Support Vector Machine and Random Forest

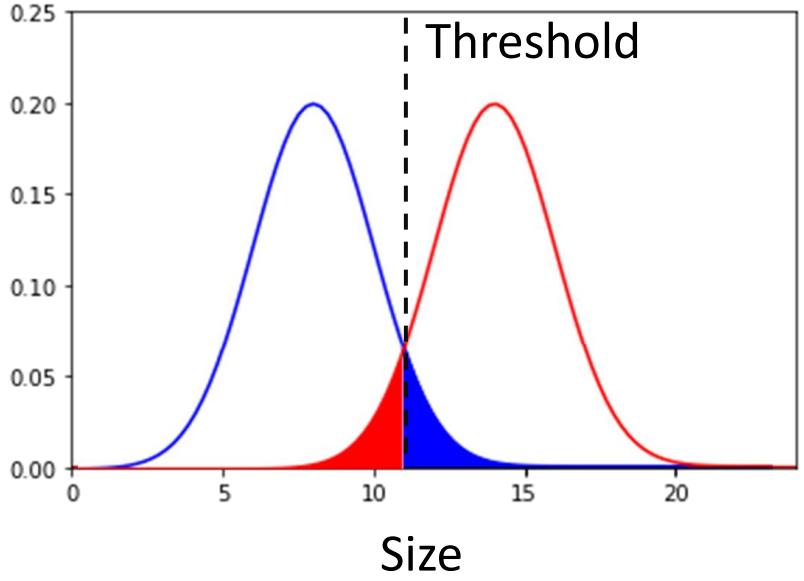
Machine Learning Summer Course 2020

Krishnakant Saboo

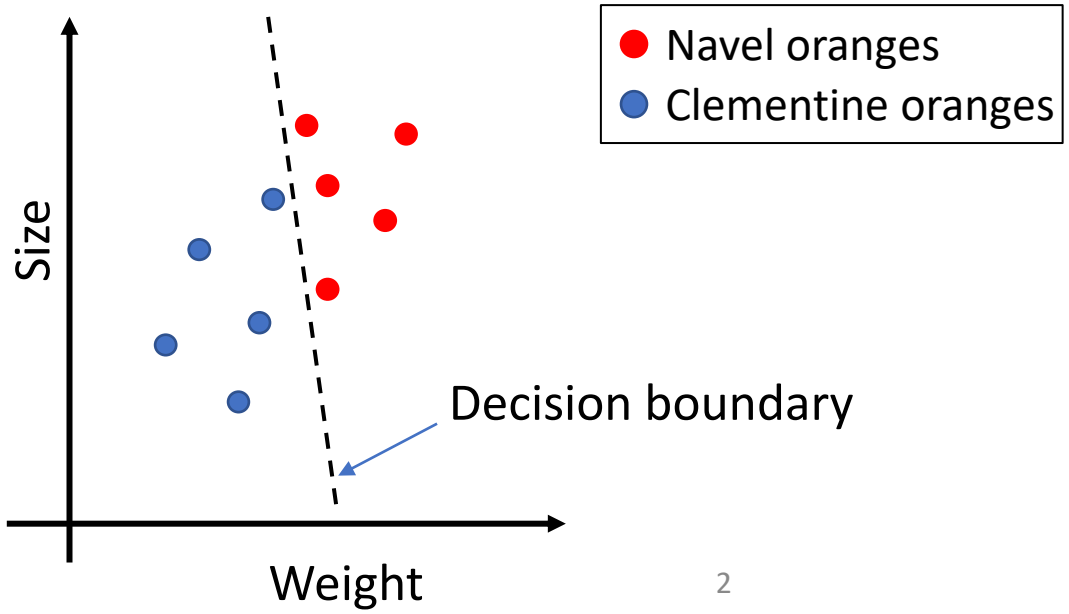
11th July 2020

Decision boundary

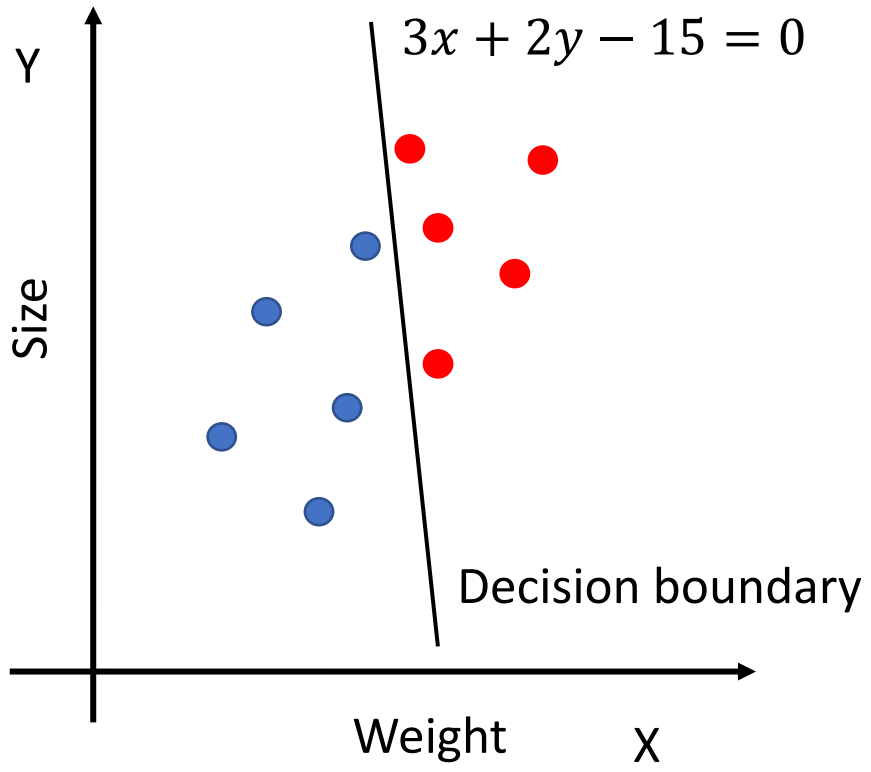
- kNN is computationally expensive because distance from N points needs to be computed for every new sample
- Threshold rule is computationally inexpensive
  - The example was for a single dimension and the distribution was known
- Need a “threshold” for multi-dimensional data: ***decision boundary***

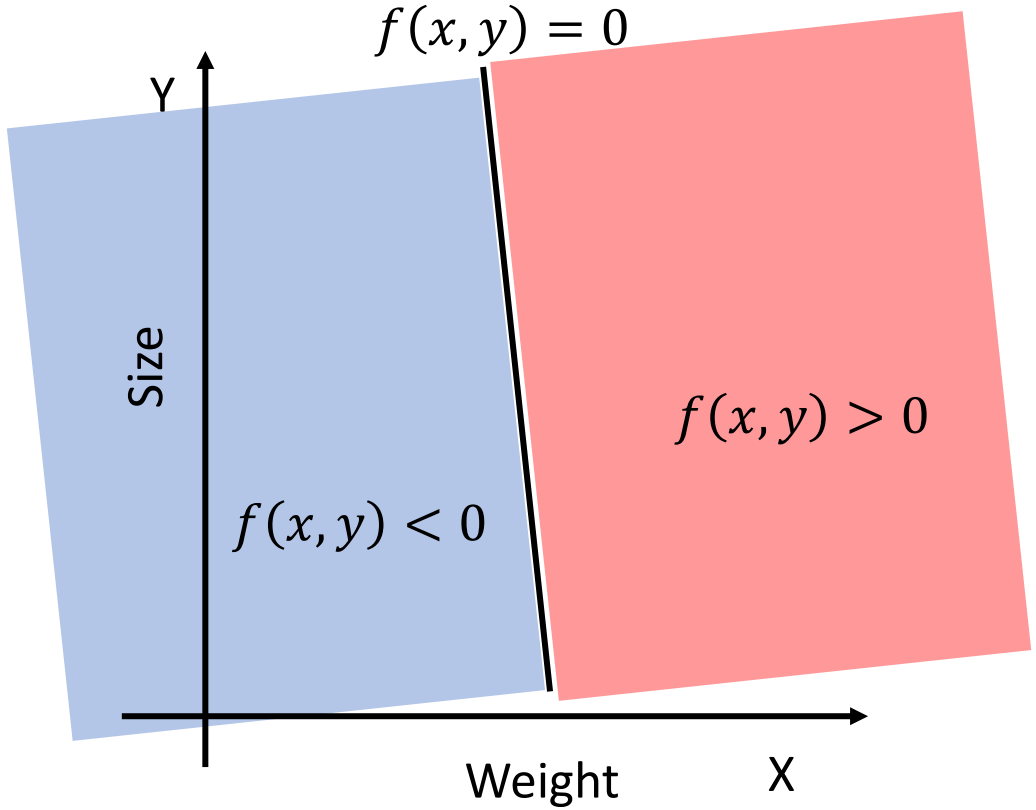


Example (Lec 2). Type of new orange is predicted by comparing its size with the threshold.



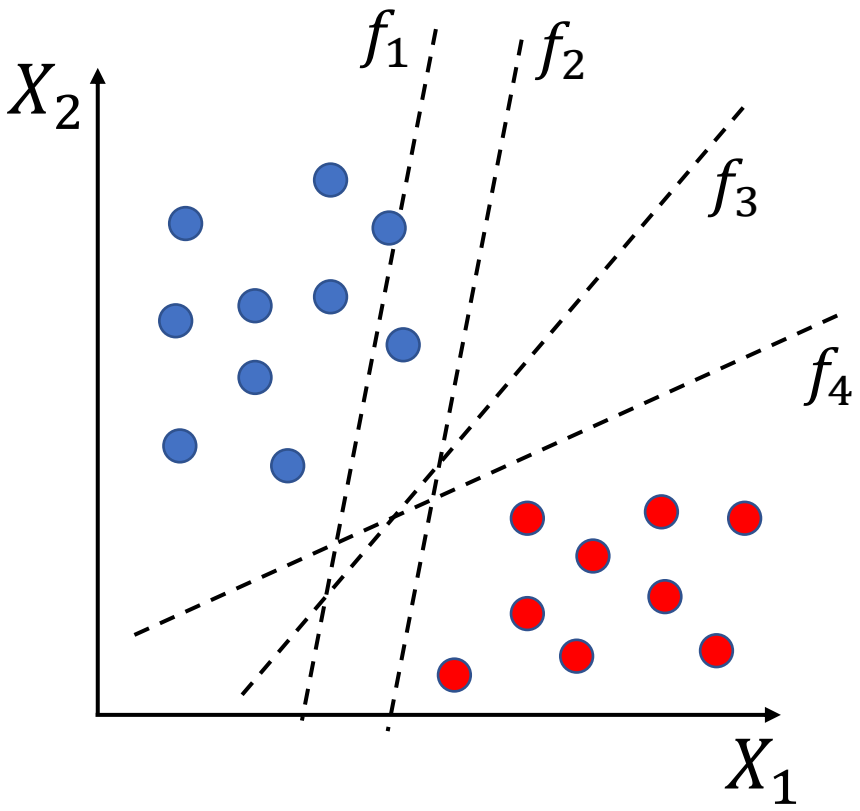
Classification using a decision boundary

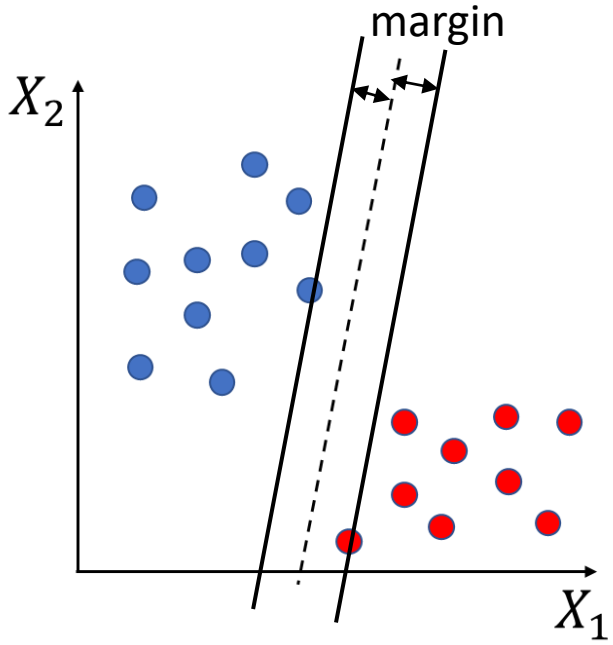




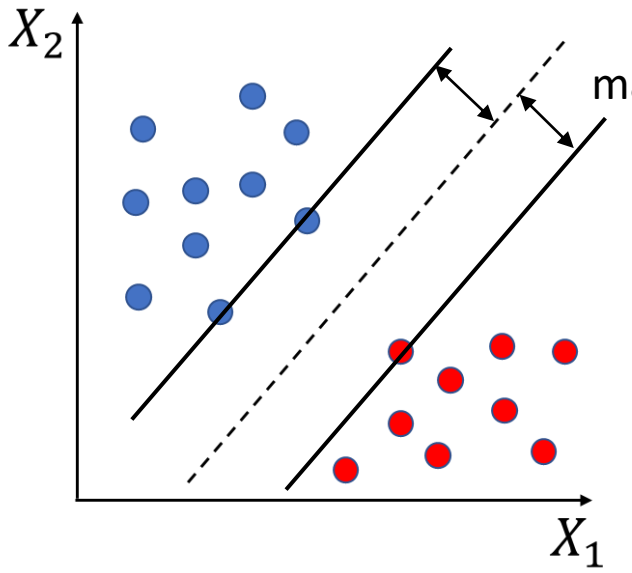
- New sample: (x_{new}, y_{new}) and decision boundary $f(x, y) = 0$
- If $f(x_{new}, y_{new}) < 0$ then class 0 (blue)
- If $f(x_{new}, y_{new}) > 0$ then class 1 (red)

Which boundary to use?





Small margins



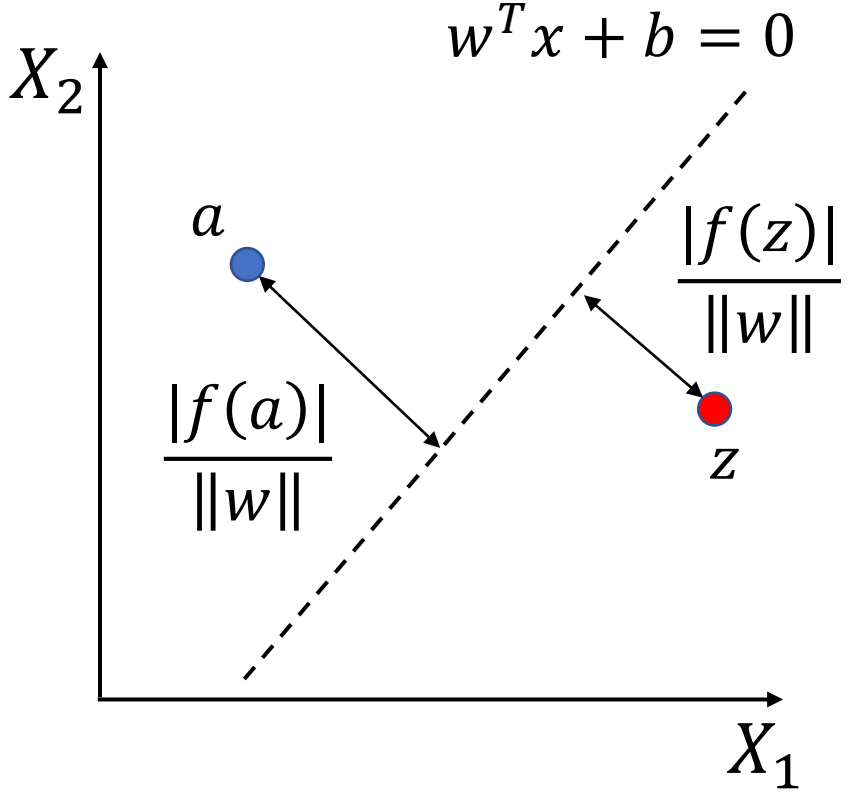
Large margins

- Criterion 1: Should classify all the samples correctly
- Criterion 2: Margin should be large to reduce generalization error

Preliminaries

- Vectors $w = [w_1, w_2], x = [x_1, x_2]$
- $f(x) = w_1 x_1 + w_2 x_2 + b = w^T x + b$
- Equation of line: $f(x) = 0$
- Norm of vector w: $\|w\| = \sqrt{w_1^2 + w_2^2}$
- Distance of point $a = [a_1, a_2]$ from the line:

$$\frac{|f(a)|}{\|w\|}$$



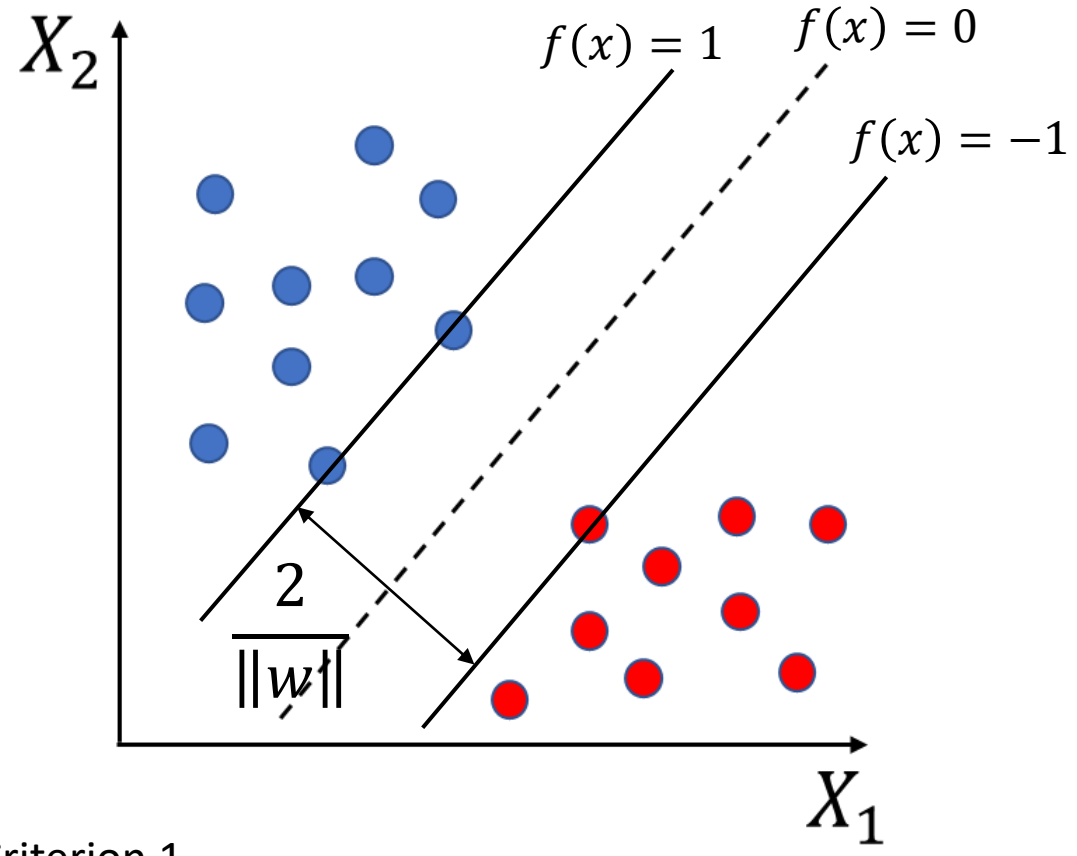
- Equations hold for hyperplanes (dimensions $d \geq 3$)

Support Vector Machine

- Training data
  $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$
  - $y_i \in \{-1, 1\}$
  - $x_i = [x_{i1}, x_{i2}, \dots, x_{id}]$
- Decision boundary:
  $$f(x) = w^T x + b = 0$$
- Optimization problem:

$$\max_{w,b} \frac{2}{\|w\|} \quad \leftarrow \text{Criterion 2}$$

$$s.t. \quad y_i f(x_i) \geq 1 \ \forall i \quad \leftarrow \text{Criterion 1}$$



SVM optimization problem

Note the square

$$\max_{w,b} \frac{2}{\|w\|} \quad \Rightarrow \quad \min_{w,b} \frac{\|w\|^2}{2}$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1 \ \ \forall\, i \qquad\qquad s.t. \quad y_i(w^T x_i + b) \geq 1 \ \ \forall\, i$$

$$\Rightarrow \quad \max_{\lambda_1,\ldots,\lambda_N} \min_{w,b} \mathcal{L}(w, b, \lambda_1, \ldots, \lambda_N)$$

where

$$\mathcal{L}(w, b, \lambda_1, \ldots, \lambda_N) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \lambda_i \{y_i(w^T x_i + b) - 1\}$$

and

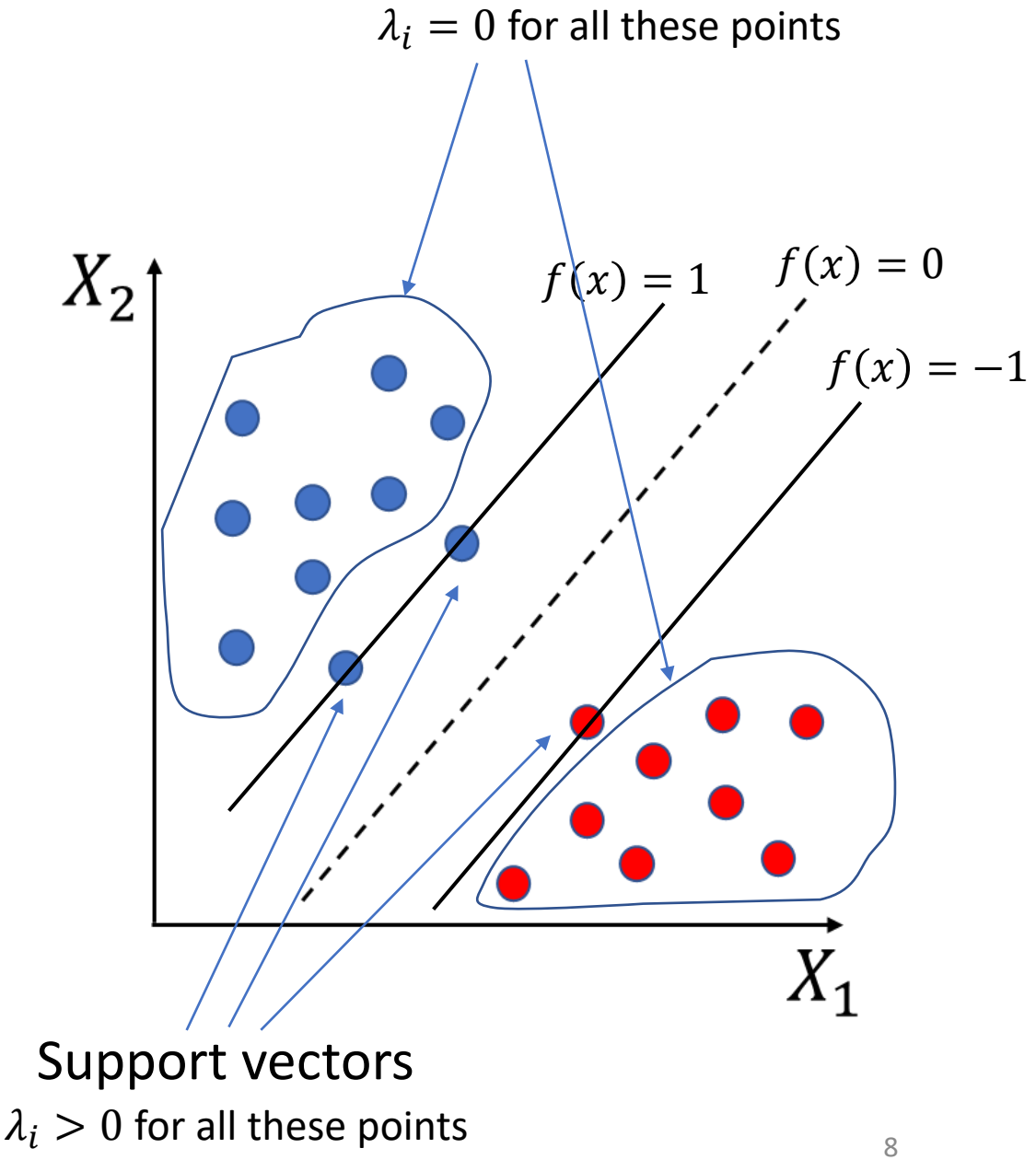
$$\lambda_i \geq 0 \quad \forall i$$

SVM solution

- Use standard tools to optimize
- Solution:

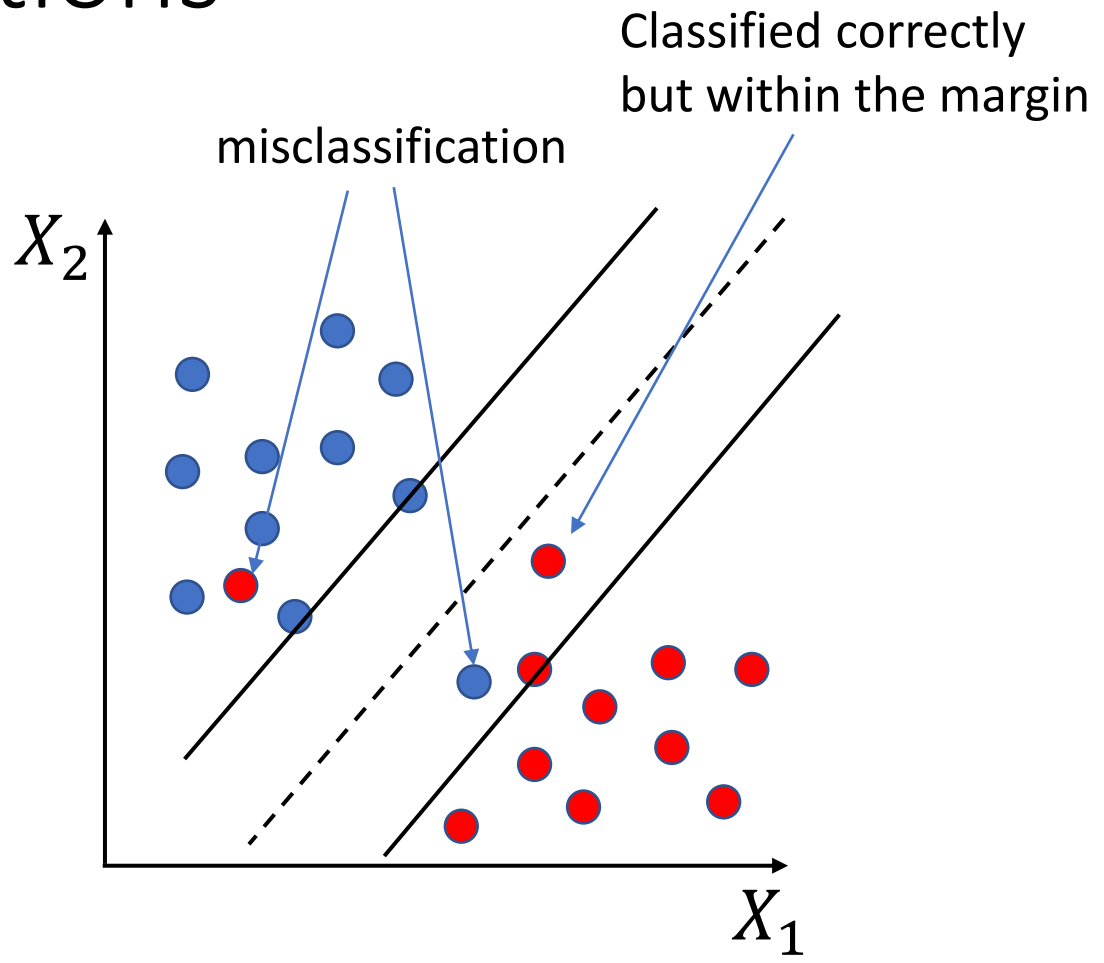
$$w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

- $\lambda_i = 0$ for all points except the ones on the margins



Allowing some misclassifications

- Real-world data is not always completely separable
- Modify SVM to allow for
  - Some misclassifications
  - Some points to be correctly classified but lie within the margin
- Optimization problem with almost the same with an extra parameter $C > 0$
  - Controls how many points are within the margin or misclassified
  - $C \to 0$ allows more misclassifications during training
  - $C \to \infty$ makes the model more complex



SVM algorithm

- Training data: $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$
- Set parameter $C > 0$
- Training: Solution (best w, b) depends only on the support vectors

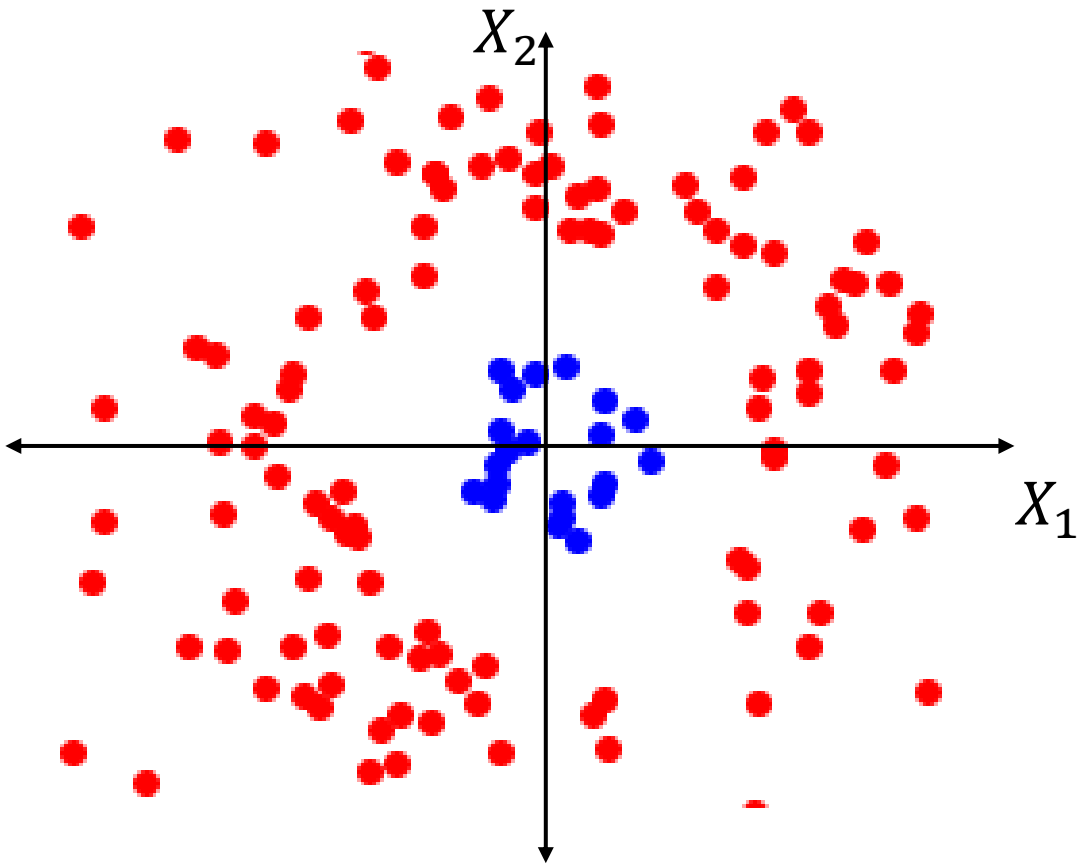
$$w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

- For a new sample x_{new}, decision is made as follows

$$f(x_{new}) = w^T x_{new} + b = \sum_{i=1}^{N} \lambda_i y_i x_i^T x_{new} + b$$

$$\hat{y}_{new} = \begin{cases} -1, & f(x_{new}) < 0 \\ 1, & f(x_{new}) > 0 \end{cases}$$

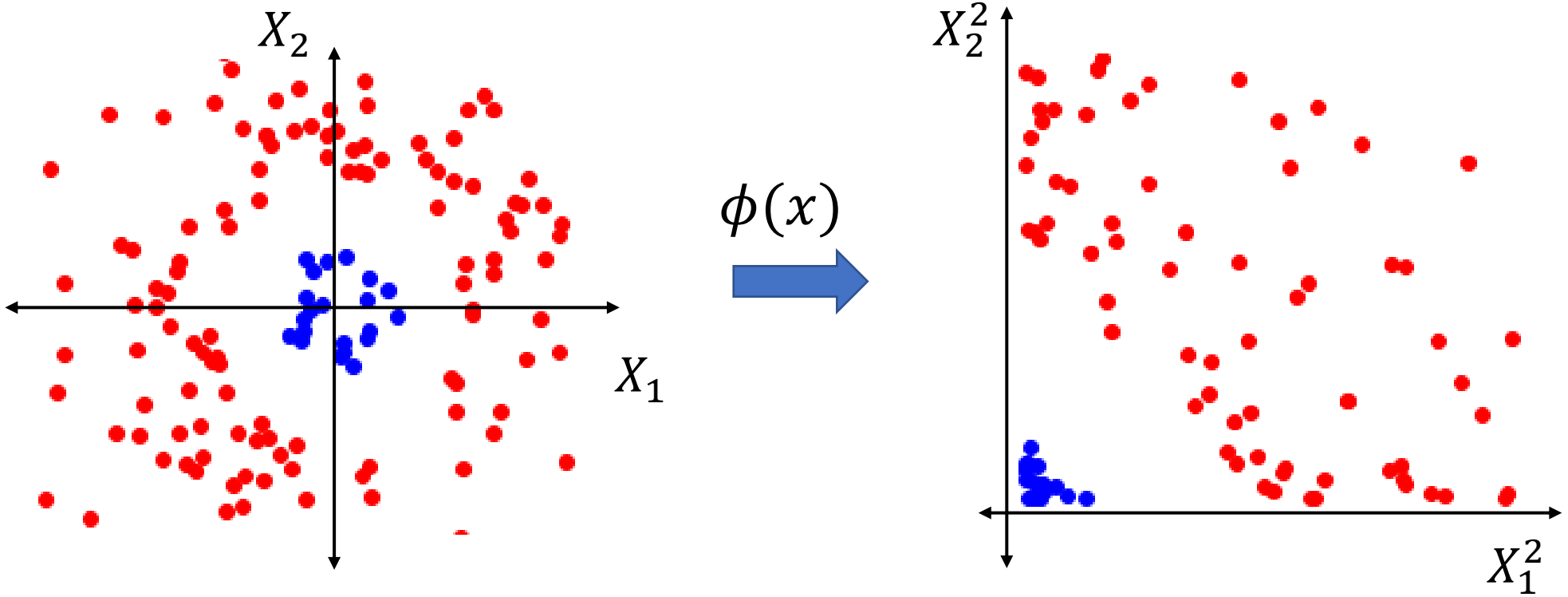
Non-linear decision boundary



Will linear SVM be able to differentiate between the two classes?

Transformation

- Points $x = [x_1, x_2]$
- ***Transform*** the point $x \rightarrow \phi(x)$ where $\phi(x) = [x_1^2, x_2^2]$



- SVM can classify the transformed data

Kernel SVM

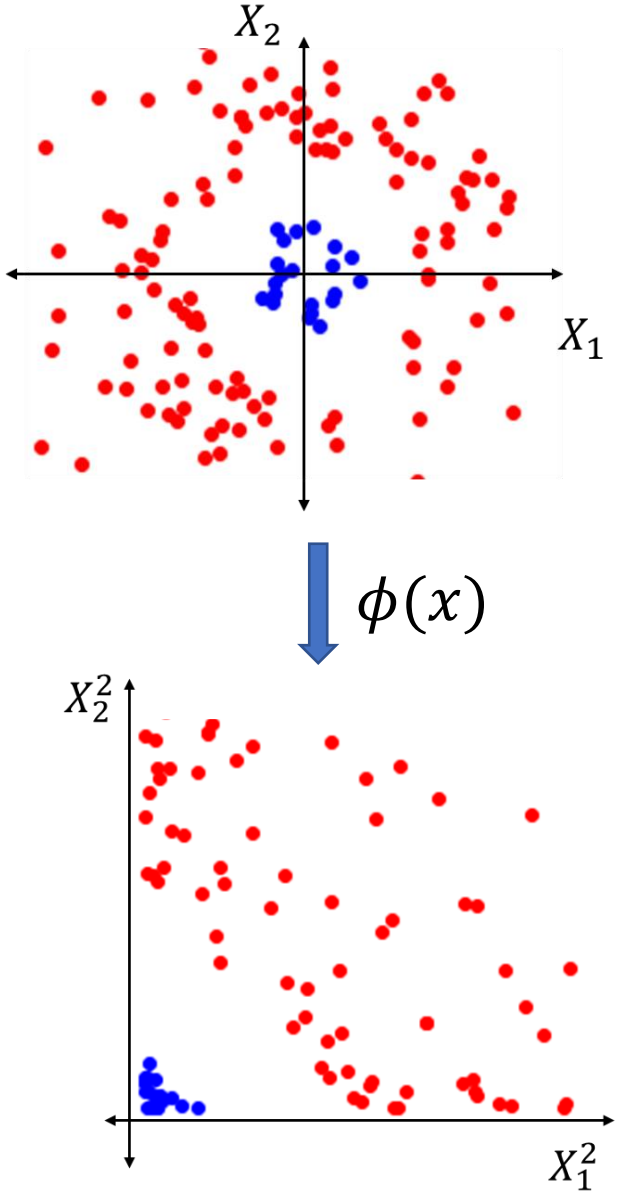
When transformation $\phi(x)$ is applied to the data, SVM parameters w becomes

$$w = \sum_{i=1}^{N} \lambda_i y_i \phi(x_i)$$

$$f(x_{new}) = \sum_{i=1}^{N} \lambda_i y_i \phi(x_i)^T \phi(x_{new}) + b$$

$$= \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{k}(x_i, x_{new}) + b$$

kernel



Kernel SVM

$$f(x_{new}) = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{k}(x_i, x_{new}) + b$$

- Non-linear decision boundaries
- Decision depends only on the
  - Support vectors
  - Kernel $\boldsymbol{k}(x_i, x_{new})$
- Commonly used kernels $\boldsymbol{k}(x_i, x_j)$
  - Polynomial: $\left(1 + x_i^T x_j\right)^m$
  - Radial basis function: $\exp(-\gamma \left\| x_i - x_j \right\|^2)$

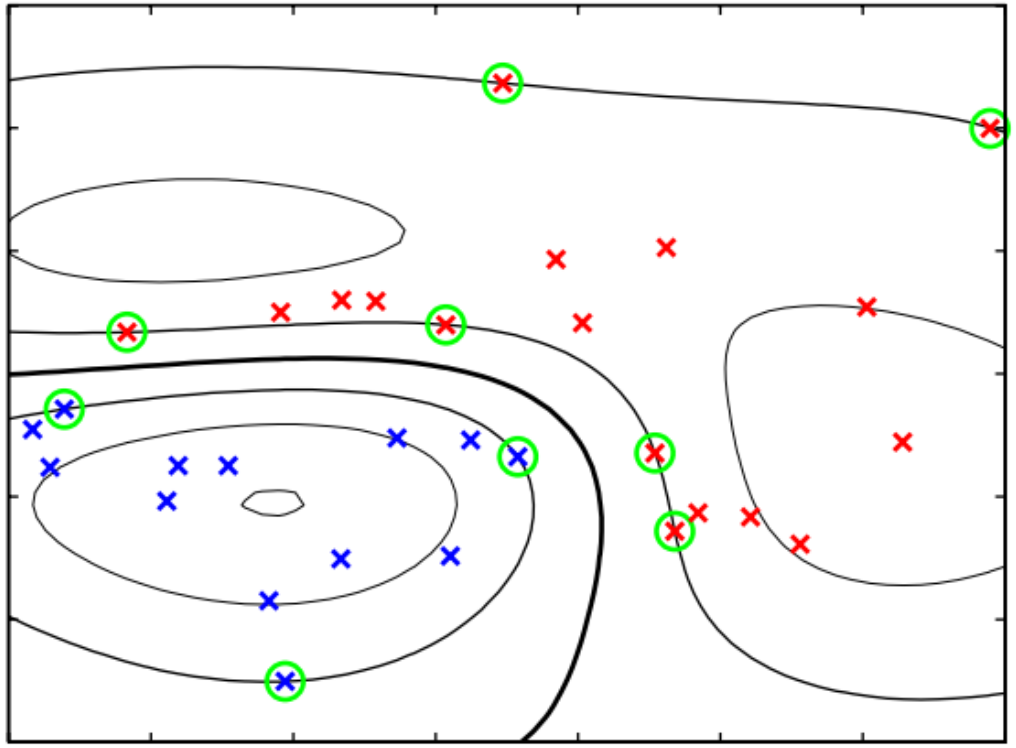


Figure 7.2 Example of synthetic data from two classes in two dimensions showing contours of constant $y(\mathbf{x})$ obtained from a support vector machine having a Gaussian kernel function. Also shown are the decision boundary, the margin boundaries, and the support vectors.

Image source: Bishop, (2006) “Pattern Recognition and Machine Learning”.

Kernel SVM algorithm

- Training data: $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$
- Set regularization parameter $C > 0$
- Choose kernel and its parameter (for e.g., m for polynomial)
- Train the model
- For a new sample x_{new}, decision is made as follows

$$f(x_{new}) = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{k}(x_i, x_{new}) + b$$

$$\hat{y}_{new} = \begin{cases} -1, & f(x_{new}) < 0 \\ 1, & f(x_{new}) > 0 \end{cases}$$

- Decision only depends on the support vectors and the kernel
  - Even ϕ is not required

Effect of hyperparameters γ on RBF kernel and C

- RBF kernel:

$$\exp(-\gamma \|x_i - x_j\|^2)$$

- As γ increases, curvature increases
  - For very large γ, radius of influence of SV only includes SV
- Small C allows more misclassifications
  - As C increases, more and more misclassified points lie more in the margins

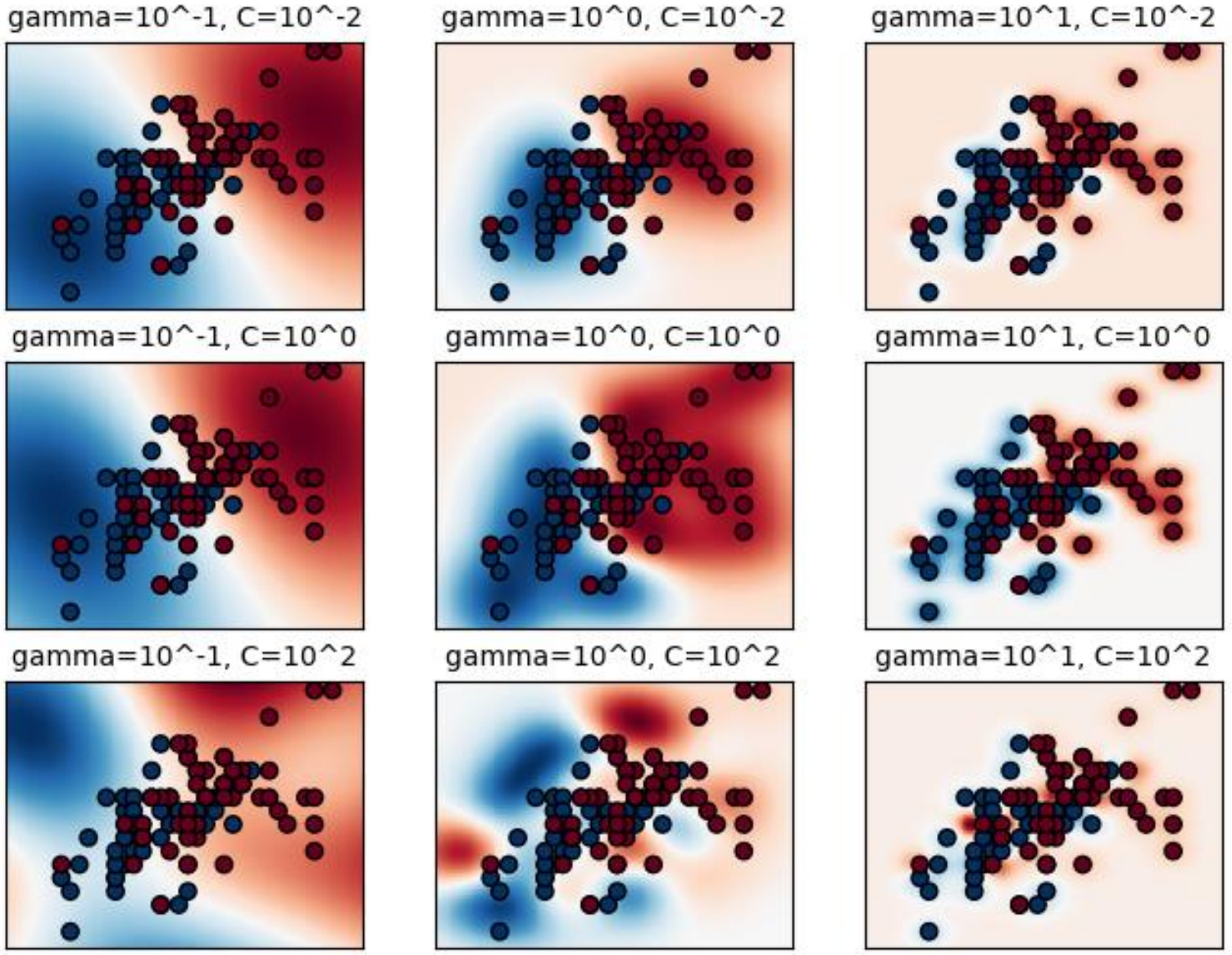


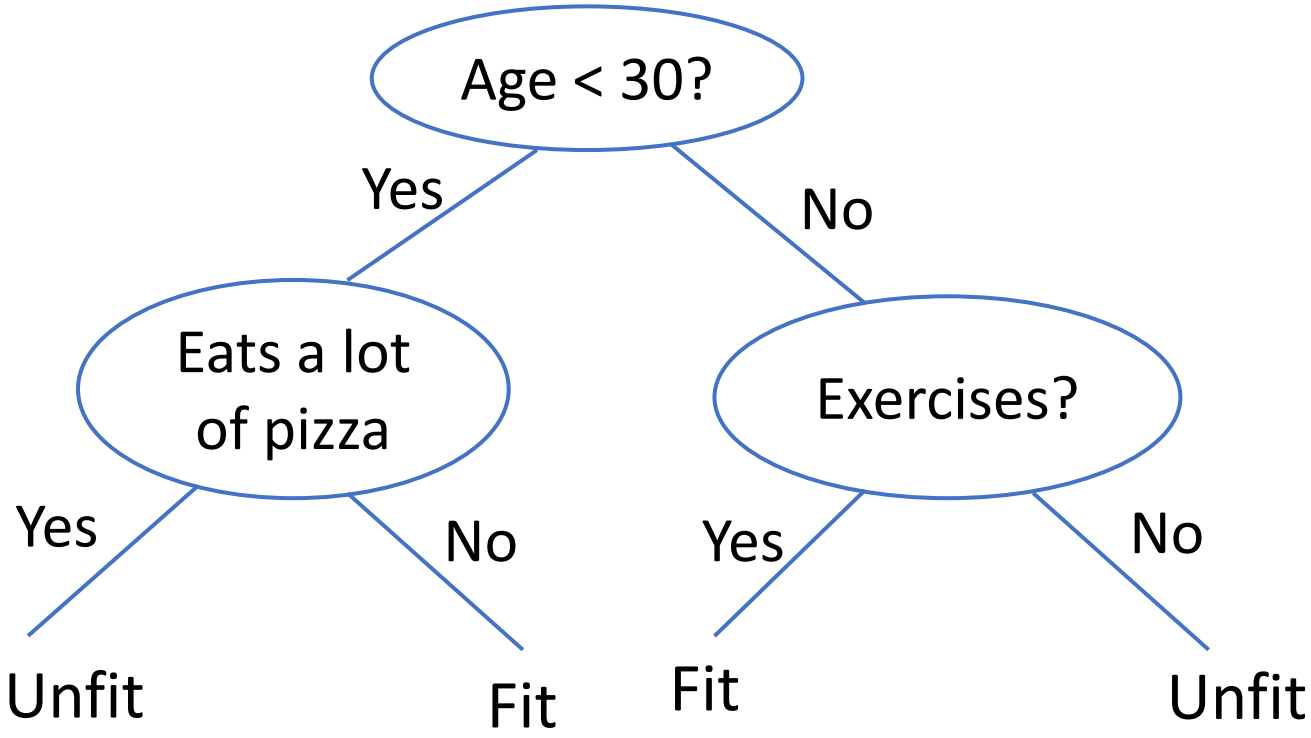
Image source: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

Decision trees and Random forests

Decision tree examples

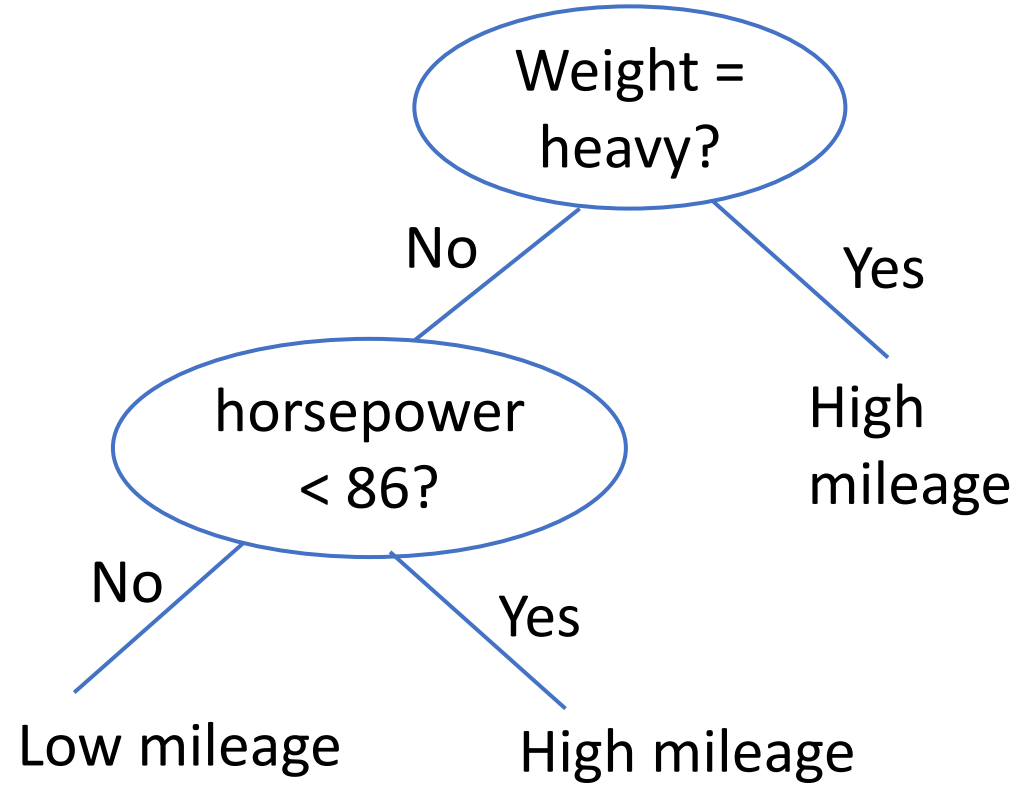
Is a person fit?

Data: age, pizza, exercise



Does a car have a high mileage?

Data: weight, horsepower

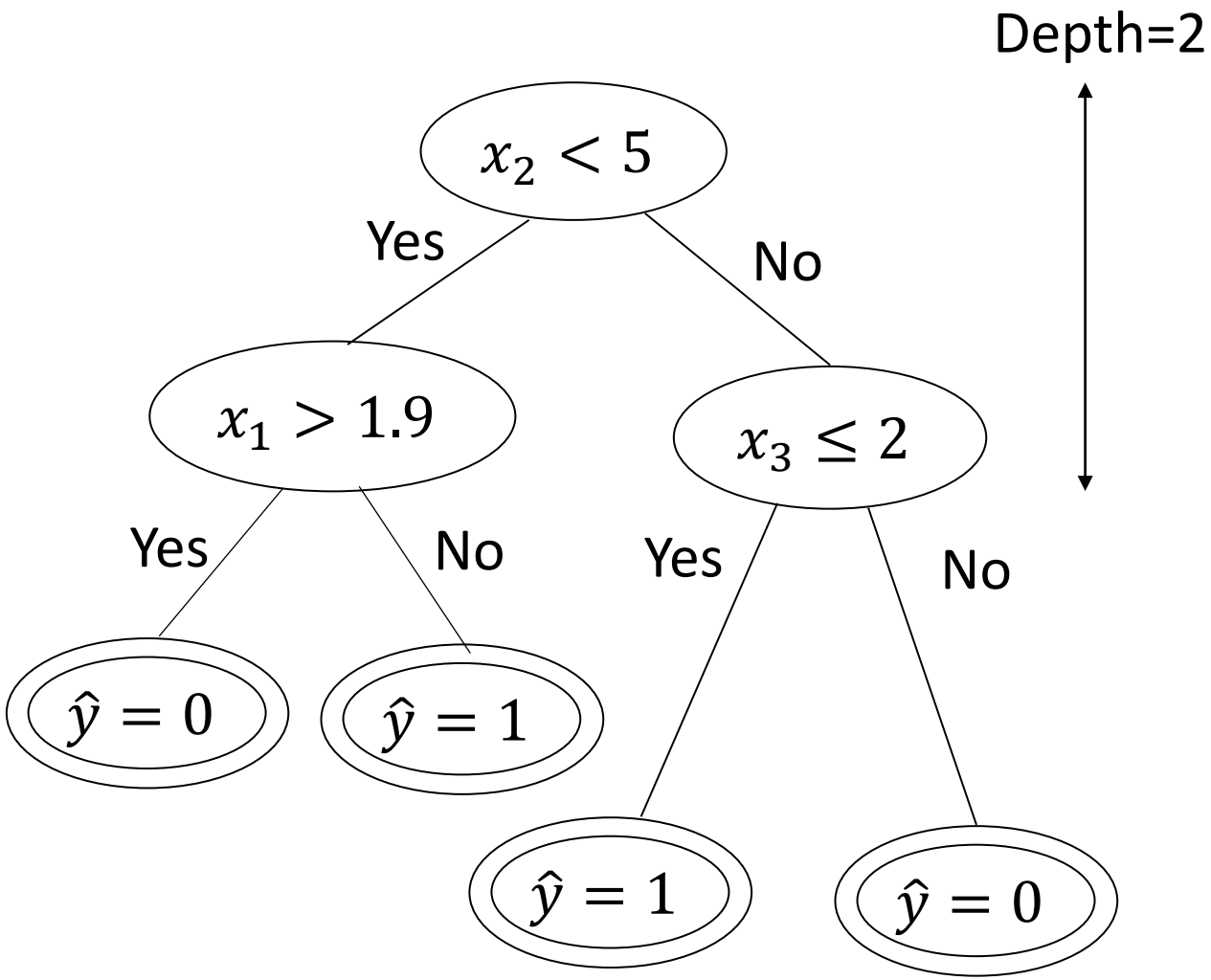


Make decision based on sequential questions that consider one feature at a time

Example source: https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb

Decision tree

- A decision tree maps input $x \in R^d$ to output y using binary decision rules:
  - Each node is the tree has a **splitting rule**
  - Each **leaf node** is associated with an output value (outputs can repeat)
- Each splitting rule compare a feature to a threshold
- Using these splitting rules a path to a leaf node gives the prediction



Example decision tree for binary class classification with 3 dimensional features

Decision tree construction algorithm

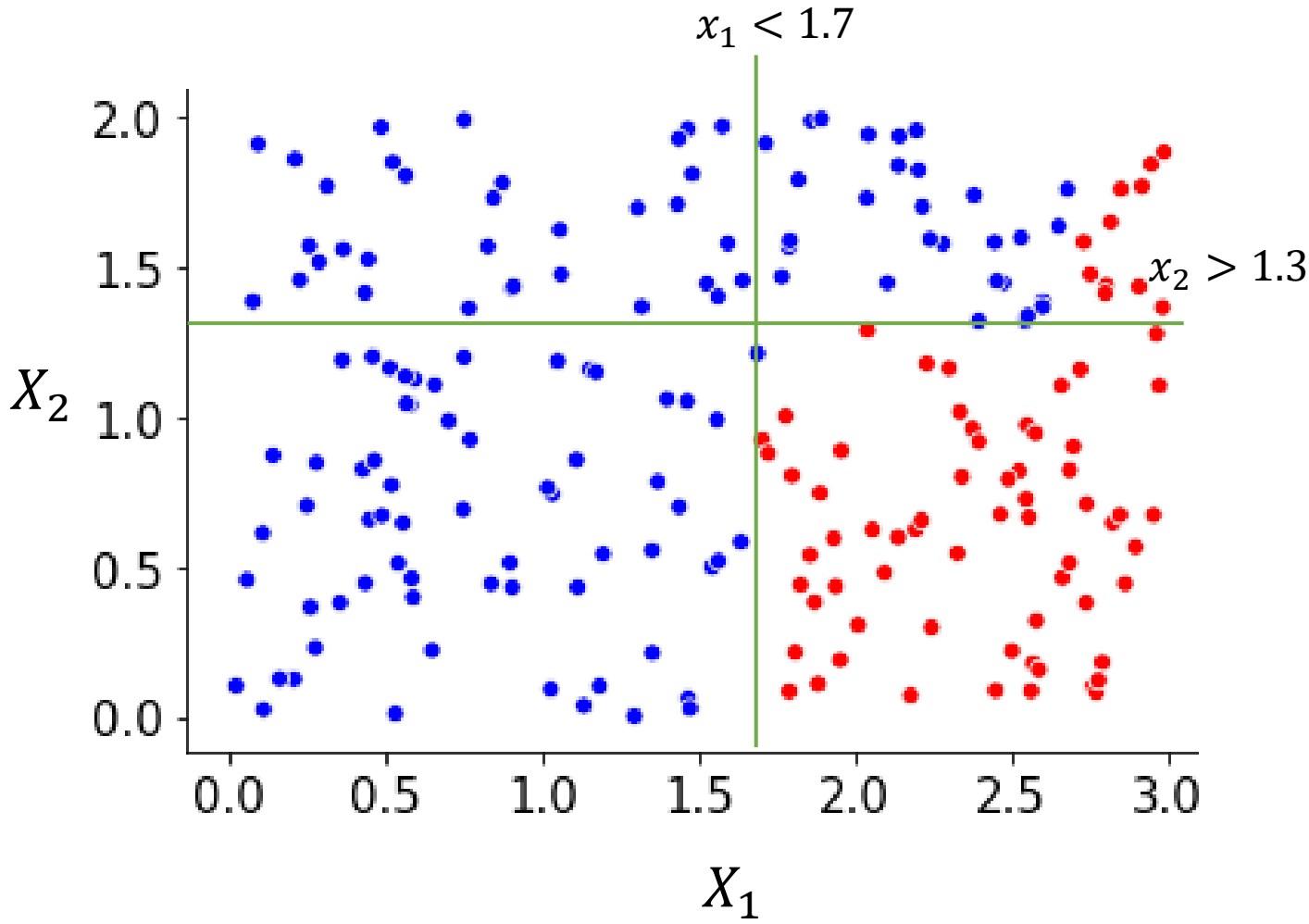
1. Start with a single leaf node containing all data
2. Loop through the following steps:
   - Pick the leaf to split that reduces uncertainty the most
   - Pick the splitting rule for the chosen leaf node using the following
     i. For each feature, check all possible splits and find the best split
     ii. With the best split of each feature determined, pick the best feature
3. Continue splitting nodes (increasing depth of tree) until stopping criteria is reached

Label/response of the leaf is majority of the data assigned to it

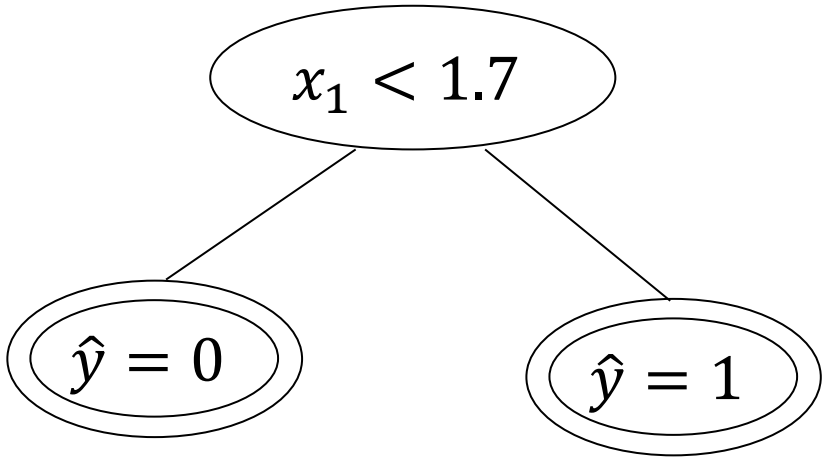
Decision tree: Solved example

Loop through the following:

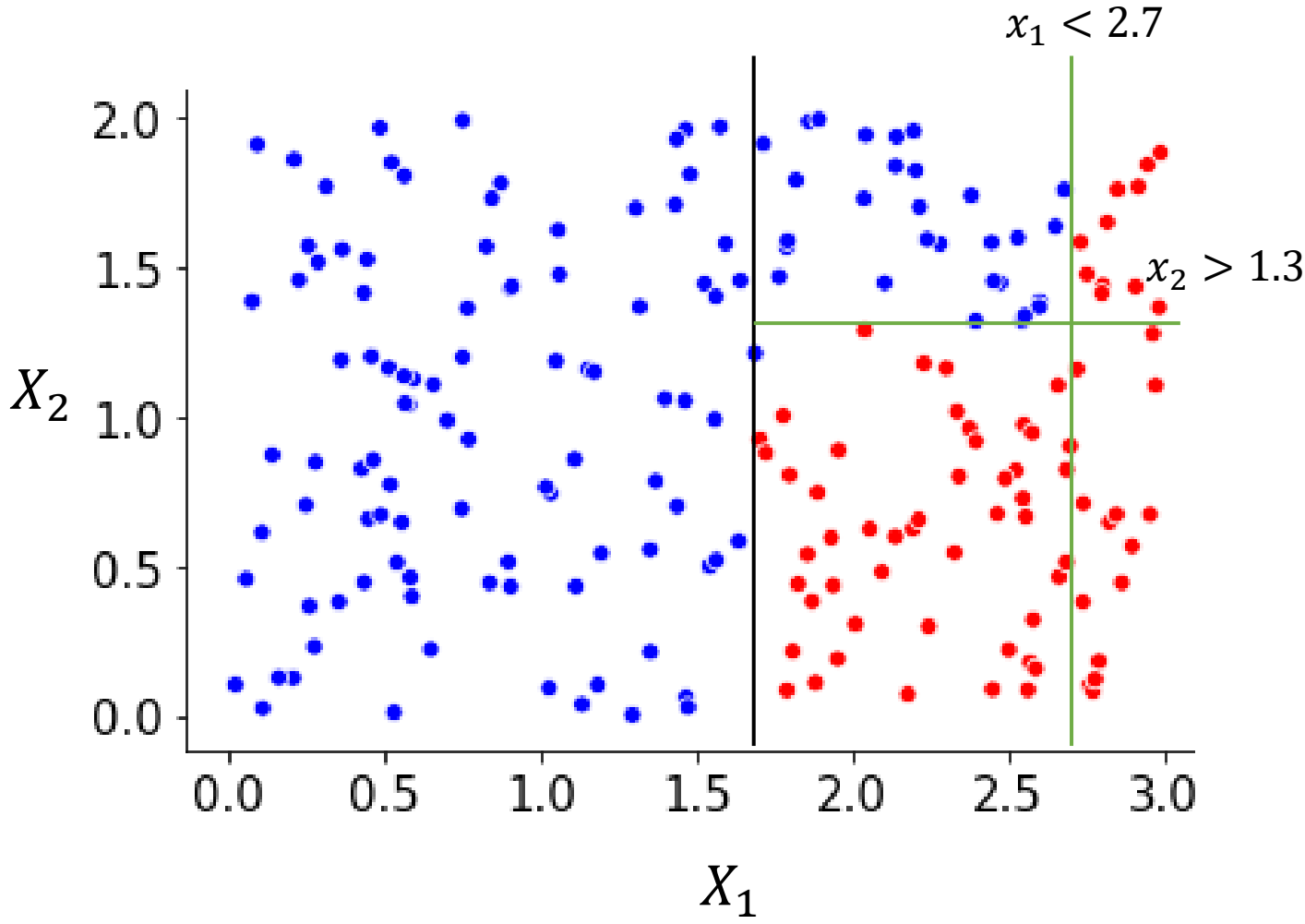
- Pick the leaf to split that reduces uncertainty the most
  - Only one option to begin with
- Pick the best splitting rule for the best feature for the chosen leaf node
  - Two options for features: x_1 and x_2
  - Best for x_1: $x_1 < 1.7$
  - Best for x_2: $x_2 > 1.3$



Decision tree: Solved example

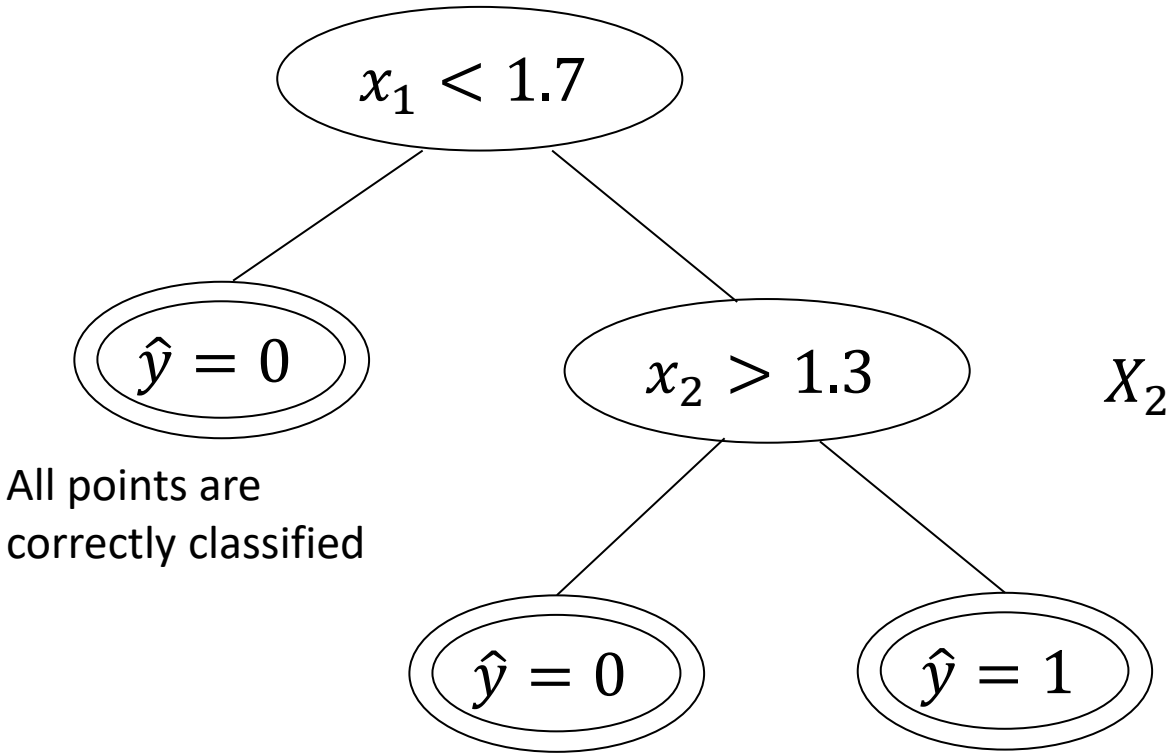


All points are
correctly classified



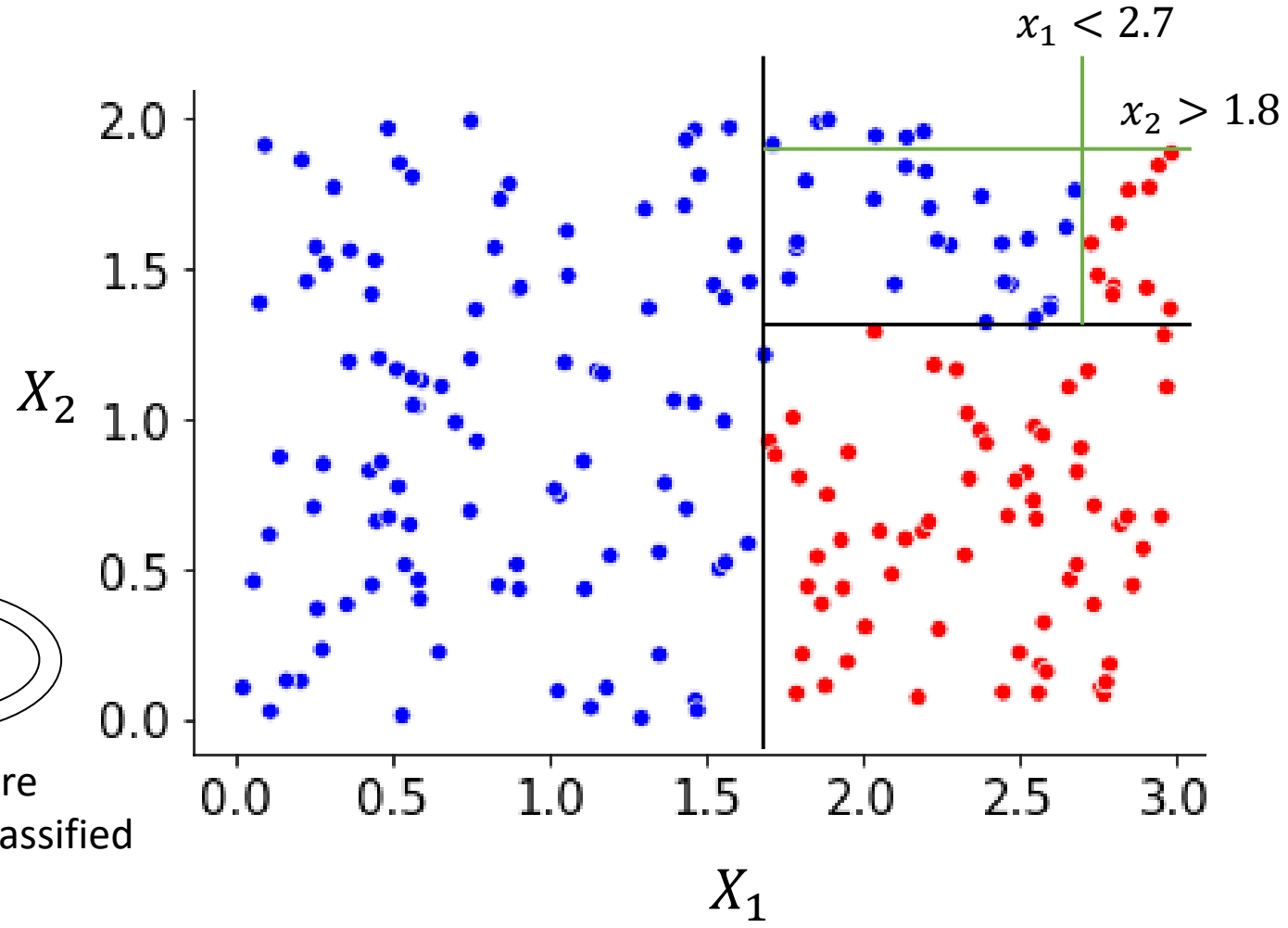
- Pick the leaf to split
- Pick the best splitting rule for the best feature
  - Best for x_1: $x_1 < 2.7$
  - Best for x_2: $x_2 > 1.3$

Decision tree: Solved example



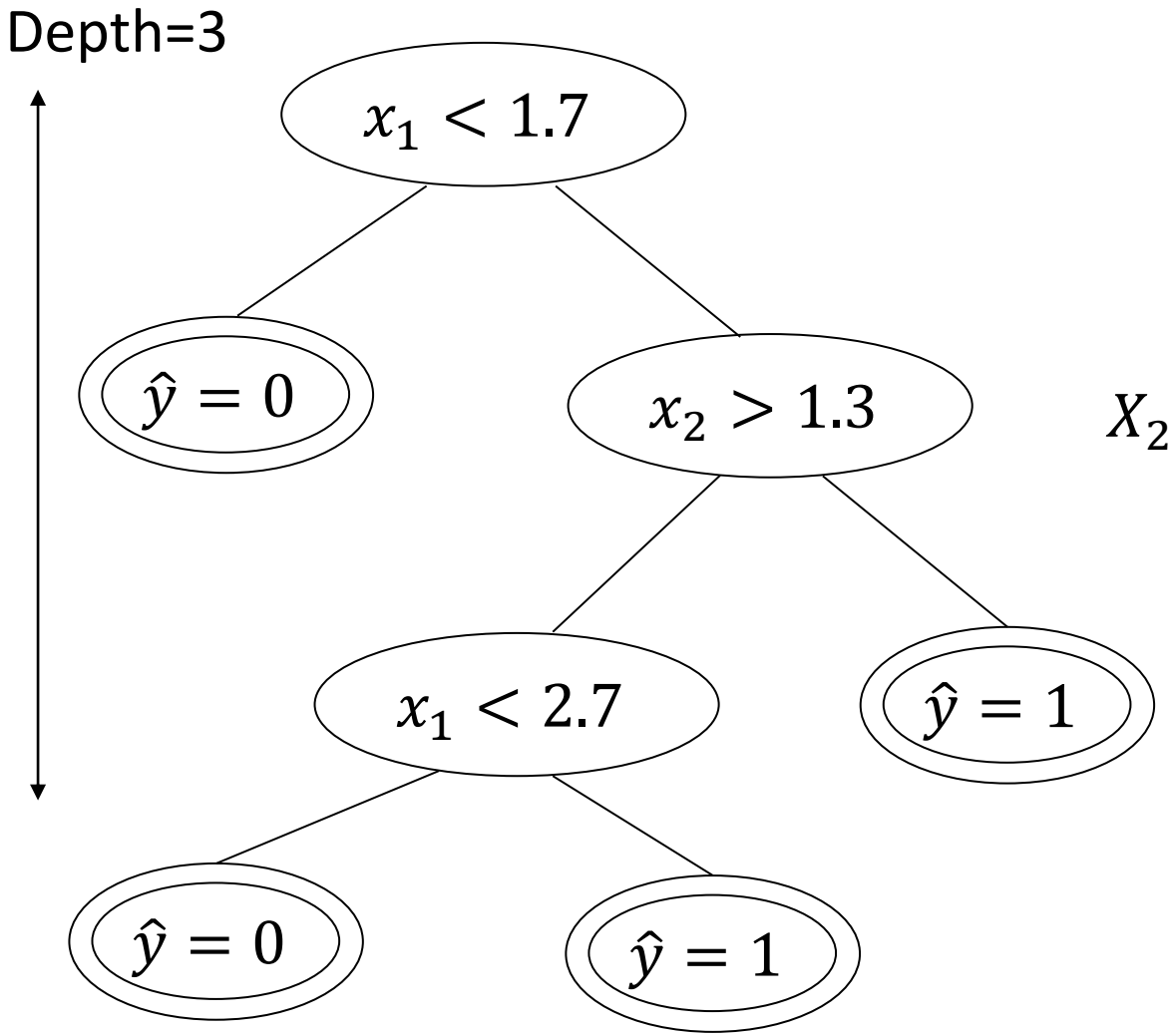
All points are correctly classified

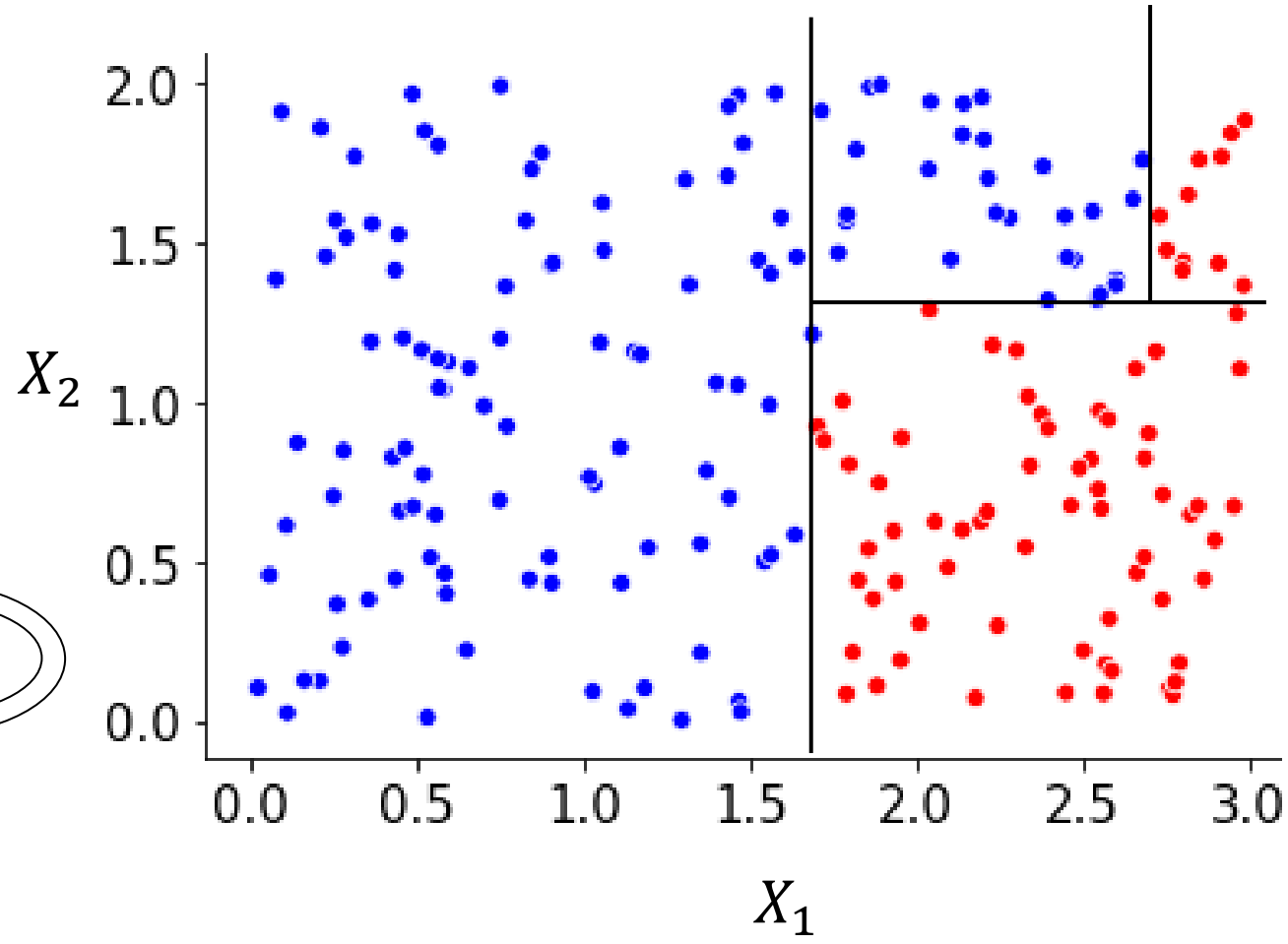
All points are correctly classified



- Best for x_1: $x_1 < 2.7$
- Best for x_2: $x_2 > 1.8$

Decision tree: Solved example





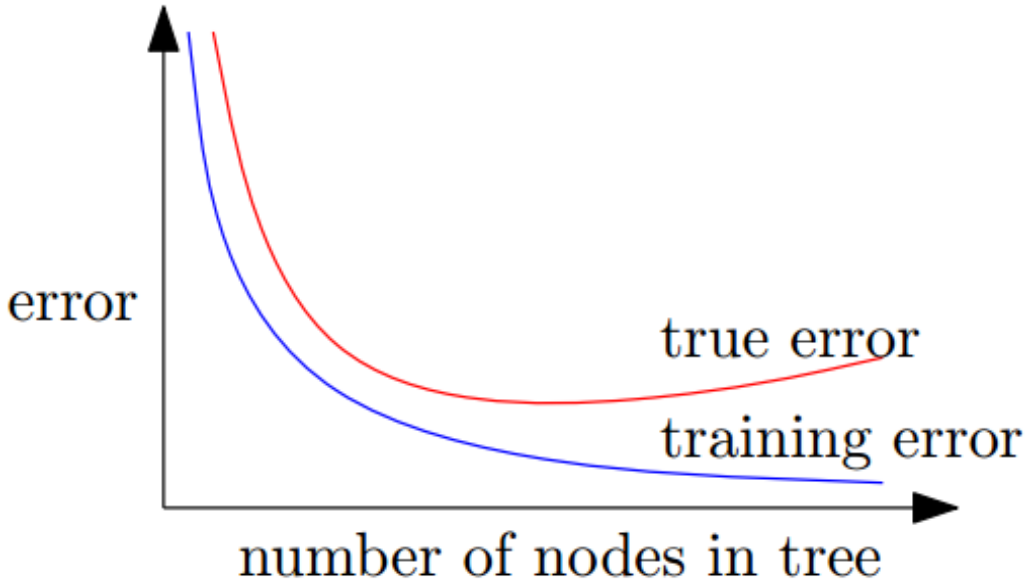
Decision tree: Advantages and limitations

Advantages:

- Non-linear decision boundaries
- Intuitive and easy to interpret

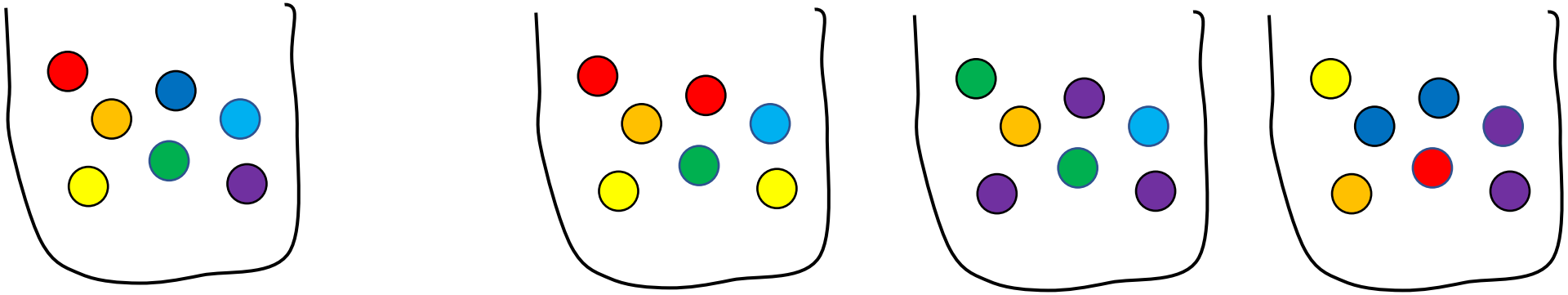
Limitations:

- Decision trees are prone to overfitting
  - Training error decreases as nodes increase
  - Testing error decreases and then increases due to overfitting



Random forest – Bagging

- Use multiple decision trees, each trained with a subset of the training data
  - Let training data have N samples
  - For each tree, randomly select N samples (with replacement) from the training data
  - Repeat this for each tree
- Aggregate the decision from multiple trees for the final decision



Training data

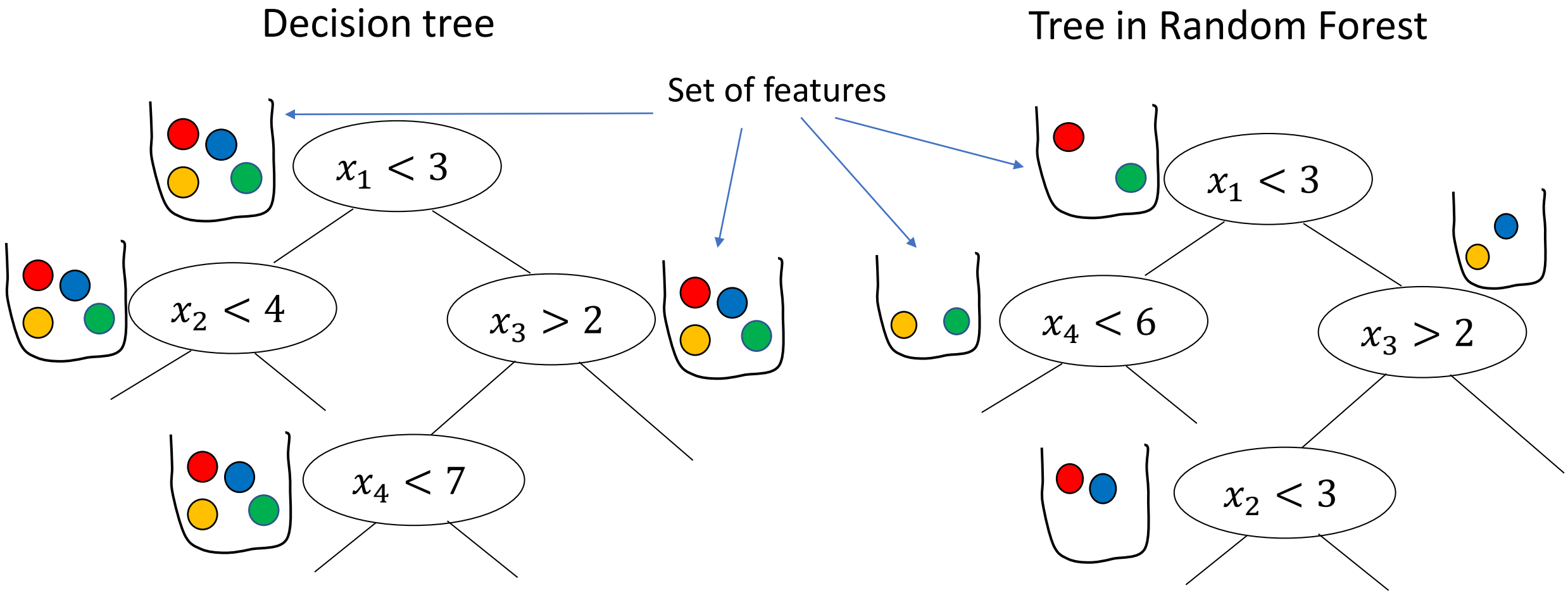
Bootstrap sample 1

Bootstrap sample 2

Bootstrap sample 3

Random forest – random feature set

- At each split, only consider a random subset of m features to choose the splitting rule
- m should be specified; typically $m \approx \sqrt{d}$ where d is the dimensionality of the data



Random forest algorithm

Algorithm:

1. Draw a bootstrap sample (sample data with replacement) $\mathcal{B}_b$ of the same size N from the training data
2. Train a tree f_b on $\mathcal{B}_b$, where each split is computed as follows:
   i. Randomly select m dimensions of $x \in R^d$, newly chosen for each splitting rule
   ii. Make the best split restricted to the subset of dimensions

Advantages

- Applicable to both regression and classification problems
- Handle categorial predictors naturally
- Computationally simple and quick to fit, even for large problems
- Can handle highly non-linear interactions and classification boundaries
- Offers some interpretability; though not as interpretable as Decision Trees
- Provides variable/feature importance

Questions?