24ASD641 Pattern Recognition

Dr. Jayakrishnan Anandakrishnan
Assistant Professor
Amrita School of Computing
Amrita Vishwa Vidyapeetham, Coimbatore

Google Scholar ORCID Web of Science

Course Objectives and Outcomes

Objectives

- Identify patterns, regularities, or structures in data to make informed decisions.
- Focus on computational properties of patterns and algorithms used to process them.

Outcomes

- CO01: To get an idea about pattern recognition with suitable examples
- CO02: To gain knowledge about parametric classification methods using Bayesian decision making approach
- CO03: To apply nonparametric techniques such as nearest neighbor, adaptive discriminant functions, and decision regions based on minimum squared error
- CO04: To gain knowledge about nonmetric methods, classification trees, and some resampling methods
- CO05: To study and apply various clustering methods

Course Units and References

Unit I: Introduction – pattern recognition systems, design cycle, learning/adaptation, applications, statistical decision theory, examples in pattern recognition and image processing.
Unit II: Parametric methods – Bayes theorem, Bayesian decision making, Gaussian case, discriminant functions, decision boundaries/regions, dimensionality problems, ROC curves, ML classification.
Unit III: Nonparametric methods – histograms, density estimation, mixture densities, kernel/window estimators, nearest neighbor techniques, adaptive discriminant functions, minimum squared error methods.
Unit IV: Nonmetric methods – decision trees, CART, algorithm-independent ML, bias-variance, jackknife and bootstrap resampling.
Unit V: Clustering – unsupervised learning, criterion functions, hierarchical (single, complete, average, Ward's), partitional (Forgy's, k-means).
References:

1. Richard O. Duda, Peter E. Hart, David G. Stork, *Pattern Classification*, 2nd Ed., Wiley, 2003.
2. Earl Gose, Richard Johnsonbaugh, Steve Jost, *Pattern Recognition and Image Analysis*, PHI, 2002.

Outline

1. Linear Discriminant Functions
2. Minimum Squared Error Discriminant Functions
3. Non-metric Methods

Linear Discriminant Functions

- In parametric estimation, we assumed that the forms for the underlying probability densities were known and used the training samples to estimate the values of their parameters.
- Instead, assume that the proper forms for the discriminant functions are known, and use the samples to estimate the values of parameters of the classifier.
- None of the various procedures for determining discriminant functions requires knowledge of the forms of underlying probability distributions, so-called nonparametric approach.
- Linear discriminant functions are relatively easy to compute and estimate the form using training samples.

Linear Discriminant Functions

Parametric vs. Nonparametric

Parametric: Assume the data follows a probability distribution (Gaussian/Normal), and just estimate parameters (mean, variance).
Nonparametric: No assumption of any distribution. Instead, directly learn from the training data itself.

Parametric Case

Training Data:

Fruit	Weight (grams)
Apple	160
Apple	170
Banana	120
Banana	130

Estimated Parameters:

- Apple: Mean $\mu = 165$, Variance $\sigma^2 = 25$
- Banana: Mean $\mu = 125$, Variance $\sigma^2 = 25$

New fruit weight: $x = 150$ grams

Classification Using Gaussian Likelihood

$$P(x|\text{class}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

For Apple:

$$P(150|\text{Apple}) = \frac{1}{\sqrt{2\pi(25)}} \exp\left(-\frac{(150-165)^2}{2(25)}\right) =\approx 0.0026$$

For Banana:

$$P(150|\text{Banana}) = \frac{1}{\sqrt{2\pi(25)}} \exp\left(-\frac{(150-125)^2}{2(25)}\right) =\approx 1.5 \times 10^{-6}$$

Conclusion: The fruit is more likely to be an **Apple**.

Non-parametric Case

- No assumption of Gaussian or any probability distribution
- Instead, we can use a function $g(x)$, ie, a linear discriminant function

Simple Rule:

$$g(x) = ax + b$$

$$\begin{cases} \text{If } g(x) > 0, & \text{then classify as Apple} \\ \text{If } g(x) < 0, & \text{then classify as Banana} \end{cases}$$

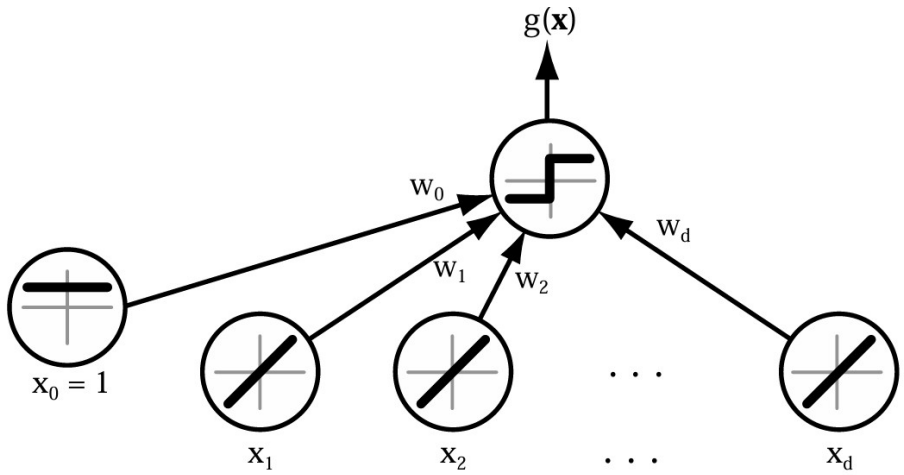
Discriminant Function for Classification

- A discriminant function helps in decision-making.
- For example: Given a point X, assign it to class ω_1 or class ω_2 depending on the value of a function.

$$\begin{cases} \text{If function} > 0, & \text{assign to class } \omega_1 \\ \text{If function} < 0, & \text{assign to class } \omega_2 \end{cases}$$

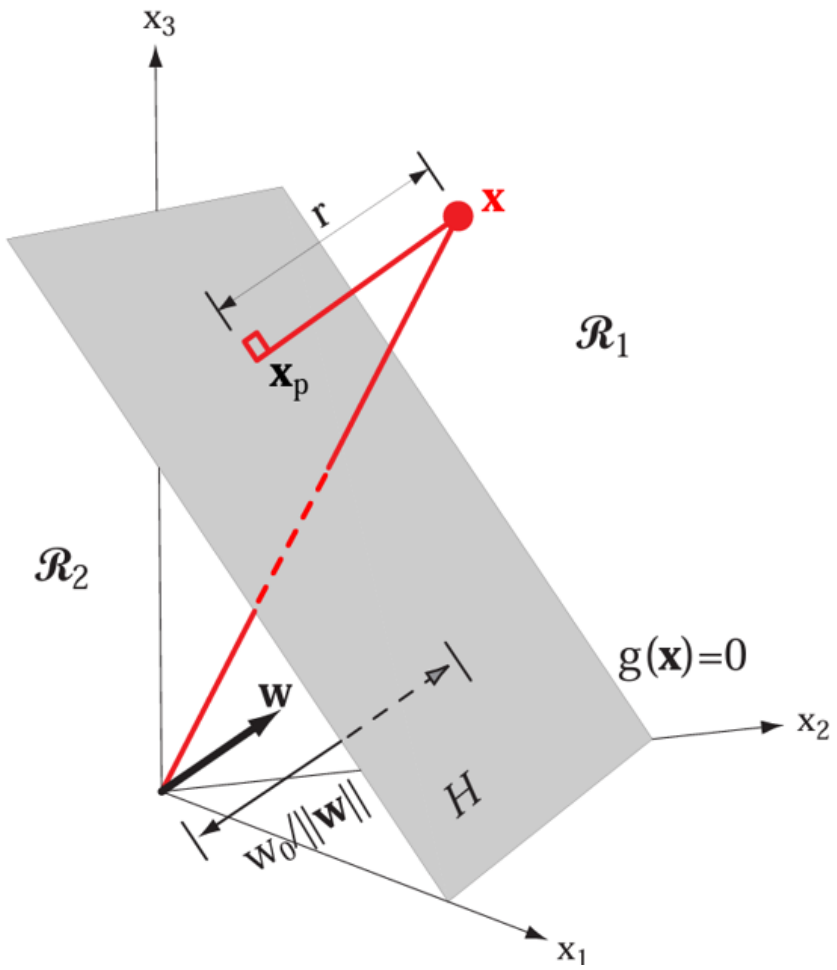
Linear Classifier

- **w** is the weight vector
- w_0 is the bias or threshold weight
- $g(x) = 0$ defines the decision surface
- Decision surface separates classes
- Two-category case
- Multi-category case



$$g(x) = \mathbf{w}^T x + w_0$$

Linear Classifier



Two-Category Classification

A two-category case can be defined as a decision between class ω_1 and class ω_2.

$$\begin{cases} g(x) > 0 & \Rightarrow \text{Class } \omega_1 \\ g(x) < 0 & \Rightarrow \text{Class } \omega_2 \end{cases}$$

Given:

$$g(x) = \mathbf{w}^T x + w_0 > 0 \Rightarrow \text{Class } \omega_1$$

This implies:

$$\mathbf{w}^T x > -w_0$$

So the decision rule becomes:

$$\begin{cases} \mathbf{w}^T x > -w_0 & \Rightarrow \text{Class } \omega_1 \\ \mathbf{w}^T x < -w_0 & \Rightarrow \text{Class } \omega_2 \end{cases}$$

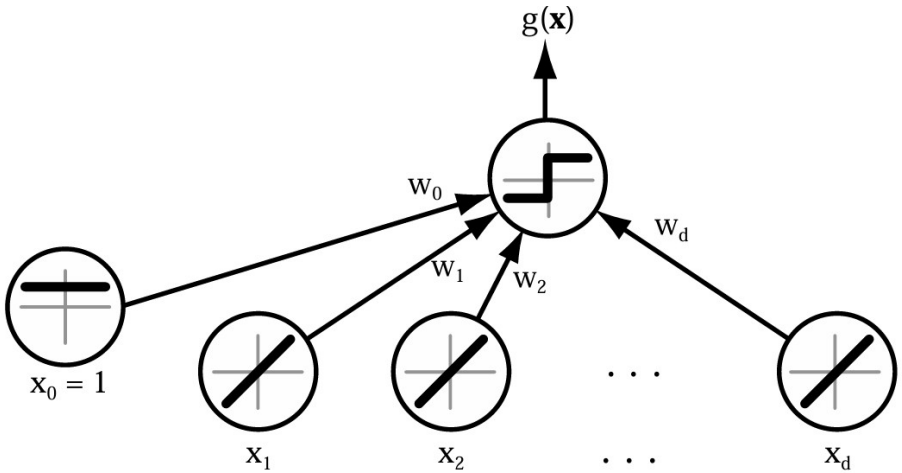
If $g(x) = 0$, the point lies on the decision boundary.

Linear Classifier

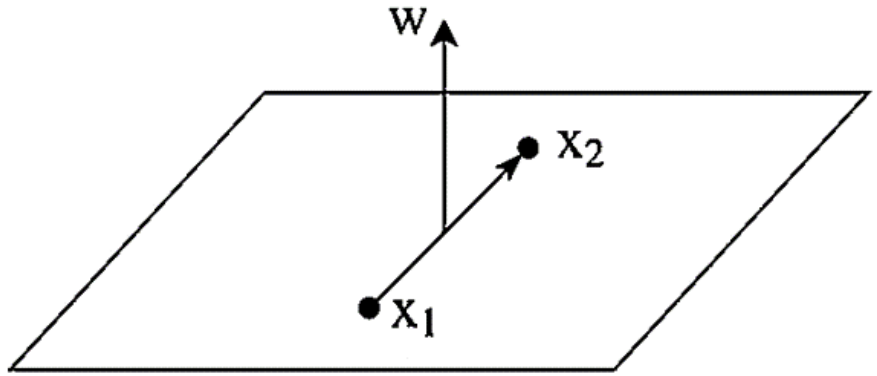
$$y = f(g(x))$$

Where:

$$y = \begin{cases} +1 & \text{if } g(x) > 0 \\ -1 & \text{if } g(x) < 0 \end{cases}$$



The equation $g(x) = 0$ defines the decision surface that separates points assigned to the categories ω_1 and ω_2.



If x_1 and x_2 are both on the decision surface, then...

Geometry of the Decision Surface

If $g(x_1) = g(x_2) = 0$, then both x_1 and x_2 lie on the decision surface.

$$\mathbf{w}^T x_1 + w_0 = w^T x_2 + w_0 = 0$$

Subtracting the two equations:

$$w^T (x_1 - x_2) = 0$$

This indicates that $\mathbf{w}$ is orthogonal (normal) to any vector lying in the hyperplane defined by the decision surface.

How $g(x)$ is Related to Distance Measure

- x_p is the projection of x on hyperplane H
- r is positive if x is on the positive side, negative if on the negative side

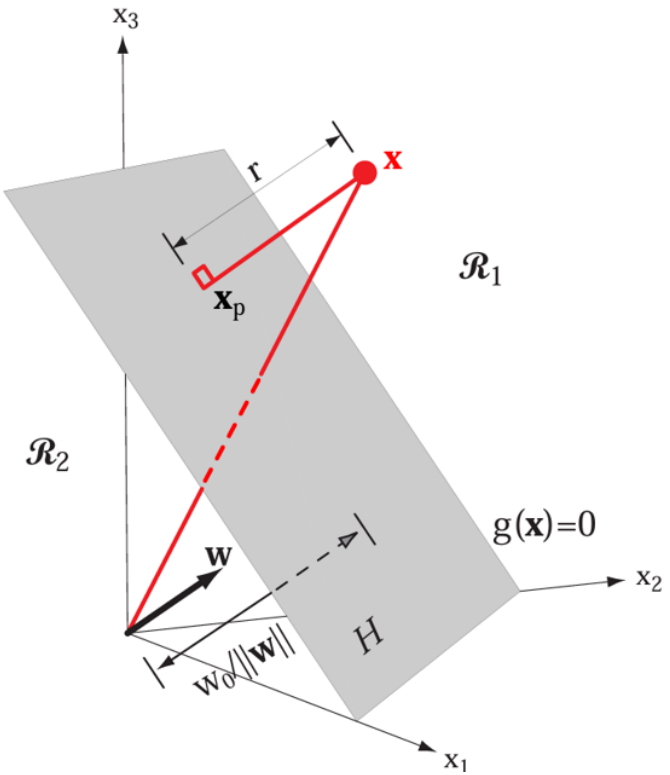
$$x = x_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

$$g(x) = \mathbf{w}^T x + w_0 = \mathbf{w}^T\left(x_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + w_0$$

$$= \mathbf{w}^T x_p + r\mathbf{w}^T\left(\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + w_0 = \mathbf{w}^T x_p + r\frac{\|\mathbf{w}\|^2}{\|\mathbf{w}\|}$$

$$= \mathbf{w}^T x_p + r\|\mathbf{w}\| + w_0$$

If x_p lies on the decision surface, then $\mathbf{w}^T x_p + w_0 = 0$, so:

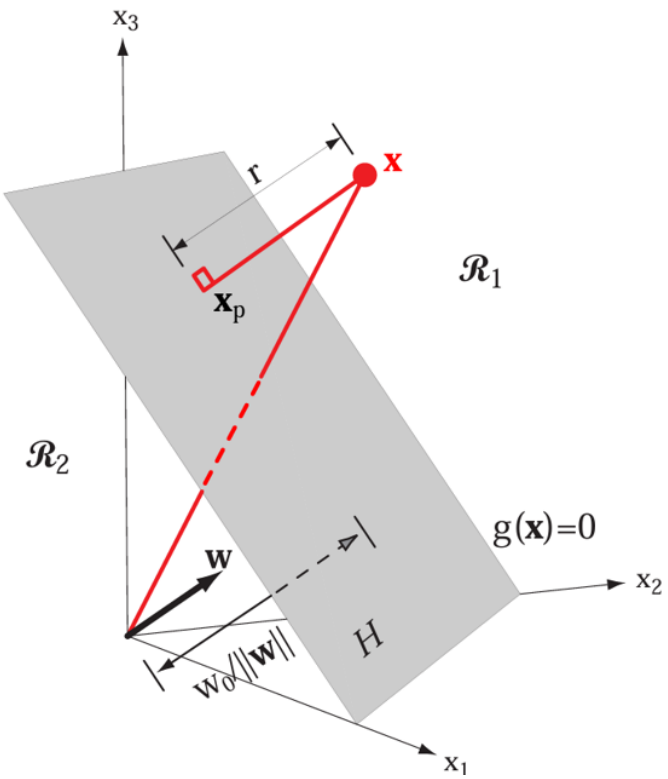


How $g(x)$ is Related to Distance Measure

- The distance from the origin to the hyperplane H is:

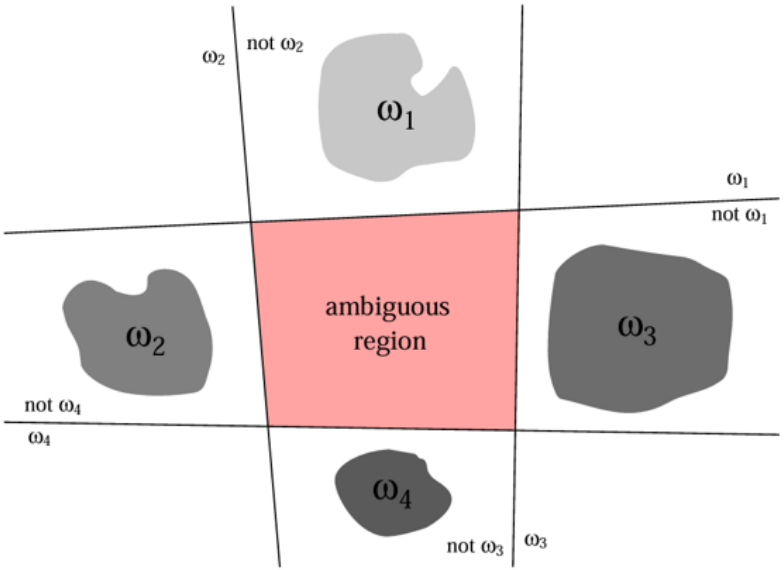
$$\frac{w_0}{\|\mathbf{w}\|}$$

- If $w_0 > 0$, the origin lies on the positive side of H
- If $w_0 < 0$, the origin lies on the negative side of H
- If $w_0 = 0$, then $g(x) = \mathbf{w}^T x$ is homogeneous and the hyperplane passes through the origin

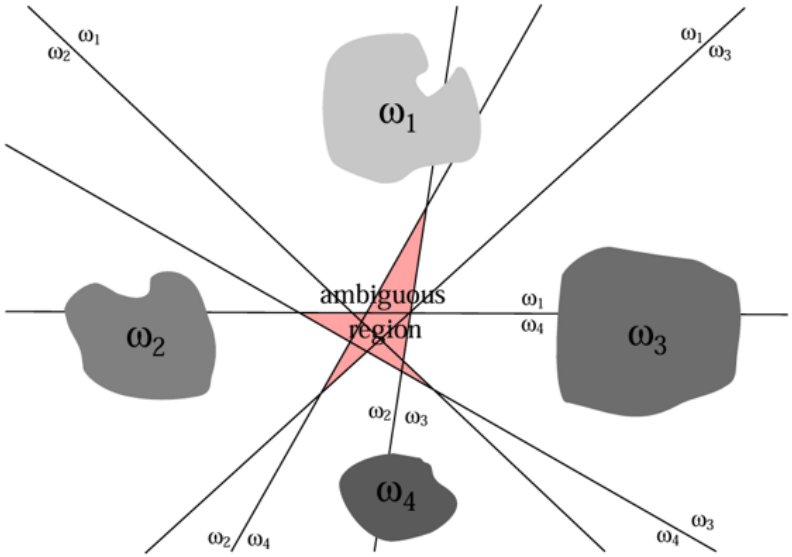


Multi Class Case

- c two-class problem (1 vs the rest)



- $c(c-1)/2$ linear discriminant one for every pair of classes



- Both lead to an ambiguous region, where if a sample falls the its difficulty in deciding
- How can it be solved?

Multi Class Case: Solution

- Can be solved by considering a linear discriminant function for each individual class

Let us consider, a linear discriminant function for the i^{th} class is:

$$g_i(x) = w_i^T x + w_{i0}, \quad i = 1, \ldots, C$$

We assign x to w_i if $g_i(x) > g_j(x)$ for $j \neq i$.

- In this case, the resulting classifier is known as a **Linear Machine**.
- The linear machine divides the feature space into C decision regions:

$$R_1, R_2, \ldots, R_C$$

with $g_i(x)$ being the largest discriminant if x lies in the region R_i.

Multi Class Case: Point on Decision Boundary

- For two contiguous regions R_i and R_j, and if point lies on the decision boundary then:

$$g_i(x) = g_j(x)$$

$$\Rightarrow \; w_i^T x + w_{i0} = w_j^T x + w_{j0}$$

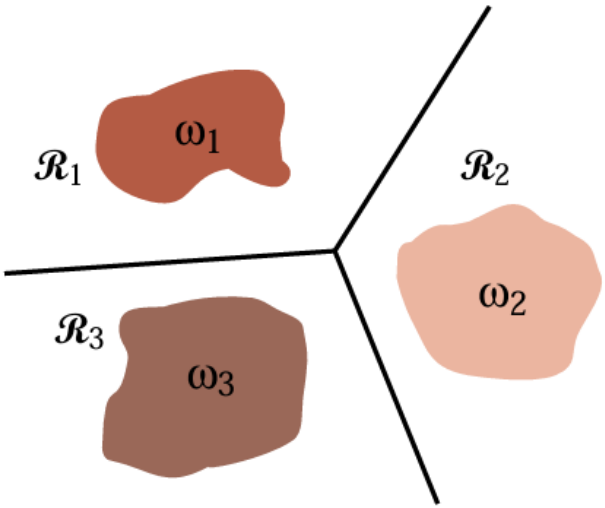
$$\Rightarrow \; (w_i - w_j)^T x + (w_{i0} - w_{j0}) = 0$$

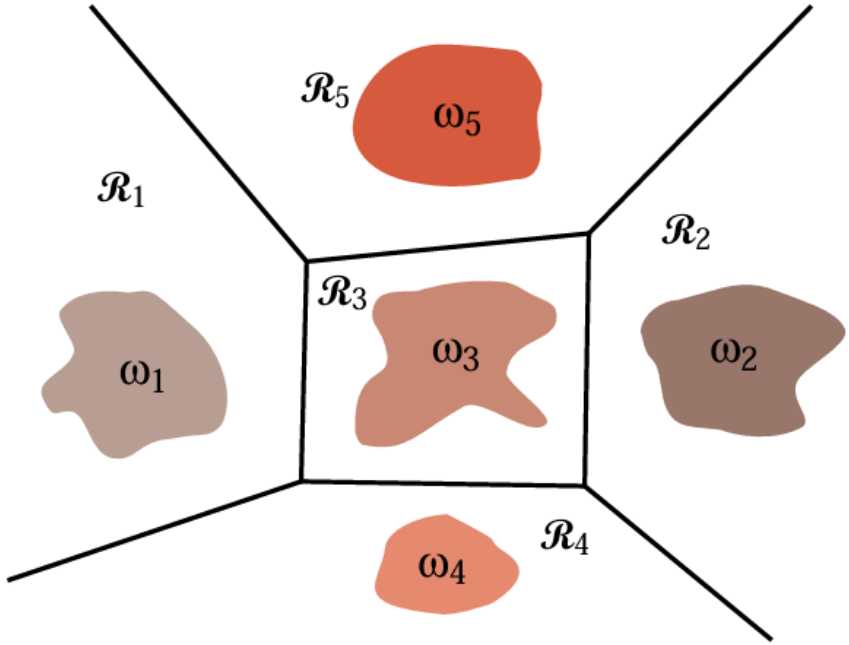
It shows that $(w_i - w_j)$ is normal to H_{ij} (separating plane between class i & j).

So the algebraic distance from x to H_{ij} is

$$r = \frac{g_i(x) - g_j(x)}{\|w_i - w_j\|}$$

Multi Class Case: Regions





Minimum Squared Error Discriminant Functions

- Possibilities to extract the weight vector all at once without any iterative process
- Neural network has an iterative process to learn the weight a
- Can you learn a in a single step?

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{12} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

- Let X be an $n \times d$ matrix
- Now apply augmentation. Why augmentation?

Minimum Squared Error Discriminant Functions

- Because of the bias (intercept) term need to be added in linear discriminant functions
- The linear discriminant function for a single data point x_i

$$g(x_i) = a_0 + a_1 x_{i1} + a_2 x_{i2} + \cdots + a_d x_{id}$$

- What is a_0, its the augmentation to wright vector a
- Augmented vector Y, whose each data point y_i has an extra 1 at the beginning

$$\mathbf{Y} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1d} \\ 1 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nd} \end{pmatrix}$$

‘

Minimum Squared Error Discriminant Functions

$$\begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1d} \\ 1 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nd} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

- Now you can write as a linear equation $Ya = b$, and $a = Y^{-1}b$
- $a = Y^{-1}b$ is only possible if Y is a square, invertible matrix
- is it true for the real world?
- In the real world, usually you have more data points n than features d
- How to get the first linear equation?
- Multiply the first row of Y by column wright a equal to the first element in b
- Now, we don't have the exact weight vector a, then the error will happen
- Error $e = Ya - b$

Minimum Squared Error Discriminant Functions

- If error e is there, then we can write the error as the sum of squared errors, and the function is given as

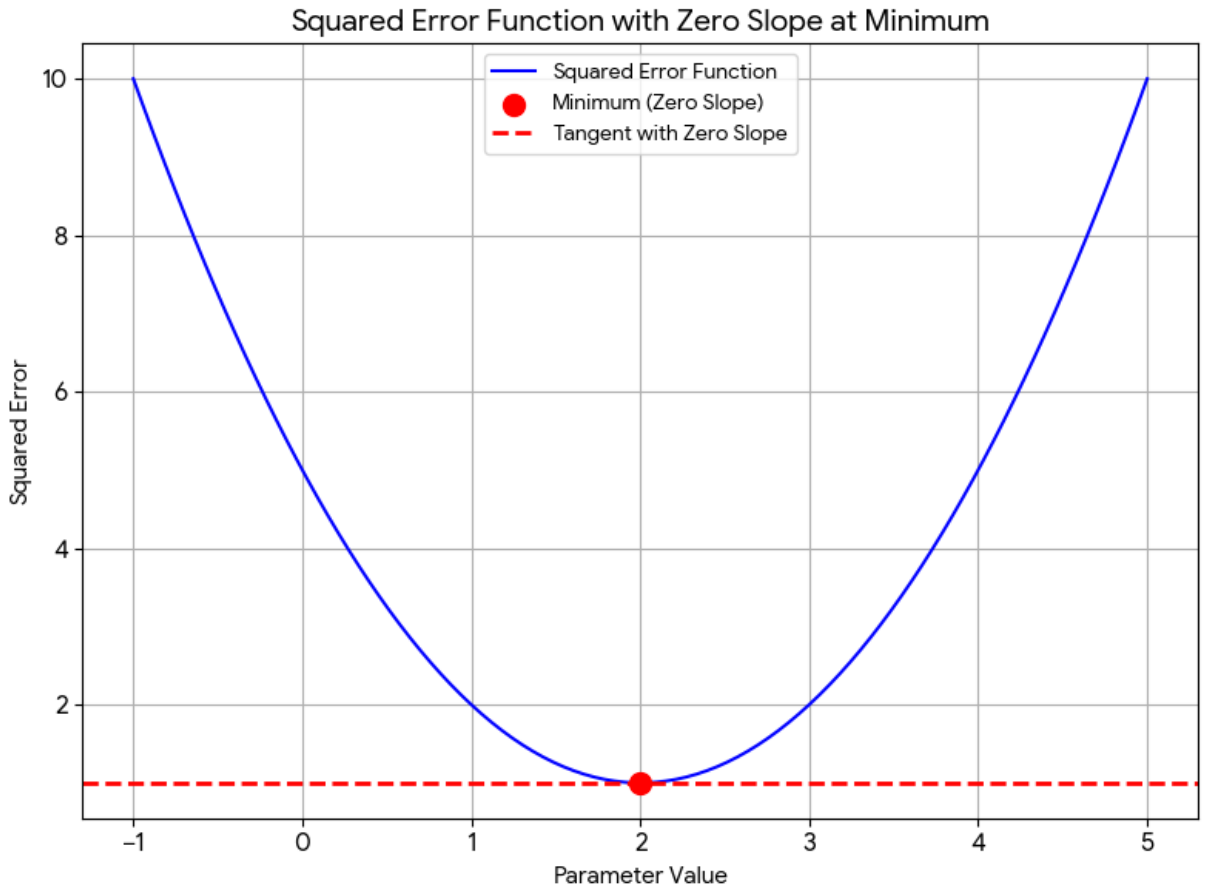
$$J_s(a) = \|Ya - b\|^2 = \sum (a^T y_i - b_i)^2$$

  Note: a is the weights, including bias, y is the input sample with a bias term, b is the target
- Now we have to minimize this error, take the gradient, and equate it to zero provides a simple solution
- Differentiate w.r.t a

$$\nabla J_s(a) = \sum_{i=1}^{n} 2(a^T y_i - b_i) y_i \Rightarrow 2Y^T(Ya - b)$$

Minimum Squared Error Discriminant Functions

- When you equate the gradient of the error function to zero, you are finding the point where the function's slope is flat. For a convex function like the squared error, this flat point is the global minimum.



Minimum Squared Error Discriminant Functions

- Equate it to zero to get a simple solution

$$2Y^T(Ya - b) = 0$$

$$Y^T Ya = Y^T b$$

$$a = (Y^T Y)^{-1} Y^T b$$

- Now you can write a as

$$a = Y^\dagger b, \quad \text{where } Y^\dagger = (Y^T Y)^{-1} Y^T$$

- $Y^\dagger$ is called the pseudoinverse
- Indicating if we know b, we can compute our solution weight a from the pseudoinverse
- Regularized pseudoinverse is given as

$$Y^\dagger \equiv \lim_{\epsilon \to 0} (Y^T Y + \epsilon I)^{-1} Y^T$$

Minimum Squared Error Discriminant Functions

Suppose we have the following two-dimensional points for two categories: $\omega_1 : (1,2)^t, (2,0)^t$ and $\omega_2 : (3,1)^t, (2,2)^t$. Construct a classifier with a pseudoinverse.

$$\omega_1 : \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}2\\0\end{bmatrix} \quad \omega_2 : \begin{bmatrix}3\\1\end{bmatrix}, \begin{bmatrix}2\\3\end{bmatrix}$$

Augmented vectors are:

$$y_1 = [1,1,2], \quad y_2 = [1,2,0], \quad y_3 = [-1,-3,-1], \quad y_4 = [-1,-2,-3]$$

Matrix $\mathbf{Y}$ is:

$$\mathbf{Y} = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 0 \\ -1 & -3 & -1 \\ -1 & -2 & -3 \end{pmatrix}$$

The pseudoinverse of $\mathbf{Y}$ is given by:

$$\mathbf{Y}^+ = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T$$

Minimum Squared Error Discriminant Functions

$$\mathbf{Y}^{+} = \begin{pmatrix} \frac{5}{4} & \frac{13}{12} & \frac{3}{4} & \frac{7}{12} \\ -\frac{1}{2} & -\frac{1}{6} & -\frac{1}{2} & -\frac{1}{6} \\ 0 & -\frac{1}{3} & 0 & -\frac{1}{3} \end{pmatrix}$$

We arbitrarily let all the margins/labels be equal, i.e.,

$$\mathbf{b} = (1, 1, 1, 1)^t = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Our solution is:

$$\mathbf{a} = \mathbf{Y}^{+}\mathbf{b} = \begin{pmatrix} \frac{11}{3} \\ -\frac{4}{3} \\ -\frac{2}{3} \end{pmatrix}$$

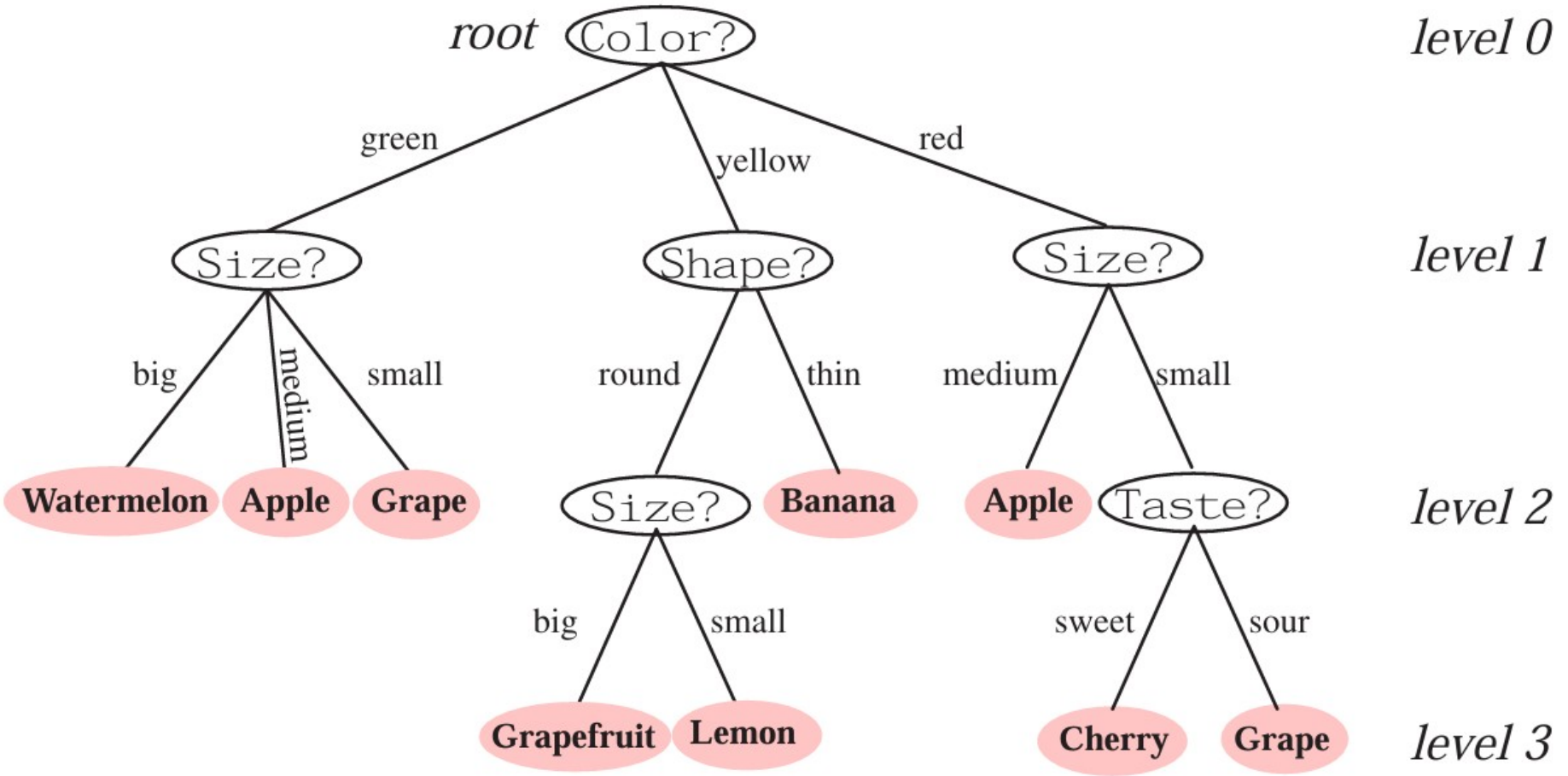
Introduction to Non-Metric Methods

- Previous pattern recognition methods involved real-valued feature vectors with clear metrics.
- Are there instances of data without clear metrics?
- For Nominal data: Data that are discrete and without any natural order notion of similarity or even ordering.
- For example, eye Colors. It can be Brown, Blue, or Green. You can count how many people have brown eyes, but you can't perform a mathematical operation like finding the average eye color.
- Solution: decision trees and string grammars.

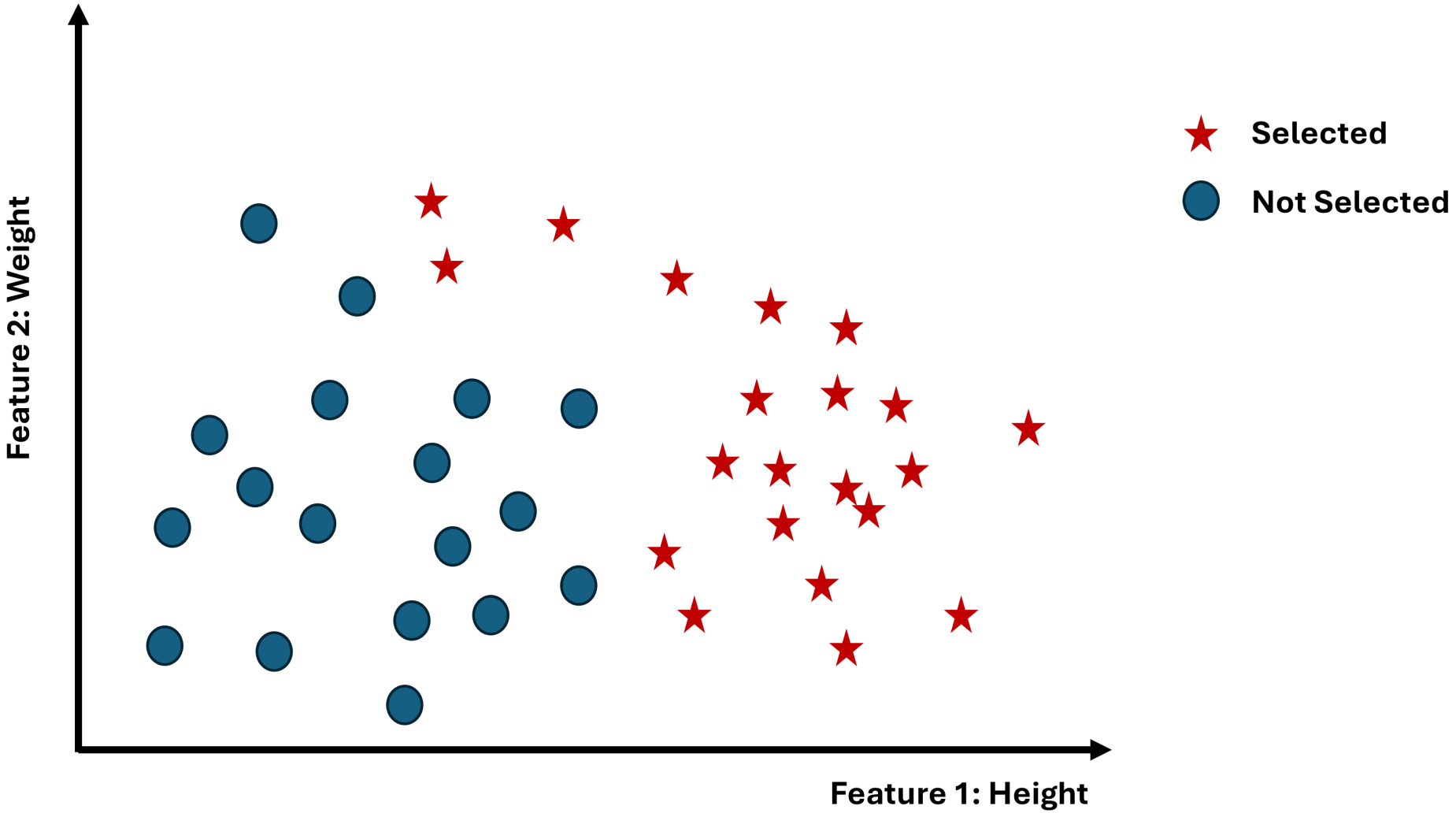
Introduction to Non-Metric Methods

- I am thinking of a fruit. Ask me up to 20 yes/no questions to determine what is the fruit am thinking of.
- How did you ask the questions?
- What underlying measure led you to the questions, if any?
- Most importantly, iterative yes/no questions of this sort require no metric and are well-suited for nominal data.

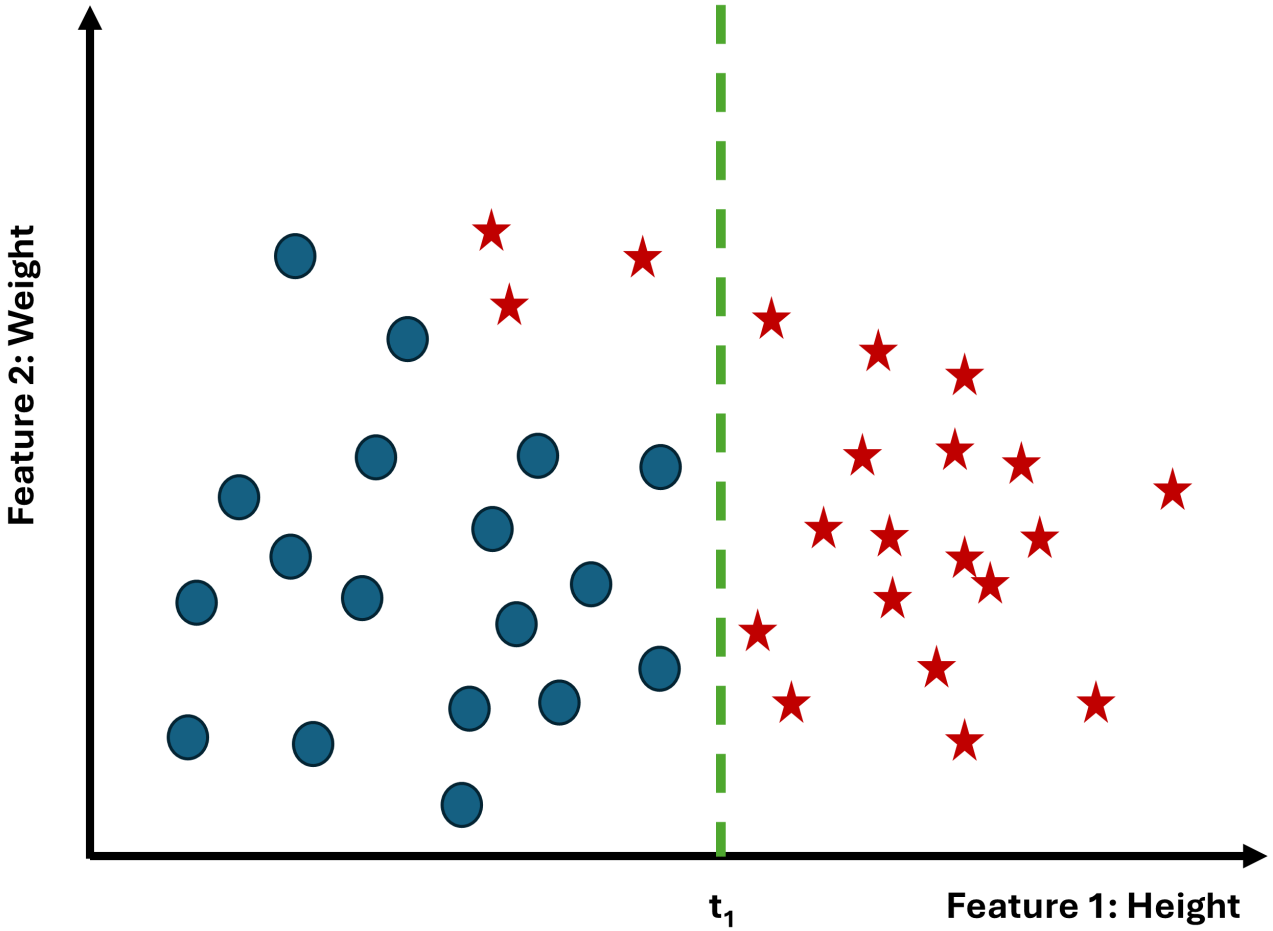
Decision Tree

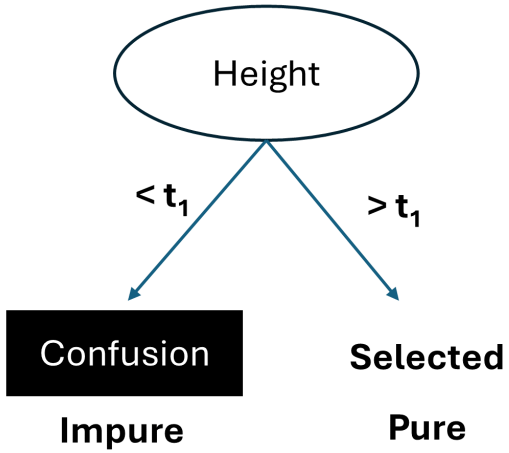


Decision Tree

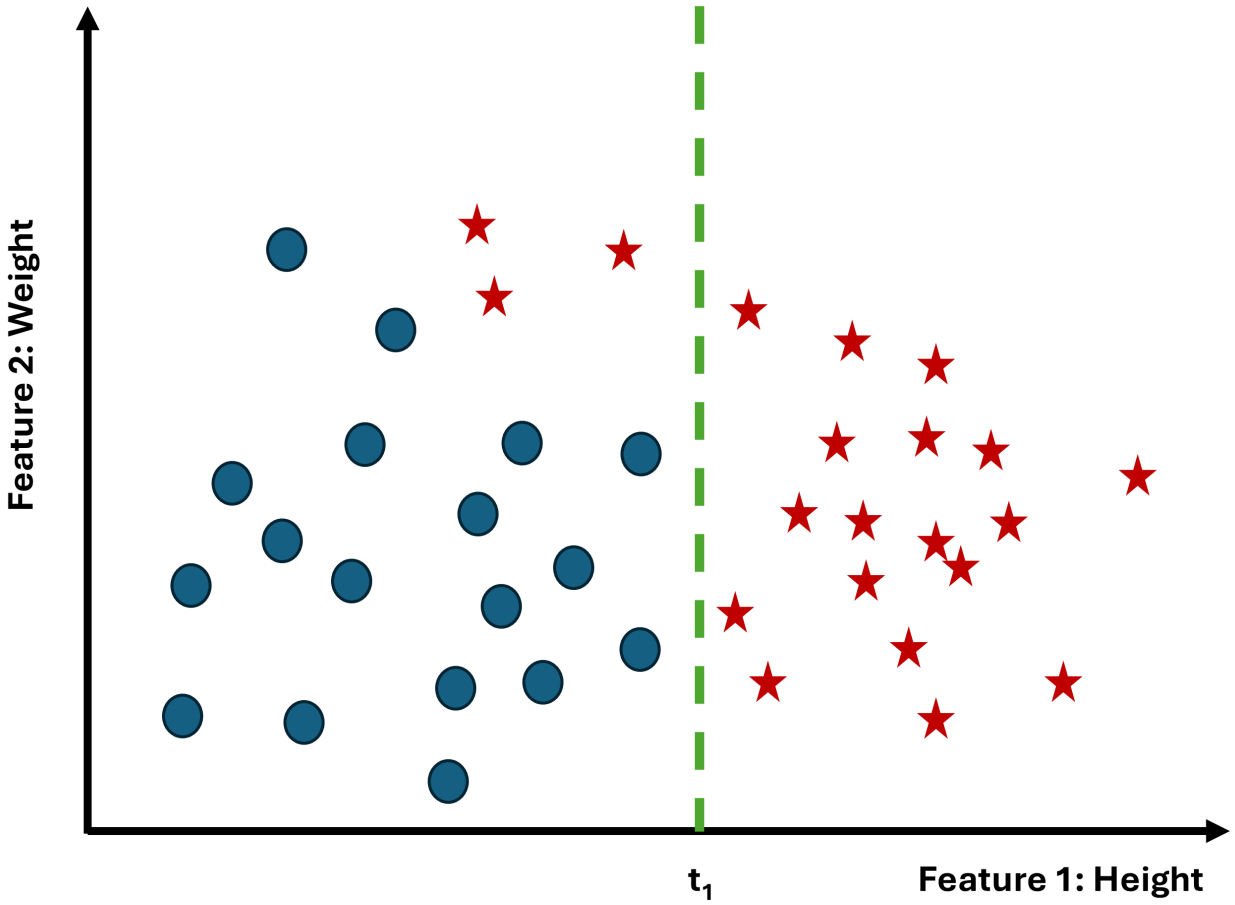


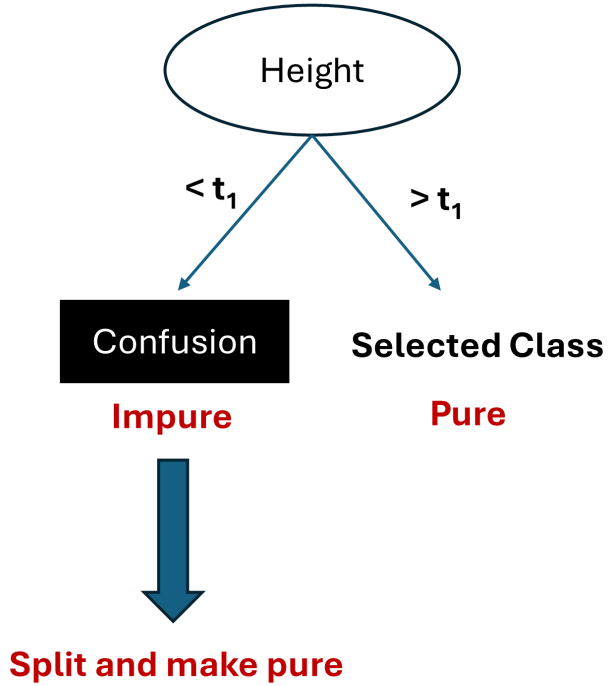
Decision Tree



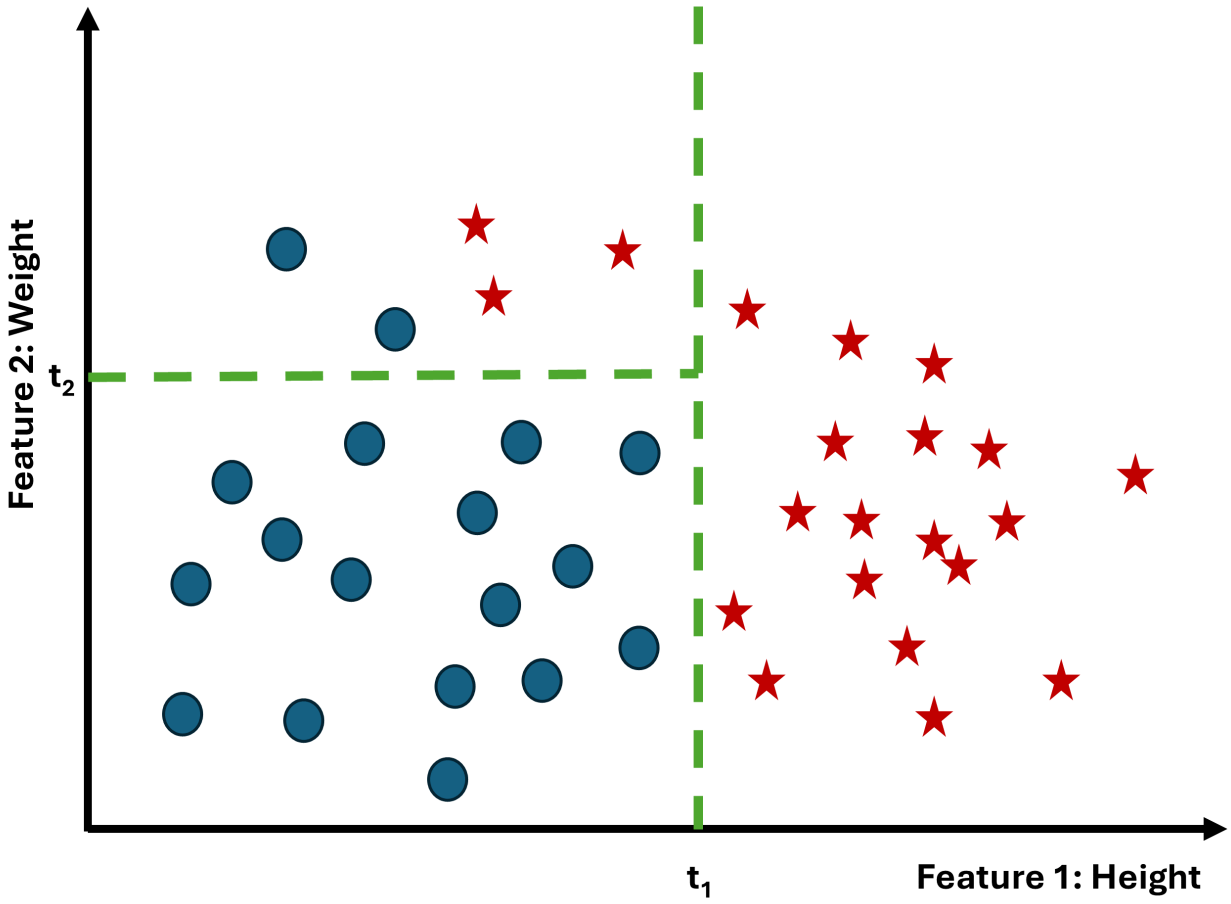


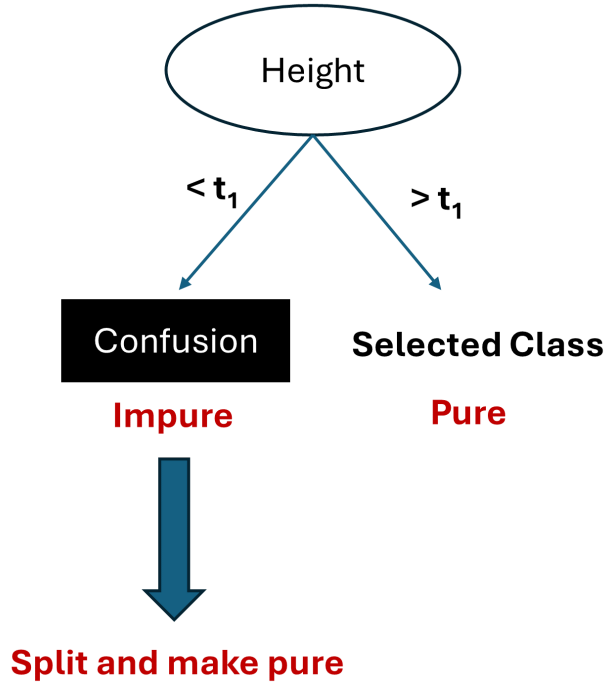
Decision Tree



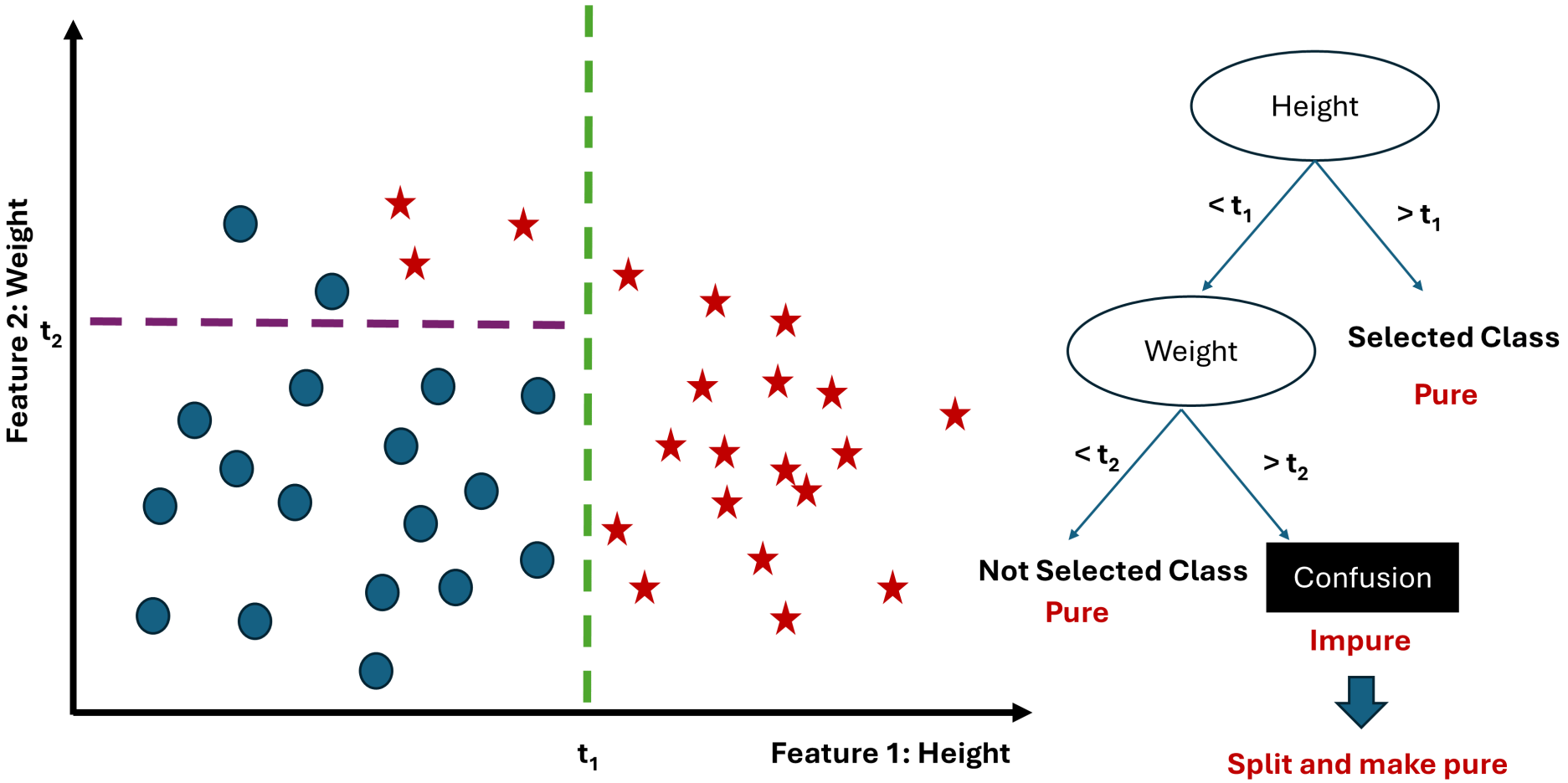


Decision Tree

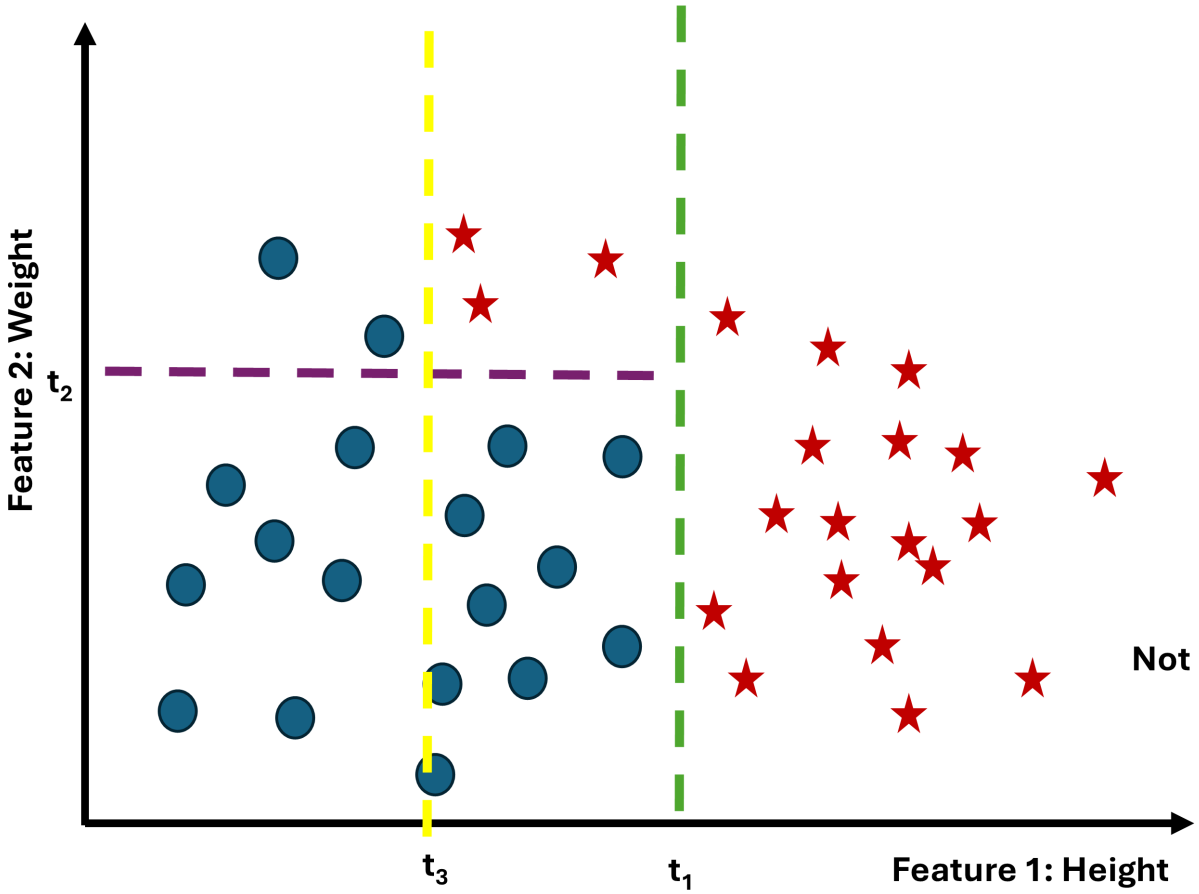


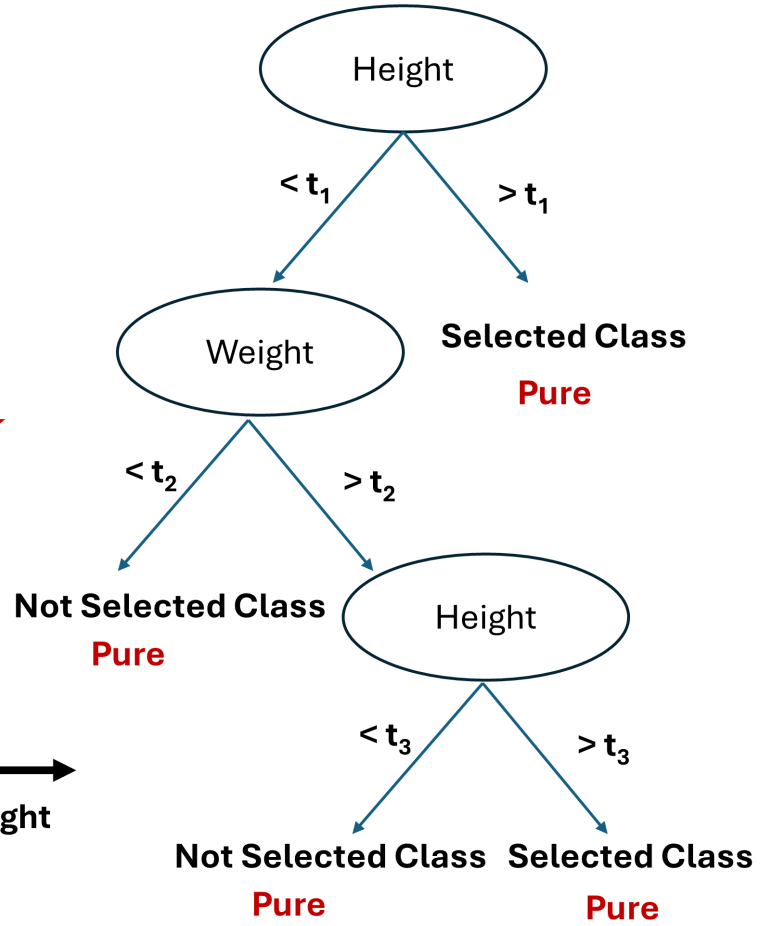


Decision Tree



Decision Tree





Decision Tree

- A decision tree starts with a root node at the very top. This is where the classification of a particular pattern begins by checking a specific property that was chosen during the tree's learning phase.
- The tree then branches out from the root node. Each branch corresponds to a different possible value of the property being checked.
- You follow the branch that matches the value of your pattern, which leads you to a new node where the process is repeated.
- This step-by-step process continues, with the tree checking one property after another, until you reach a leaf node.
- A leaf node signifies that a final decision has been reached and the pattern has been classified.
- This logical, easy-to-follow structure is what makes decision trees highly interpretable.

Decision Tree-CART

- Consider a labeled dataset D with a set of features chosen for classification.
- The goal is to arrange these features into a decision tree that achieves high classification accuracy.
- A decision tree works by recursively splitting the data into smaller subsets.
- When all samples in a subset belong to the same class, the node is said to be pure, and further splitting is unnecessary.
- In practice, complete purity is rare, so we must decide whether to stop with an imperfect split or continue growing the tree with additional features.
- CART adopt a greedy(i.e., non backtracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner

CART Strategy

- The basic CART strategy for recursively defining a tree is:
  - Given the data at a node, either declare it a leaf or choose a property to split the data into subsets.
- In this process, six key questions arise:
  1. How many branches should be created from a node?
  2. Which property should be tested at a node?
  3. When should a node be declared a leaf?
  4. How can we prune a tree that has grown too large?
  5. If a leaf node remains impure, how should its category be assigned?
  6. How should missing data be handled?

Number of Splits

- The number of splits at a node, called the branching factor B, is usually determined by the designer based on the test selection method.
- The branching factor can vary across different parts of the tree.
- Any split with more than two branches can be represented as a sequence of binary splits.
- For this reason, DHS primarily focuses on binary tree learning.
- However, in some cases, using 3- or 4-way splits may be preferred, as binary tests or inferences can be computationally expensive.

Principle of Tree Creation

- The core principle in building decision trees is simplicity: prefer small, compact trees with fewer nodes.
- At each node N, we select a property test T that makes the descendant nodes as pure as possible.
- Let $i(N)$ represent the impurity of node N:
  - $i(N) = 0$ if all samples in the node belong to one category (pure).
  - $i(N)$ is large when categories are equally represented (impure).
- A widely used impurity measure is **Entropy**:

$$i(N) = -\sum_j P(\omega_j) \log P(\omega_j)$$

- Entropy reaches its minimum when the node contains only one class.

Principle of Tree Creation

- For the two-class case, **Variance Impurity** is defined as:

$$i(N) = P(\omega_1)P(\omega_2)$$

- For multi-class problems, the **Gini Impurity** is commonly used:

$$i(N) = \sum_{i \neq j} P(\omega_i)P(\omega_j) = 1 - \sum_{j} P^2(\omega_j)$$

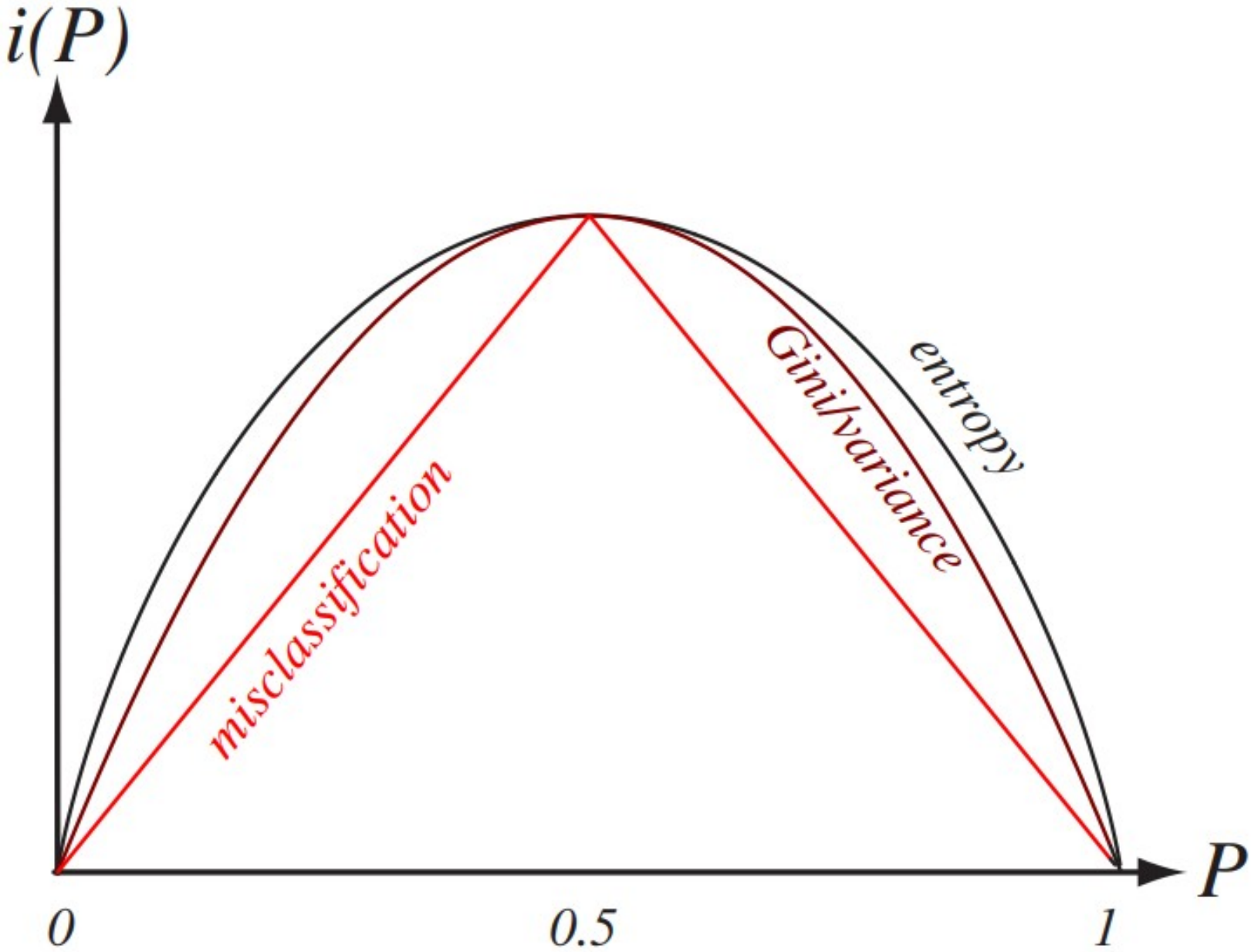
  This represents the expected error rate if the class label is chosen randomly according to the class distribution at node N.

- Another measure is the **Misclassification Impurity**: Valid for two category only

$$i(N) = 1 - \max_j P(\omega_j)$$

  This gives the minimum probability that a training sample at node N will be misclassified.

Principle of Tree Creation



Feature Selection at a Node

- **Key Question:** Given a partial tree down to node N, which feature should be selected for the property test T?
- Heuristic: choose the feature that produces the largest decrease in impurity.
- The impurity gradient is defined as:

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$

where:

  - N_L, N_R: left and right child nodes.
  - P_L: fraction of samples directed to the left subtree by test T.

- Strategy: select the feature that maximizes $\Delta i(N)$.
- If entropy impurity is used, this is equivalent to selecting the feature with the highest **Information Gain**.

Binary and Multi-Class Splits

- In the binary case, feature selection reduces to a one-dimensional optimization problem (which may have multiple optima).
- For higher branching factors, the problem becomes higher-dimensional.
- In multi-class binary tree construction, the **twoing criterion** is often used:
  - Goal: find a split that best separates the c categories into two groups.
  - Define a candidate "supercategory" C_1 (subset of categories) and C_2 (the remainder).
  - Search must consider both features and category groupings.
- This approach follows a local, greedy optimization strategy:
  - No guarantee of achieving the global optimum in accuracy.
  - No guarantee of obtaining the smallest possible tree.
- In practice, the specific choice of impurity function has little effect on the final classifier's accuracy.

When to Stop Splitting?

- If the tree grows until each leaf has only one sample (minimum impurity), it will be **overfitted** and fail to generalize.
- If the tree is stopped too early, training error remains high and performance suffers.
- Common strategies to decide when to stop splitting:
  1. Use **cross-validation** to determine optimal stopping.
  2. Set a **threshold** on the impurity gradient.
  3. Add a **tree-complexity penalty term** and minimize.
  4. Apply a **statistical test** on the impurity gradient.

Stopping Criterion Using Threshold β

- Splitting is stopped if the best candidate split at a node reduces impurity by less than a preset threshold β:

$$\max_s \Delta i(s) \leq \beta$$

- **Benefits:**
  - Unlike cross-validation, the tree is trained on the full dataset.
  - Leaf nodes can occur at different depths, which adapts to varying data complexity.
- **Drawback:**
  - Choosing an appropriate value for β is non-trivial.

Stopping with a Complexity Term

- Define a global criterion function that balances complexity and accuracy:

$$\alpha \cdot \text{size} + \sum_{\text{leaf nodes}} i(N)$$

  where:
  - **size**: number of nodes or links in the tree.
  - α: positive constant controlling the trade-off.
- Splitting continues until this global criterion is minimized.
- With entropy impurity, this measure is related to the **Minimum Description Length (MDL) principle**.
- The sum of leaf node impurities represents the uncertainty of the training data given the tree model.
- **Drawback:** The challenge is how to appropriately set the constant α.

Statistical Testing for Stopping Splits

- During tree construction, estimate the distribution of impurity gradients Δi across the current nodes.
- For any candidate split, test if Δi is significantly different from zero.
- Possible approaches:
  - Use a **Chi-squared test**.
  - More generally, apply a **hypothesis testing** framework to check whether a split is better than a random split.
- Example: Suppose there are n samples at node N.
  - A candidate split s sends Pn samples to the left and $(1 - P)n$ samples to the right.
  - Under a random split:
    - Pn_1 of ω_1 samples go left.
    - Pn_2 of ω_2 samples go left.
    - Remaining samples go to the right branch.

Chi-Squared Test for Splitting

- The **Chi-squared statistic** measures how much a candidate split s deviates from a random split:

$$\chi^2 = \sum_{i=1}^{2} \frac{(n_{iL} - n_{ie})^2}{n_{ie}}$$

where:

  - n_{iL} = number of ω_i patterns sent to the left under split s.
  - $n_{ie} = Pn_i$ = expected number sent left under a random split.

- Interpretation:
  - Larger $\chi^2 \rightarrow$ greater deviation from random splitting.
  - If χ^2 exceeds a critical value (based on significance level), reject the null hypothesis of randomness.
  - In this case, accept split s as meaningful.

Pruning in Decision Trees

- Stopping-split criteria bias trees toward large impurity reductions near the root.
- These methods do not account for possible future splits deeper in the tree.
- **Pruning** is an alternative strategy:
  - First, grow the tree fully (exhaustive construction).
  - Then, consider all pairs of neighboring leaf nodes for elimination.
  - If eliminating the pair only slightly increases impurity, replace them with their common ancestor node as a leaf.
- Characteristics of pruning:
  - Often produces unbalanced trees.
  - Avoids the local nature of early stopping.
  - Uses the full training dataset.
  - Involves higher computational cost.

Points to Note

- As we know from Ugly Duckling and various empirical evidence, the selection of features will ultimately play a major role in accuracy, generalization, and complexity
- Furthermore, the use of multiple variables in selecting a decision rule may greatly improve the accuracy and generalization.

Problem Link + Additional Learning:
YouTube Video Link