  - Reduce bias



Source: S. Shafieian et al. (2022) Multi-layer stacking ensemble learners for low footprint network intrusion detection

Unsupervised Learning

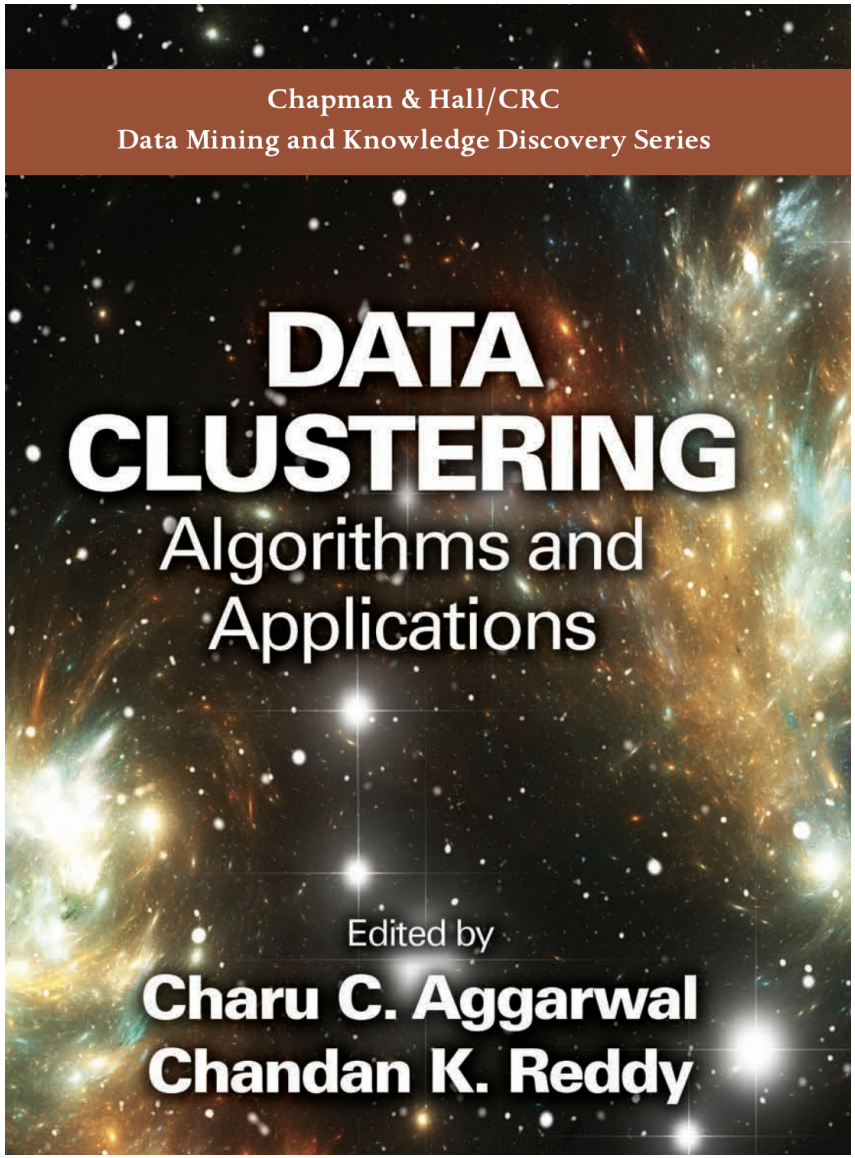
The problem is **unsupervised**
because the category labels are not given.

Unsupervised Learning

- Association rules
- Clustering
- Matrix factorization
- Dimension reduction
- …

Unsupervised Learning

- Association rules
- **Clustering**
- Matrix factorization
- Dimension reduction
- …

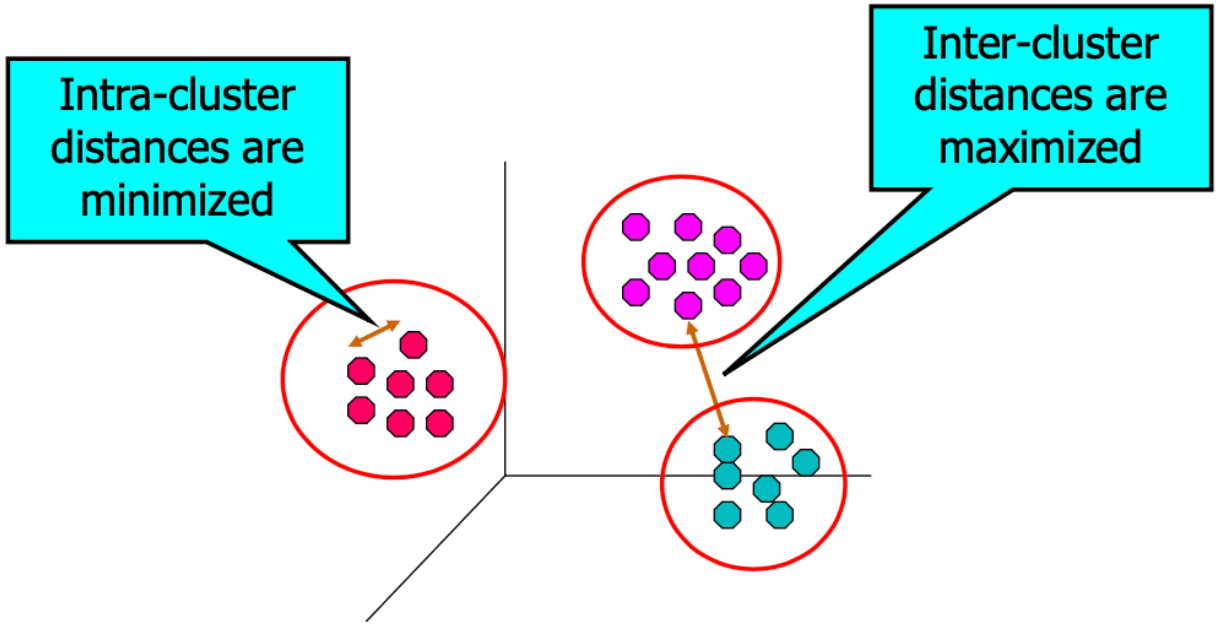


Clustering

hard clustering

soft clustering

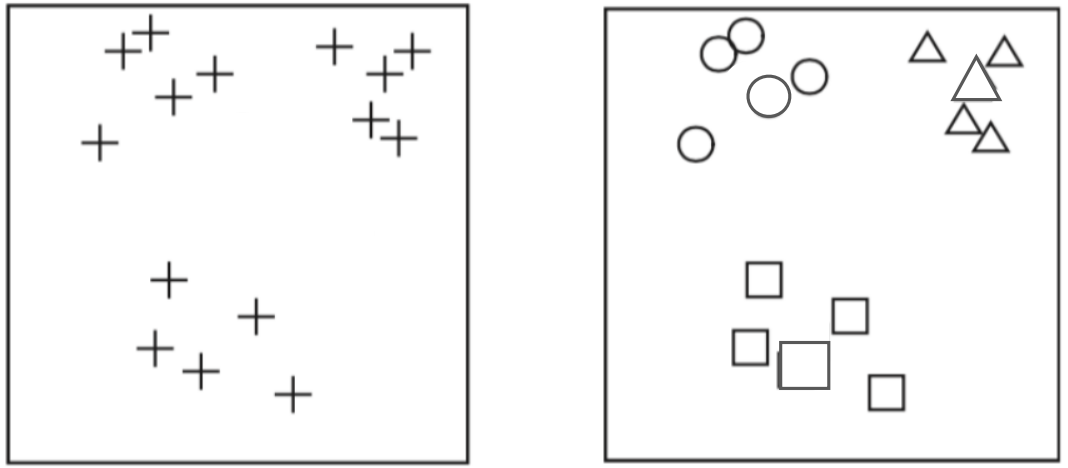
- Given a set of objects, place them in groups such that the objects in a group are similar (or related) to one another and different from (or unrelated to) the objects in other groups



Divide into K groups

Clustering: *K*-Means Clustering (hard clustering)

- e.g., *K*=3



K-Means Clustering

- Given a dataset $D = \{x_1,\ldots,x_n\}$, we partition data points into K disjoint clusters $C = \{C_1,\ldots,C_k\}$ s.t. the Sum of Squared Errors (SSE) is minimized

$$SSE(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \| x_i - c_k \|^2$$

where c_k is the centroid of C_k and $c_k = \frac{\sum_{x_i \in C_k} x_i}{|C_k|}$

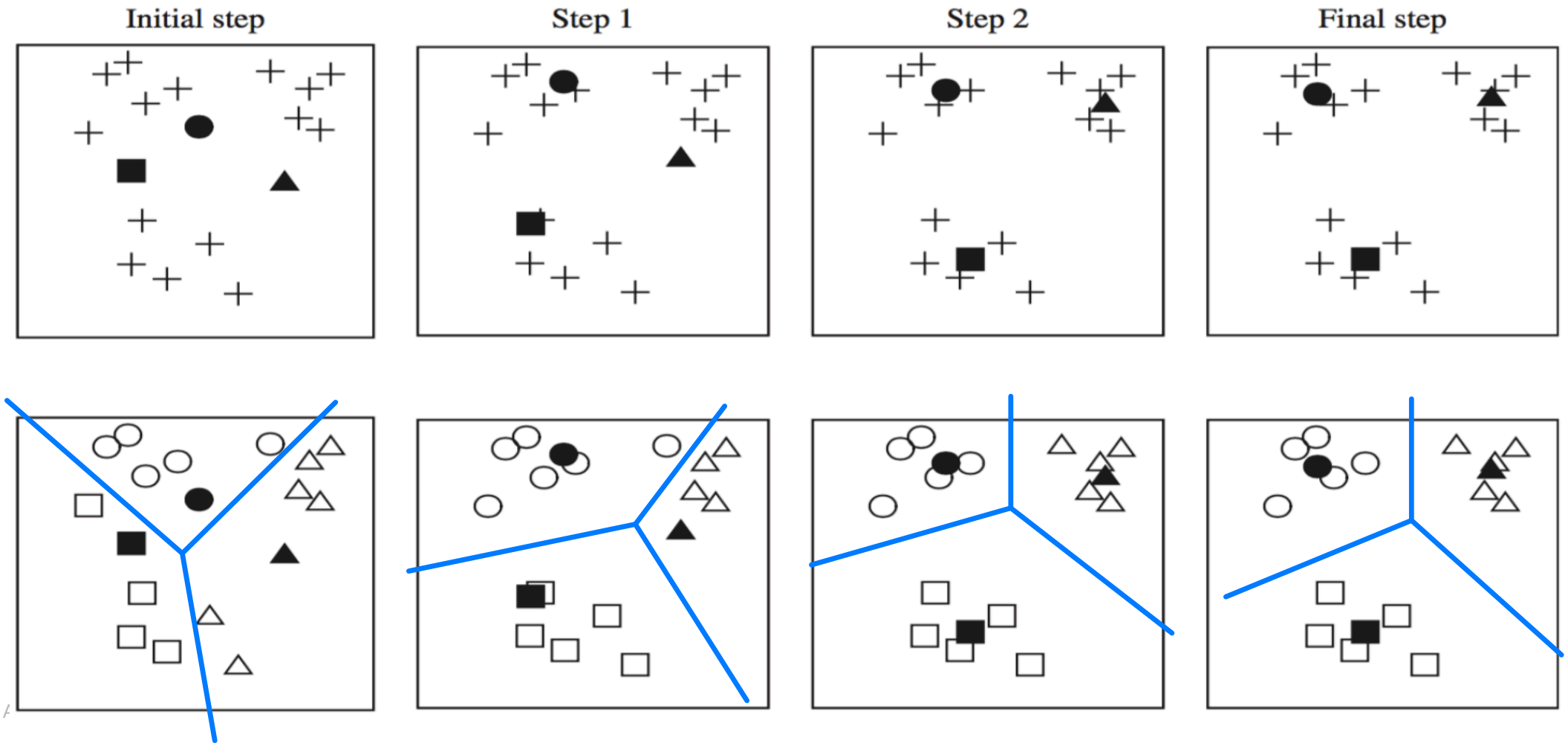
norm of C_k

Algorithm 13 K-Means Clustering

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** convergence criterion is met.

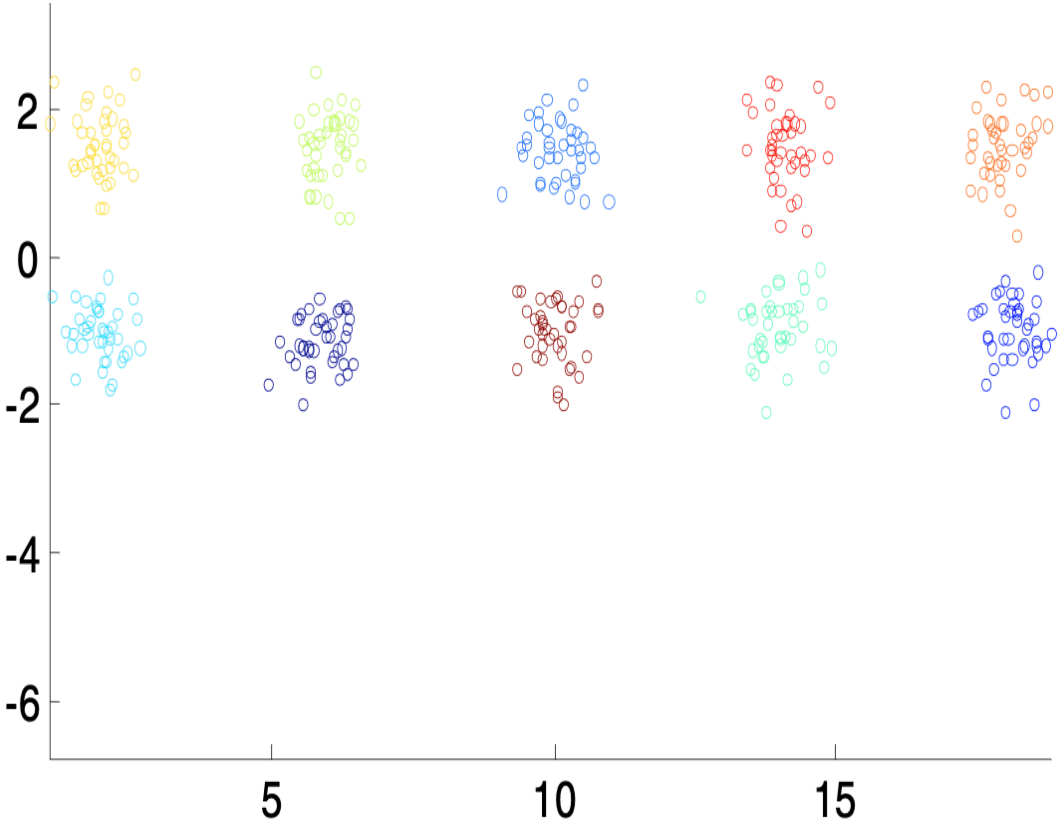
Euclidean distance (L_2 norm)

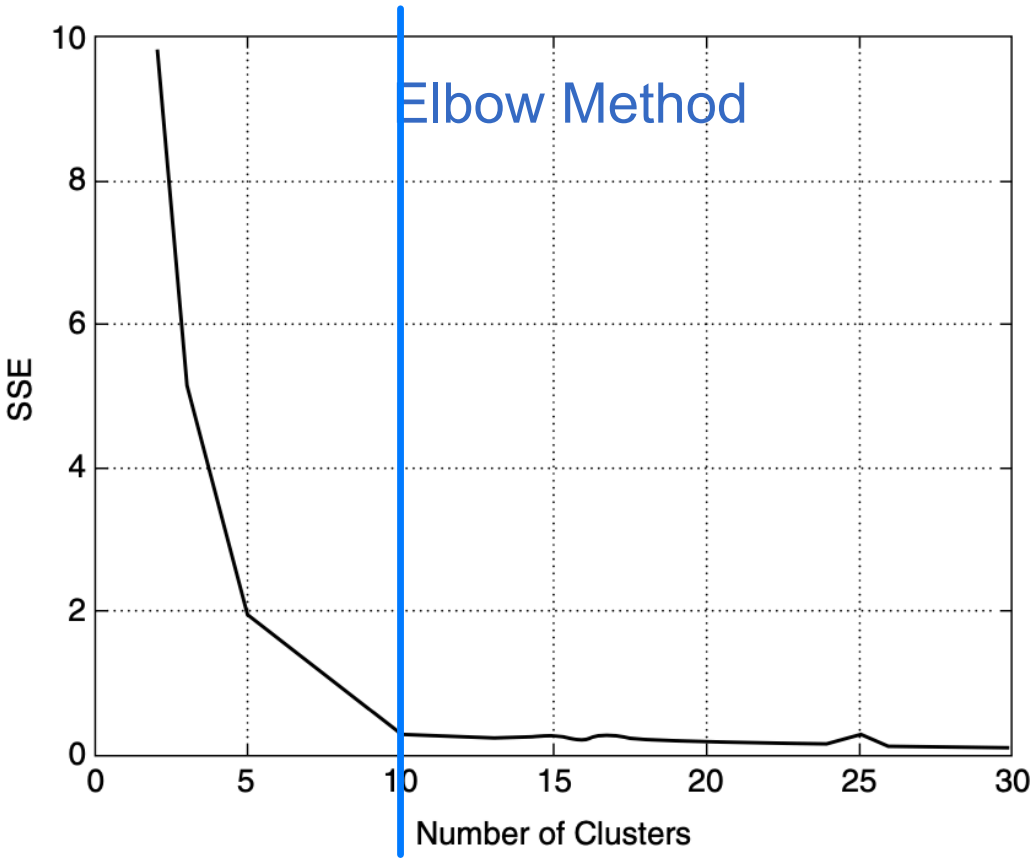
Example



Determining the Number of Clusters

- SSE can also be used to estimate the number of clusters

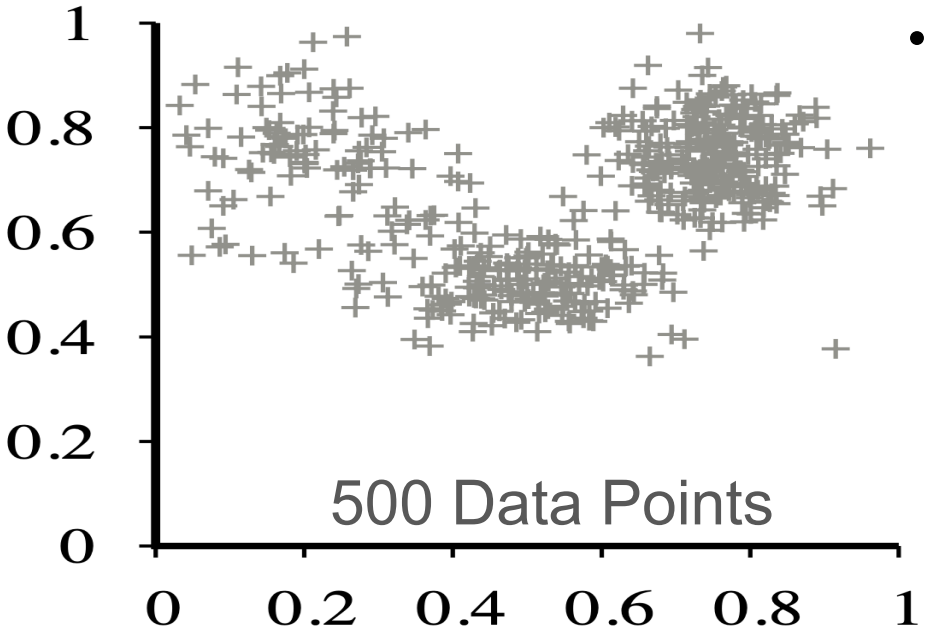




(soft clustering)

Clustering: Learning Mixture Models

- Clustering presumes that the data are generated from a mixture distribution, *P*
  - A mixture distribution has *k* components



- $\mathbf{x}$: the values of the attributes for a data point

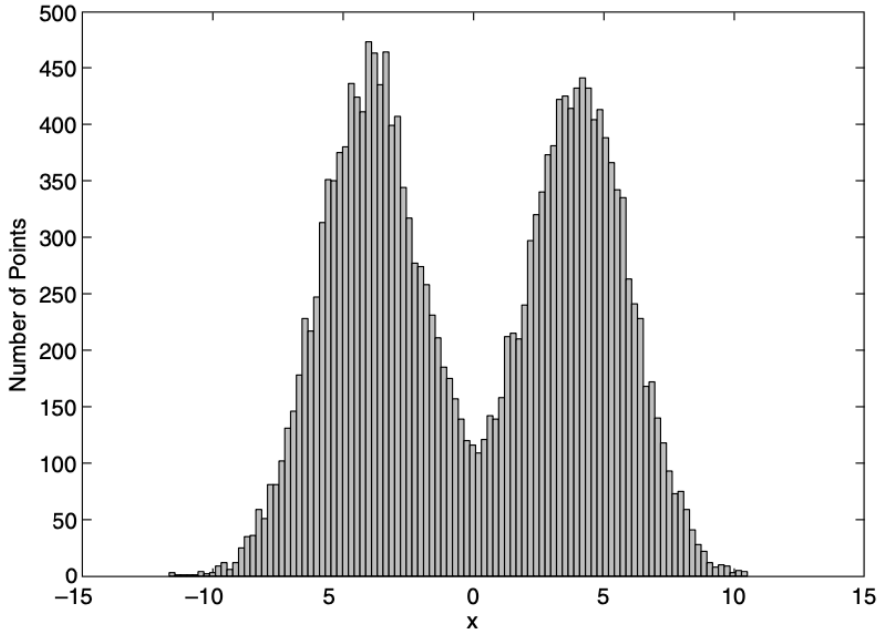
$$P\left(\mathbf{x}\right) = \sum_{i=1}^{k} P\left(C = i\right) P\left(\mathbf{x} \middle| C = i\right)$$

- *C*: the component (a random variable)

Gaussian Mixture Model (GMM)

Example: Mixtures of Gaussians

- Looks like a combination of two normal (Gaussian) distributions



(b) 20,000 points generated from the mixture model.

- One-dimensional Gaussian distribution:

$$prob(x_i|\Theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - μ: the mean
  - σ: the standard deviation

Find MLE (MAP) when some data is missing

* Issue: Marginalization is computationally expensive

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)}$$

Sol> Expectation-Maximization (EM) Algorithm

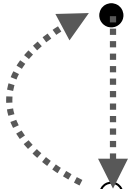
- Make an initial guess for the parameters
  - e.g., randomly select *k* data points to represent the cluster means
- Iteratively improves these estimates
  - **Expectation** step (E-step)
    - Assign each data object $\mathbf{x}_i$ to cluster C_j with the following probability based on Bayes rule

$$p(C_j \mid \mathbf{x}_i) = p(C_j)p(\mathbf{x}_i \mid C_j)/p(\mathbf{x}_i)$$

  - **Maximization** step (M-step)
    - Re-estimate model parameters
      - e.g., $\mu_j = \sum_{i=1}^{N} \mathbf{x}_i \frac{p(C_j \mid \mathbf{x}_i)}{\sum_{i=1}^{N} p(C_j \mid \mathbf{x}_i)}$

Example: EM Algorithm

- Cluster 20000 data objects into 2 clusters
- Initial guess: $\mu_1 = -2, \mu_2 = 3, \sigma_1 = \sigma_2 = 2$
- EM algorithm:



  - Expectation step
    - Assign each data object $\mathbf{x}_i$ to cluster C_j by comparing $p(C_j \mid \mathbf{x}_i)$
  - Maximization step
    - Compute new estimate for μ_1, μ_2

Example: EM Algorithm

$$p(x_i \mid C_1) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_i - m_1)^2}{2\sigma_1^2}}$$

$$= \frac{1}{\sqrt{2\pi}2} e^{-\frac{(0-(-2))^2}{2\times 2^2}} = 0.12$$

- Initial guess: $\mu_1 = -2, \mu_2 = 3, \sigma_1 = \sigma_2 = 2$
- Expectation step
  - Assign each data object $\mathbf{x}_i$ to cluster C_j by comparing $p(C_j \mid \mathbf{x}_i)$
  - e.g., $\mathbf{x}_i$ = 0,

$$p(C_1 \mid \mathbf{x}_i = 0) = \frac{p(C_1)p(\mathbf{x}_i = 0 \mid C_1)}{p(\mathbf{x}_i = 0)} = \frac{0.5p(\mathbf{x}_i = 0 \mid C_1)}{0.5p(\mathbf{x}_i = 0 \mid C_1) + 0.5p(\mathbf{x}_i = 0 \mid C_2)}$$

$$= \frac{0.12}{0.12 + 0.06} = 0.66$$

$$p(C_2 \mid \mathbf{x}_i = 0) = \frac{p(C_2)p(\mathbf{x}_i = 0 \mid C_2)}{p(\mathbf{x}_i = 0)} = \frac{0.5p(\mathbf{x}_i = 0 \mid C_2)}{0.5p(\mathbf{x}_i = 0 \mid C_1) + 0.5p(\mathbf{x}_i = 0 \mid C_2)}$$

$$= \frac{0.06}{0.12 + 0.06} = 0.33$$

Example: EM Algorithm

$$p(x_i \mid C_1) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x_i - m_1)^2}{2\sigma_1^2}}$$

$$= \frac{1}{\sqrt{2\pi}2} e^{-\frac{(0-(-2))^2}{2\times 2^2}} = 0.12$$

- Initial guess: $\mu_1 = -2, \mu_2 = 3, \sigma_1 = \sigma_2 = 2$
- Expectation step
  - Assign each data object $\mathbf{x}_i$ to cluster C_j by comparing $p(C_j \mid \mathbf{x}_i)$
  - e.g., $\mathbf{x}_i$ = 0, $p(C_1 \mid \mathbf{x}_i = 0) = 0.66, p(C_2 \mid \mathbf{x}_i = 0) = 0.33$
- Maximization step
  - Compute new estimate for μ_1, μ_2

$$\mu_1 = \sum_{i=1}^{20000} \mathbf{x}_i \frac{p(C_1 \mid \mathbf{x}_i)}{\sum_{i=1}^{20000} p(C_1 \mid \mathbf{x}_i)} \text{ and } \mu_2 = \sum_{i=1}^{20000} \mathbf{x}_i \frac{p(C_2 \mid \mathbf{x}_i)}{\sum_{i=1}^{20000} p(C_2 \mid \mathbf{x}_i)}$$