Small data technique

Lecture 13

Changho Suh

January 26, 2024

Decision trees (DTs)

Recap: DNNs

Work well with **enough data**.

Otherwise, we may face: Overfitting problem

This motivates simplifying DNNs, being tailored for tasks of interest.

Recap: CNNs

A model specialized for image data

Two key building blocks:

1. **Conv** layer (*mimicking* neurons in *visual cortex*)
2. **Pooling** layer (*mainly for reducing complexity*)

Design principles: As a network gets deeper:

1. Feature map size gets smaller;
2. # of feature maps gets bigger.

Recap: RNNs

A model specialized for time series data

Two key building blocks:

1. **Recurrent neurons**
2. **Memory cell**

Basic RNNs: Trained via truncated BTTP;
Do not well keep memory.

LSTM: Offers great performance and fast training.

Recap: Tensorflow coding for RNNs

```
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
from tensorflow.keras.layers import LSTM

(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=10000)
# Preprocessing
X_train_pad = pad_sequences(X_train, value=0, padding='post', maxlen=256)

# Basic RNN
model = Sequential()
model.add(Embedding(input_dim=10000,output_dim=100,input_length=256))
model.add(SimpleRNN(128))
model.add(Dense(1, activation='sigmoid'))

# LSTM
model_LSTM = Sequential()
model_LSTM.add(Embedding(input_dim=num_words,output_dim=100,input_length=256))
model_LSTM.add(LSTM(128))
model_LSTM.add(Dense(1, activation='sigmoid'))
```

Questions

1. What if we still have unsatisfactory performances?

   A better approach for the small data regime?

2. What about interpretability of DNNs?

Today's lectures

Will explore a technique that may enable a better performance for the small data regime, as well as offer model interpretability:

Random forests (RFs)

The **most powerful** ML algorithm in industry

Outline of today's lectures

Will study:

1. **Decision trees (DTs)**:

   Fundamental components of RFs

2. **Ensemble learning**:

A generic technique that includes RFs as a special case.

3. **RFs** in depth

Focus of Lecture 13

Will study:

1. **Decision trees (DTs)**:

   Fundamental components of RFs

2. **Ensemble learning**:

A generic technique that includes RFs as a special case.

3. **RFs** in depth

A motivating example

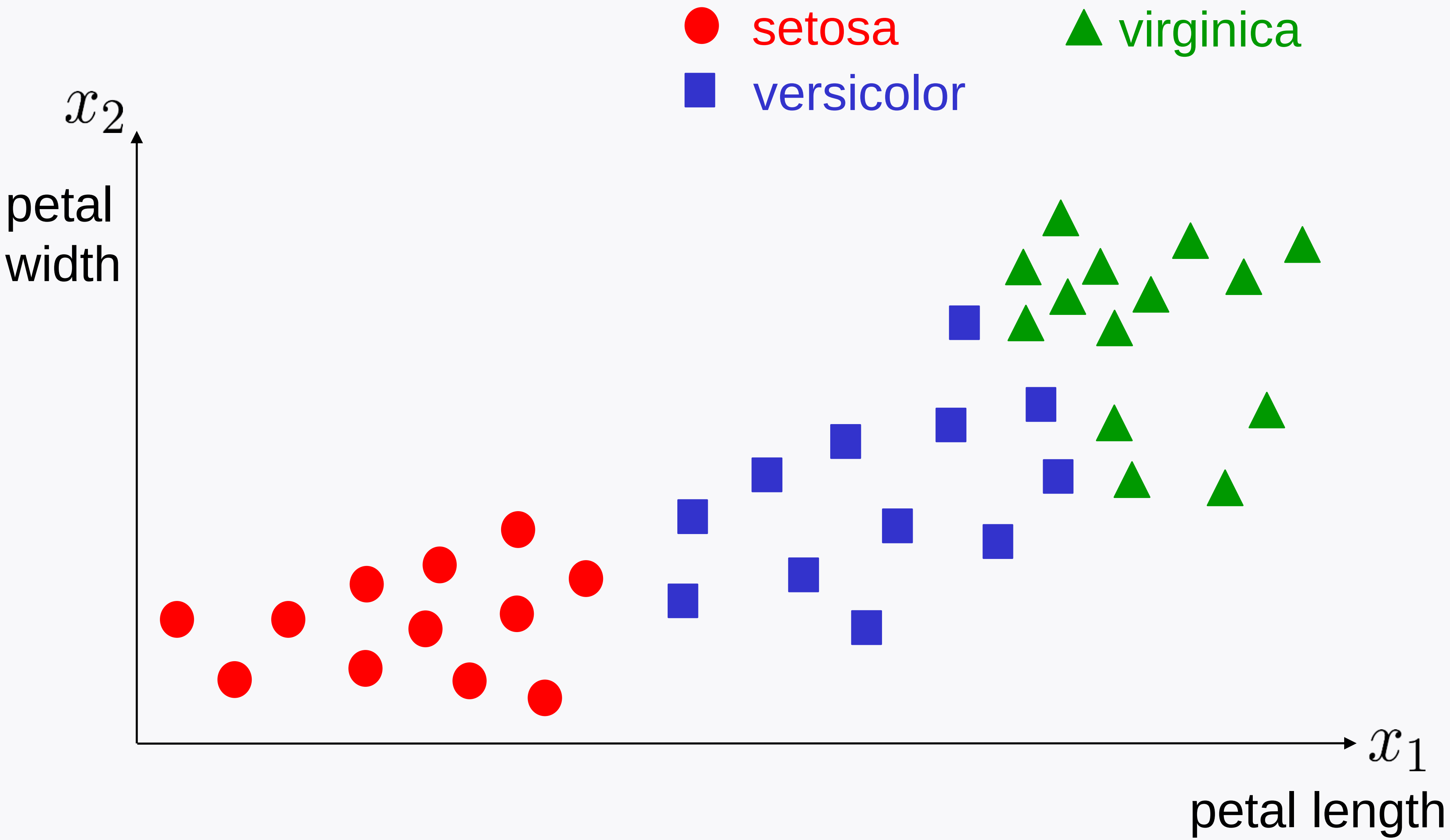
Iris plants classification:

Class: setosa versicolor virginica

Two features (out of 4): x_1 : petal length

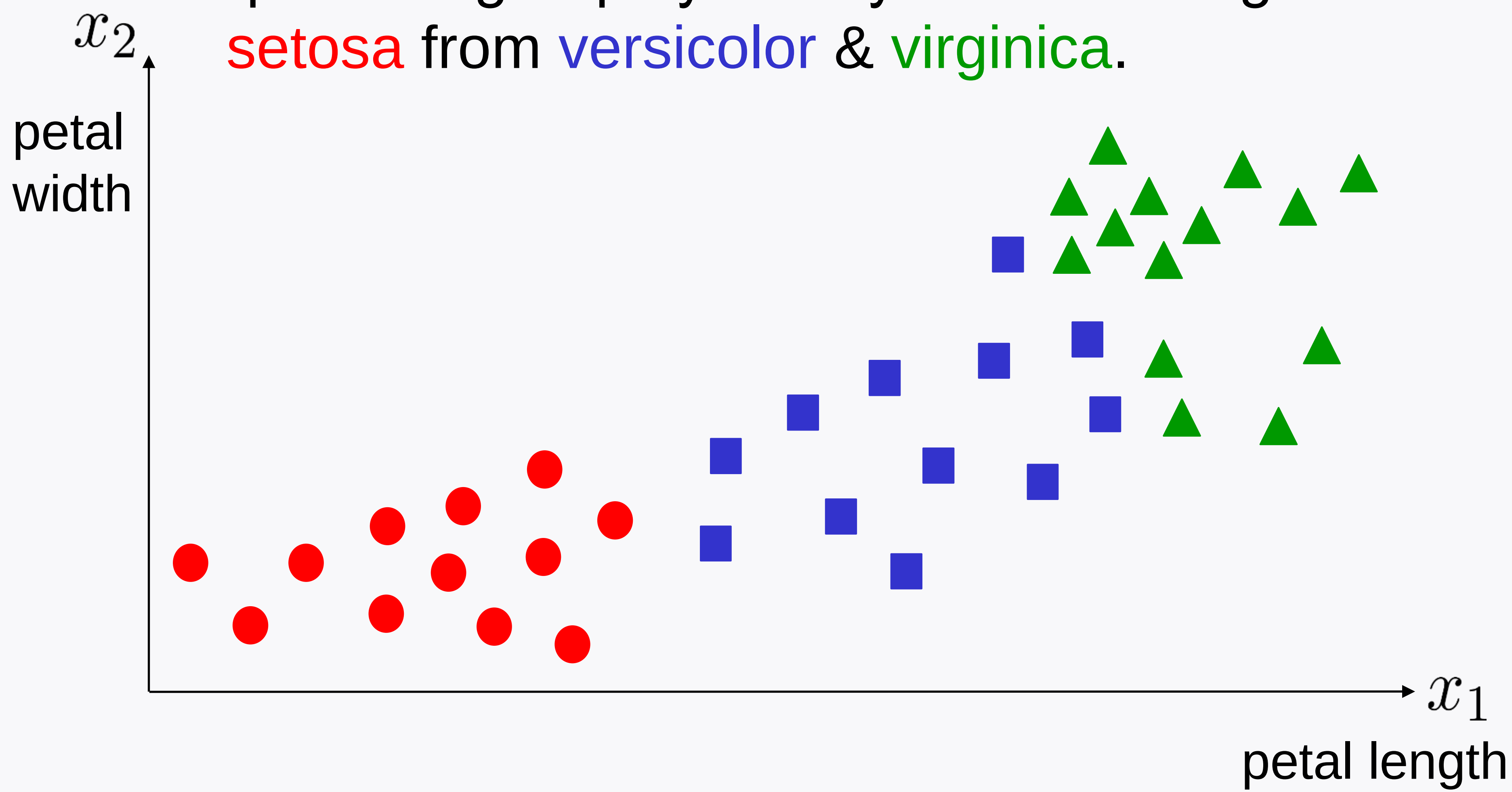
x_2 : petal width

Data distribution

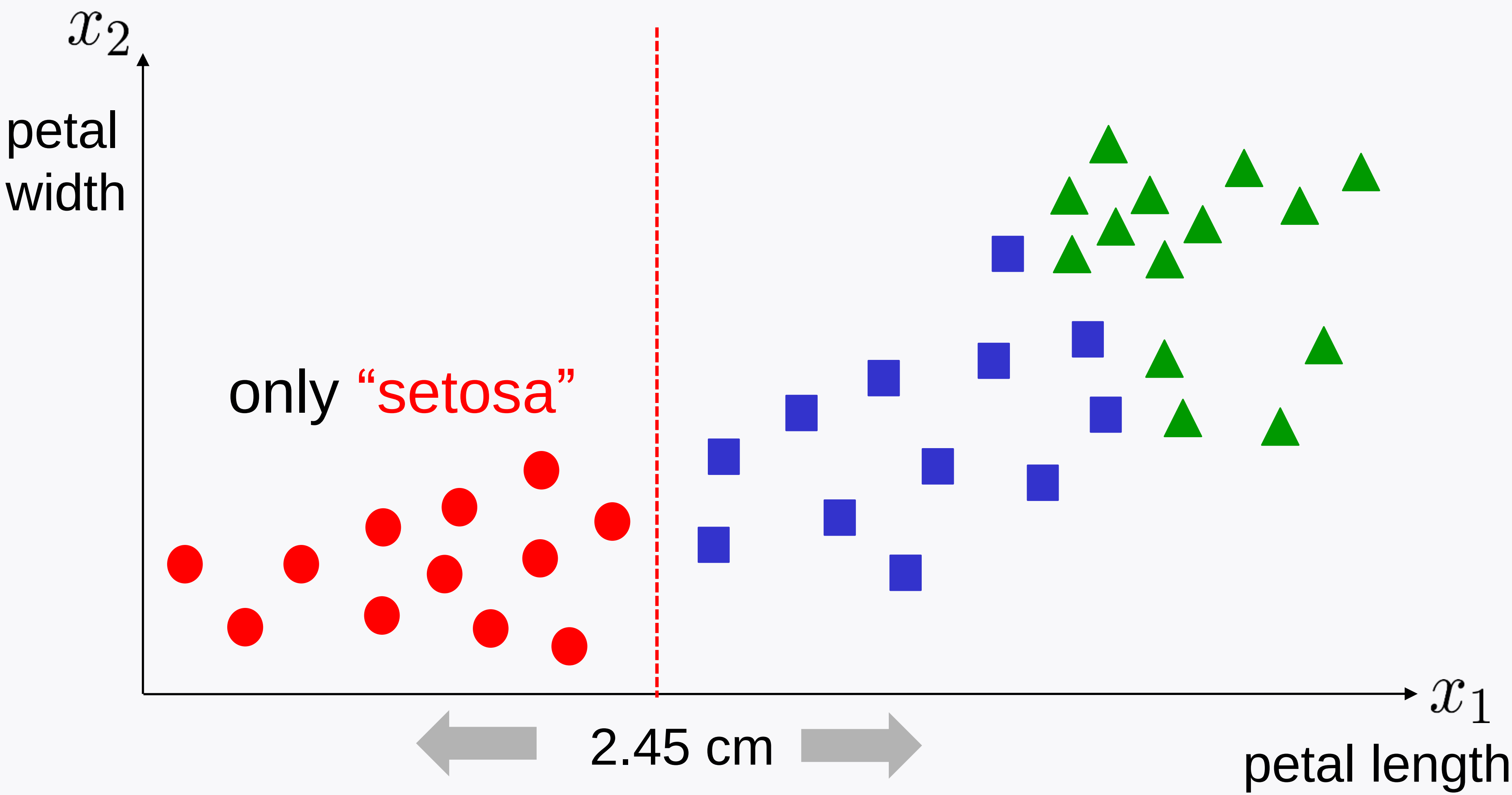


Observation

"petal length" plays a key role to distinguish setosa from versicolor & virginica.

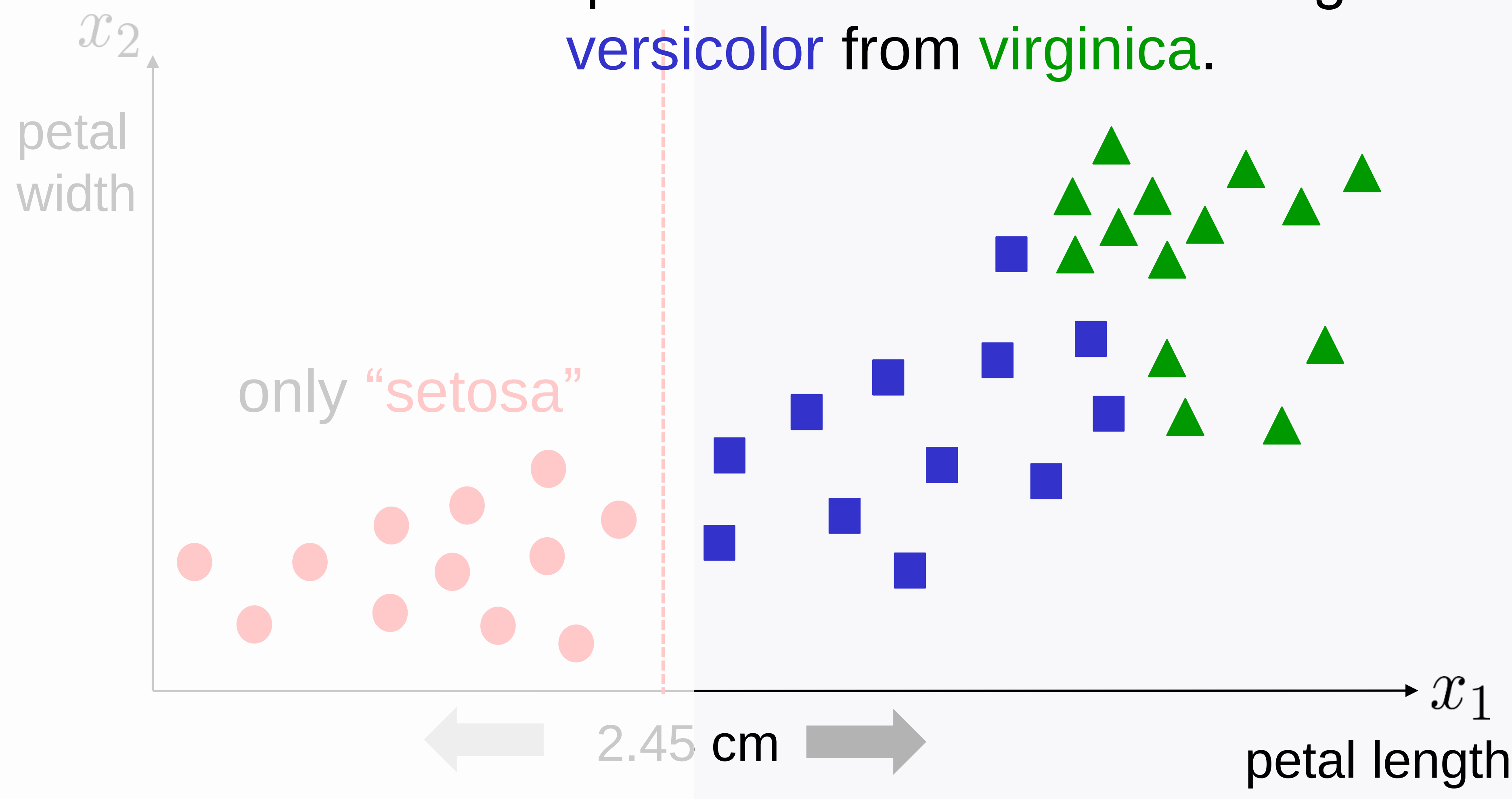


A natural attempt for classification

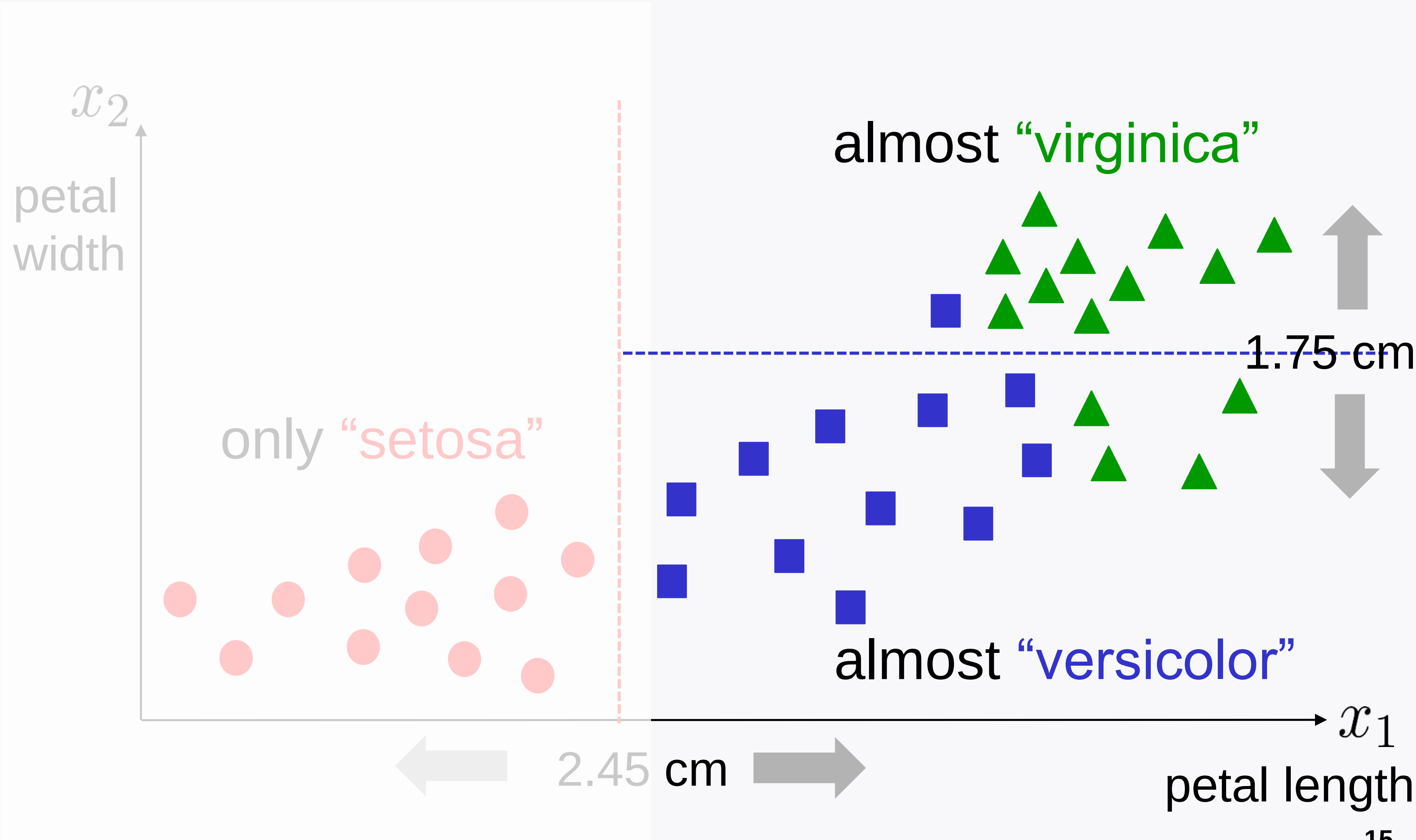


Another observation

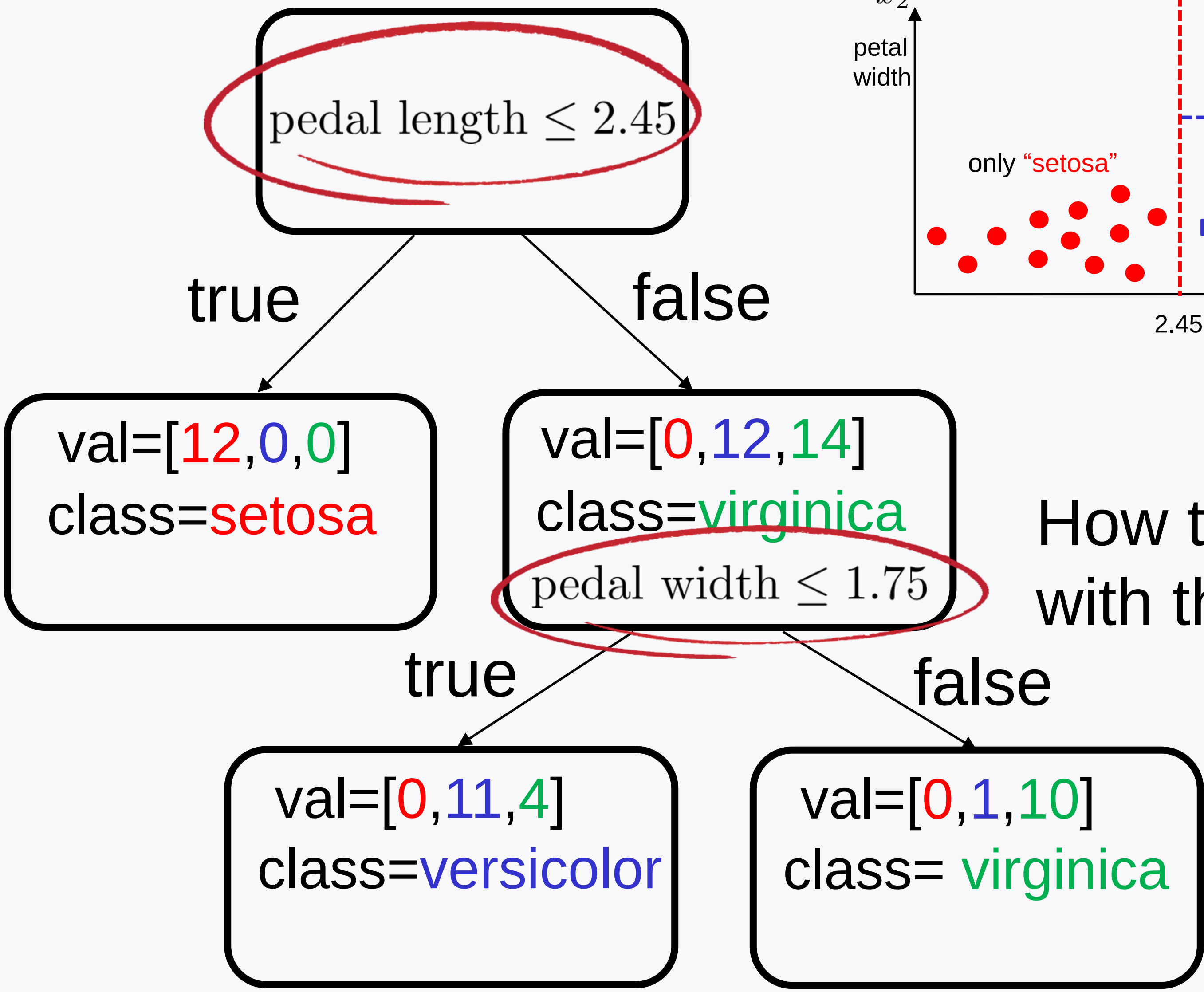
"petal width" serves to distinguish versicolor from virginica.

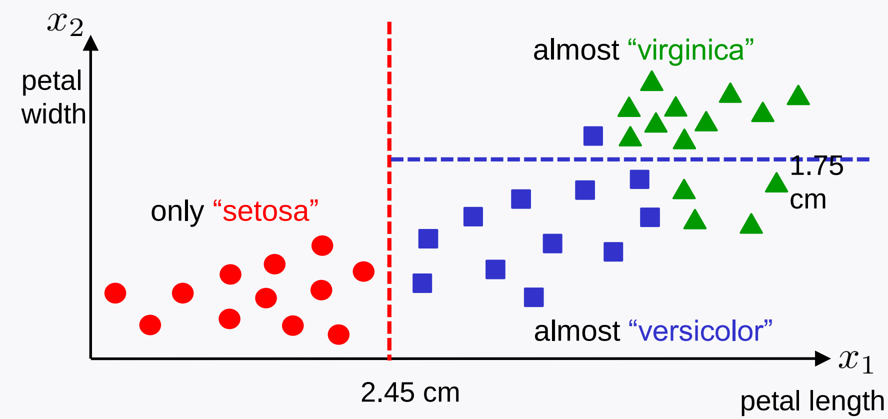


A follow-up natural attempt



Decision Tree



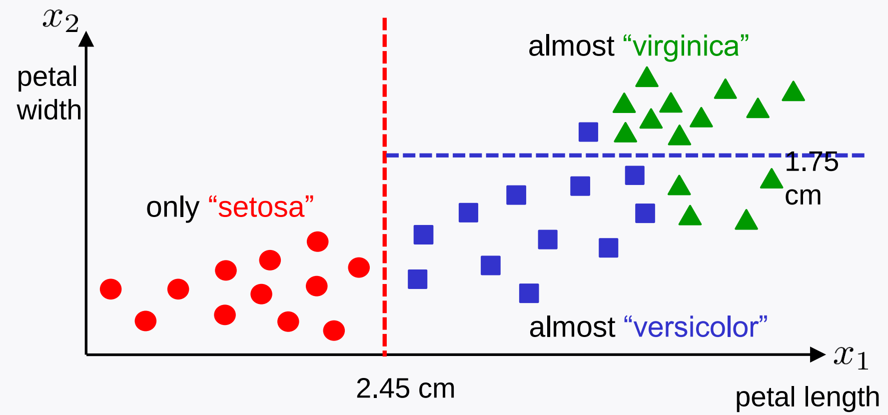


How to come up with the **conditions**?

CART (Classification And Regression Tree) algorithm

k : feature index

t_k : threshold



Step 1: Find (k, t_k) such that $J(k, t_k)$ is minimized.

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

smaller → more pure

impurity of the left split: Gini index (0~1)

$$G_{\text{left}} := 1 - \sum_{c=1}^{3} r_{\text{left},c}^2 = 1 - (1^2 + 0^2 + 0^2) = 0$$

$$G_{\text{right}} = 1 - \left(0^2 + \left(\frac{12}{26}\right)^2 + \left(\frac{14}{26}\right)^2\right) = 0.497$$

CART (Classification And Regression Tree) algorithm

Step 1: Find (k, t_k) such that $J(k, t_k)$ is minimized.

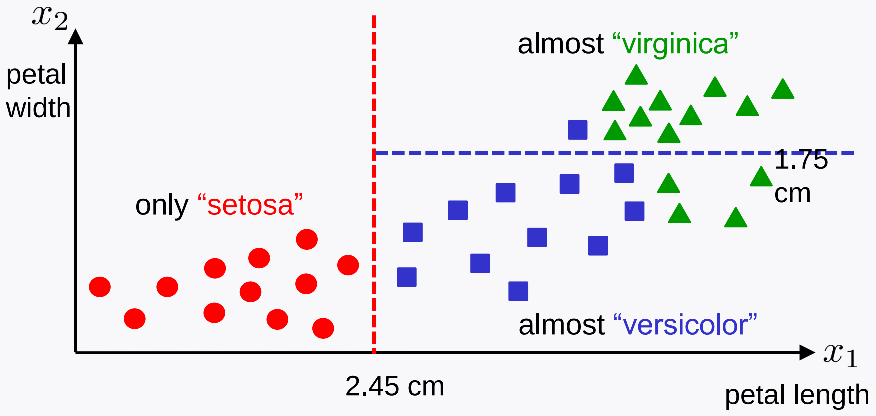
$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

Step 2: Repeat Step 1 for each split:

$$G_{\text{left}} \begin{array}{l} G_{\text{left,left}} \\ \\ G_{\text{left,right}} \end{array} \qquad G_{\text{right}} \begin{array}{l} G_{\text{right,left}} \\ \\ G_{\text{right,right}} \end{array}$$

Stopping criteria?

Stopping criteria



1. Cannot find a split that further reduces impurity.

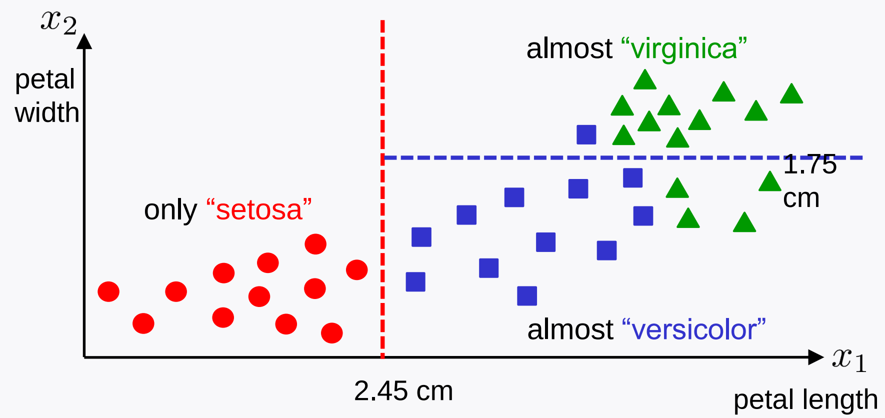
OR

hyperparameter

2. Reach "**max_depth**" (maximum tree depth).

max_depth=2 (in the example)

Hyperparameters



1. **"max_depth"**

2. **"min_samples_split"**

   Min # of samples a node must have prior to splitting.

3. **"min_samples_leaf"**

   Min # of samples a leaf node must have.

4. **"max_leaf_nodes"**

   Max # of leaf nodes

Hyperparameters vs. regularization

1. "**max_depth**" ↓ ➡ More regularized

2. "**min_samples_split**" ↑ More regularized

   Min # of samples a node must have prior to splitting.

3. "**min_samples_leaf**" ↑ More regularized

   Min # of samples a leaf node must have.

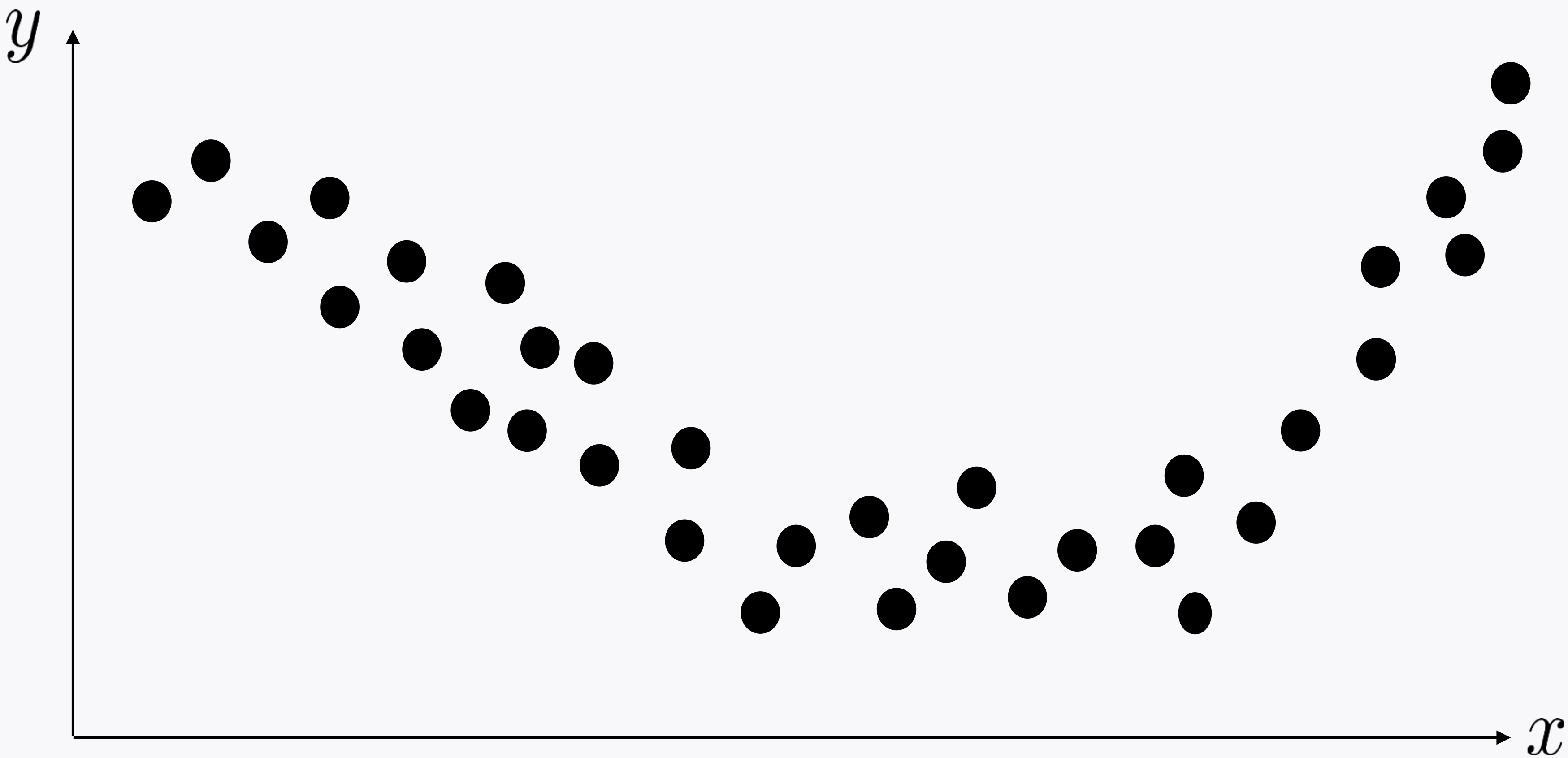
4. "**max_leaf_nodes**" ↓ More regularized

   Max # of leaf nodes

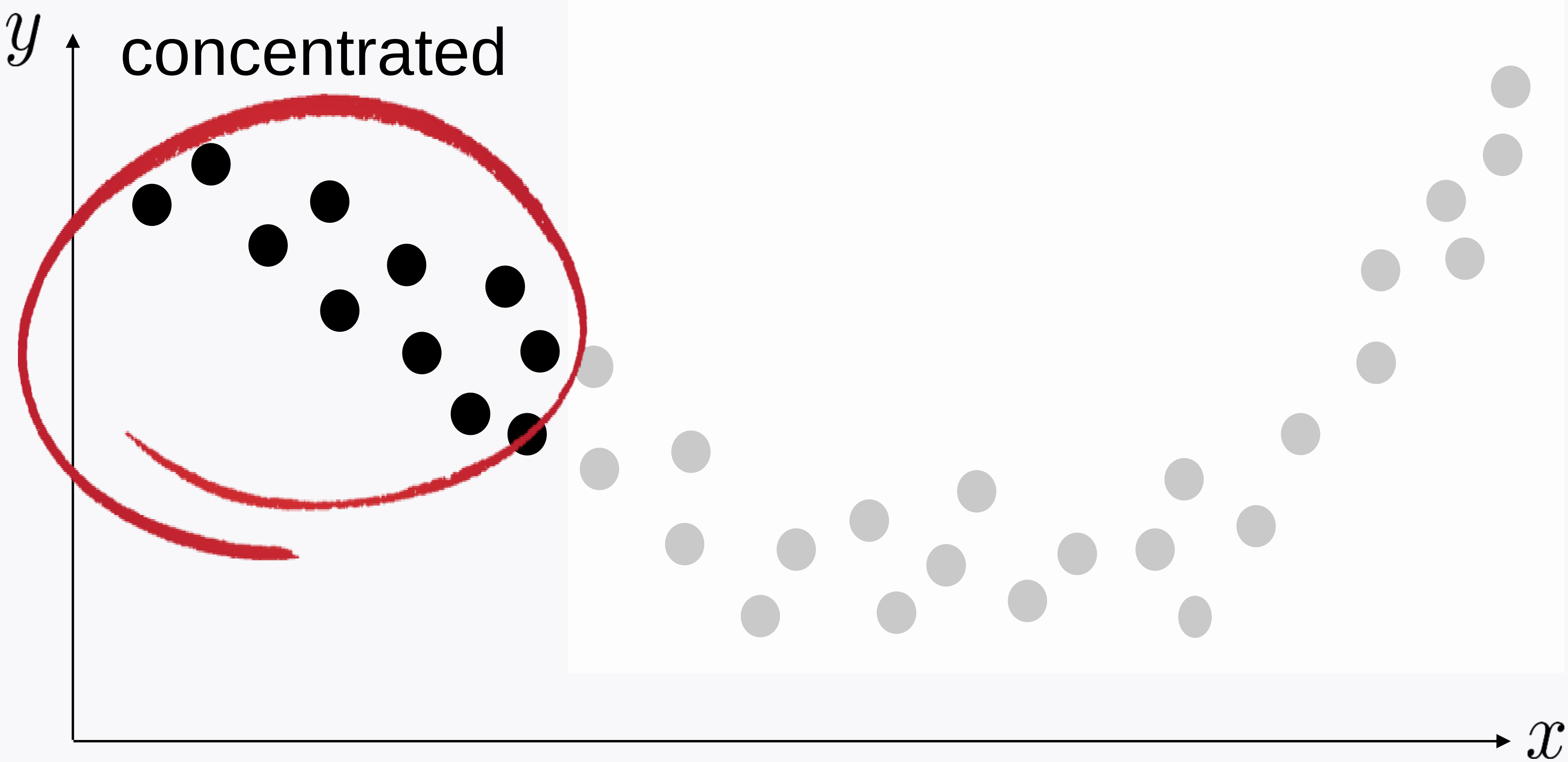
DTs for regression

A motivating example

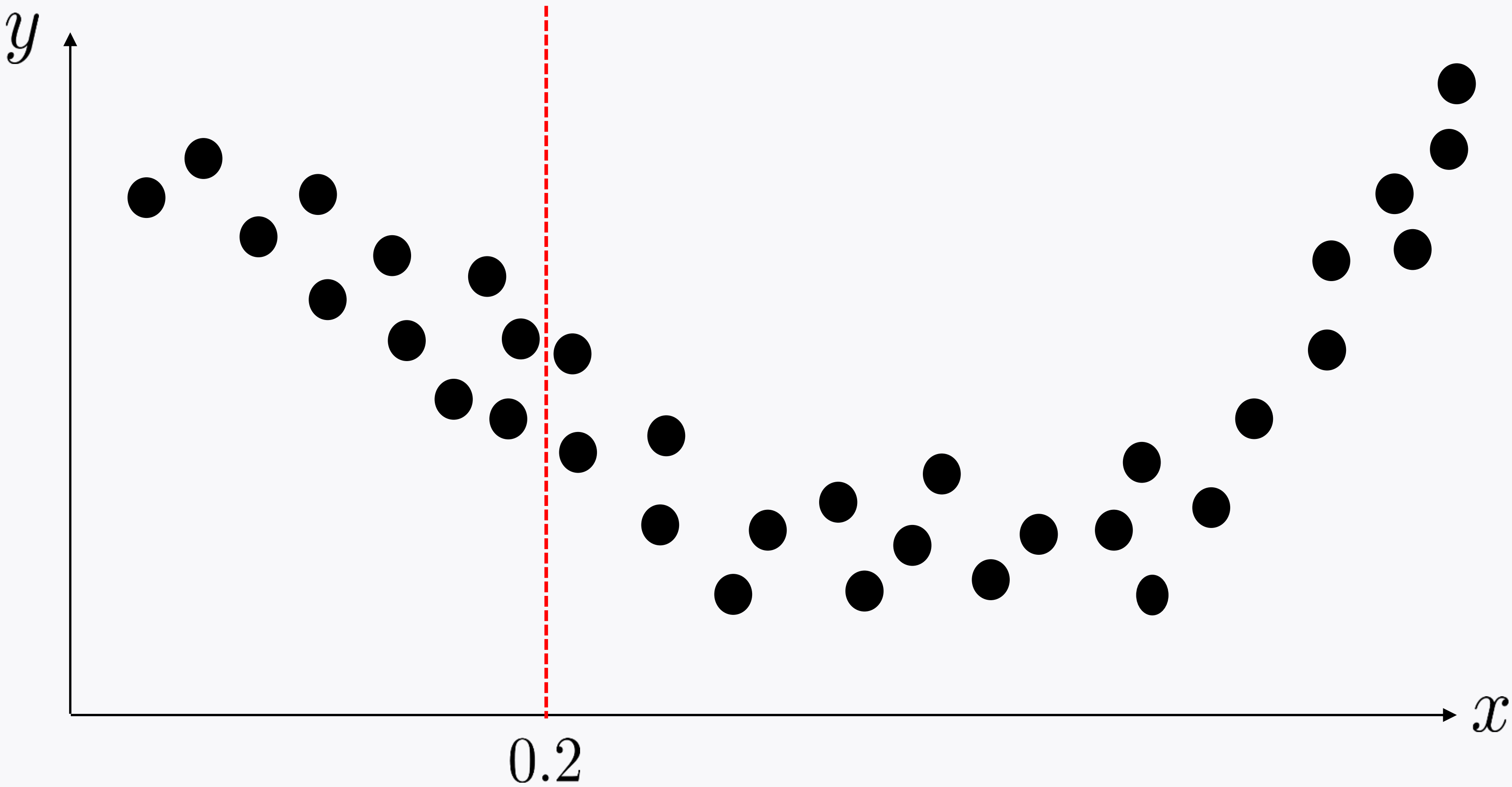
$x \in \mathbf{R} \quad y \in \mathbf{R}$



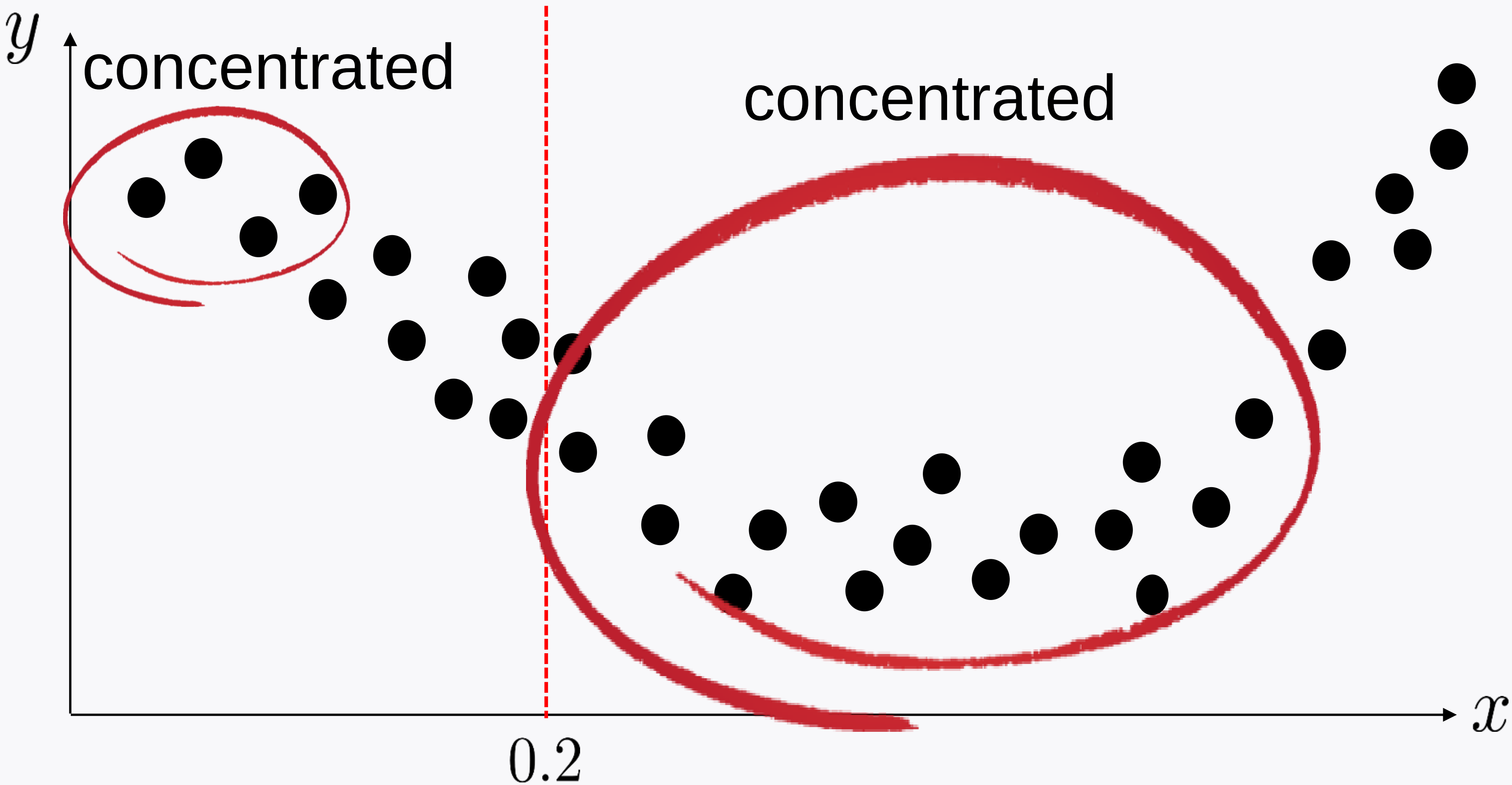
Observation



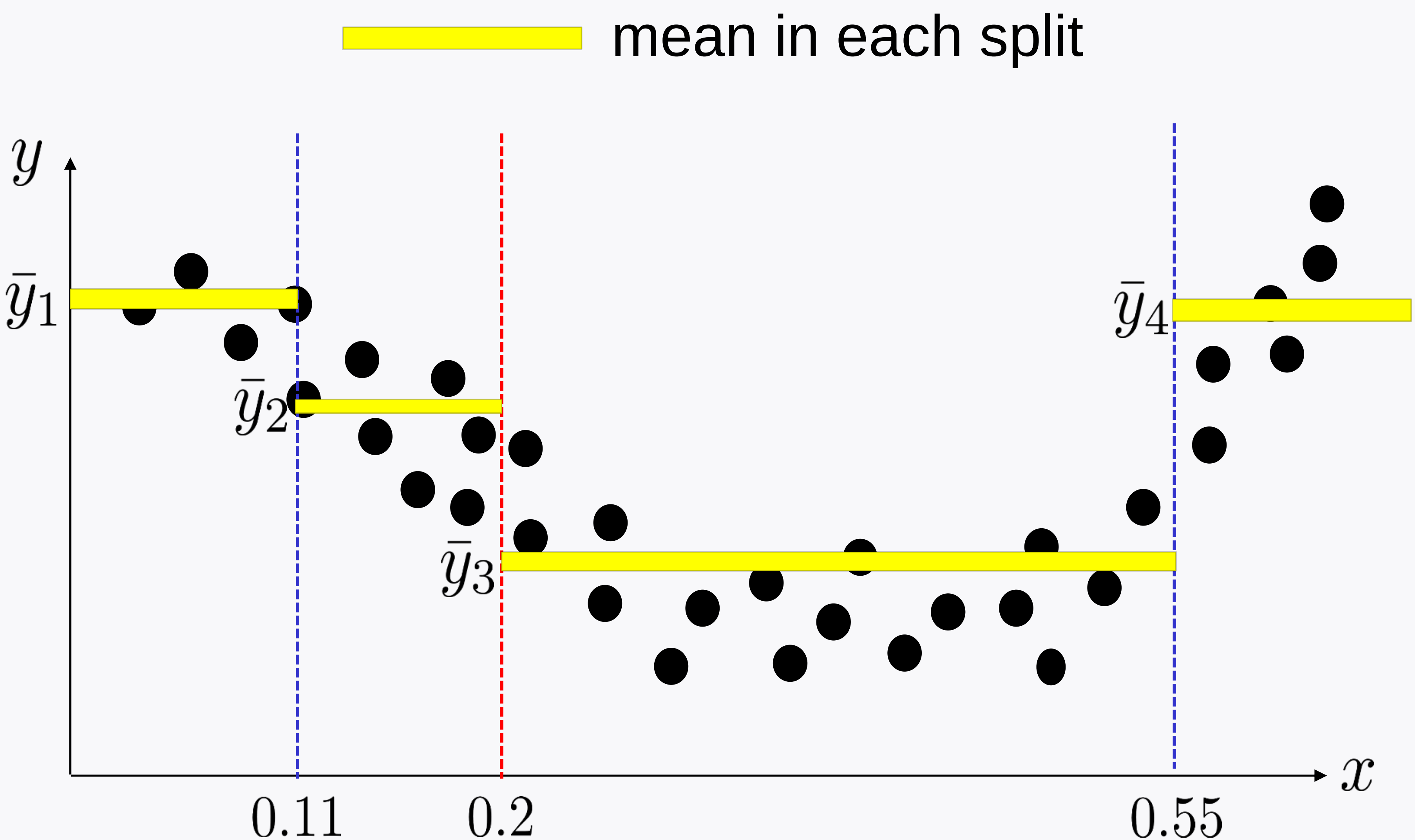
A natural attempt for separation



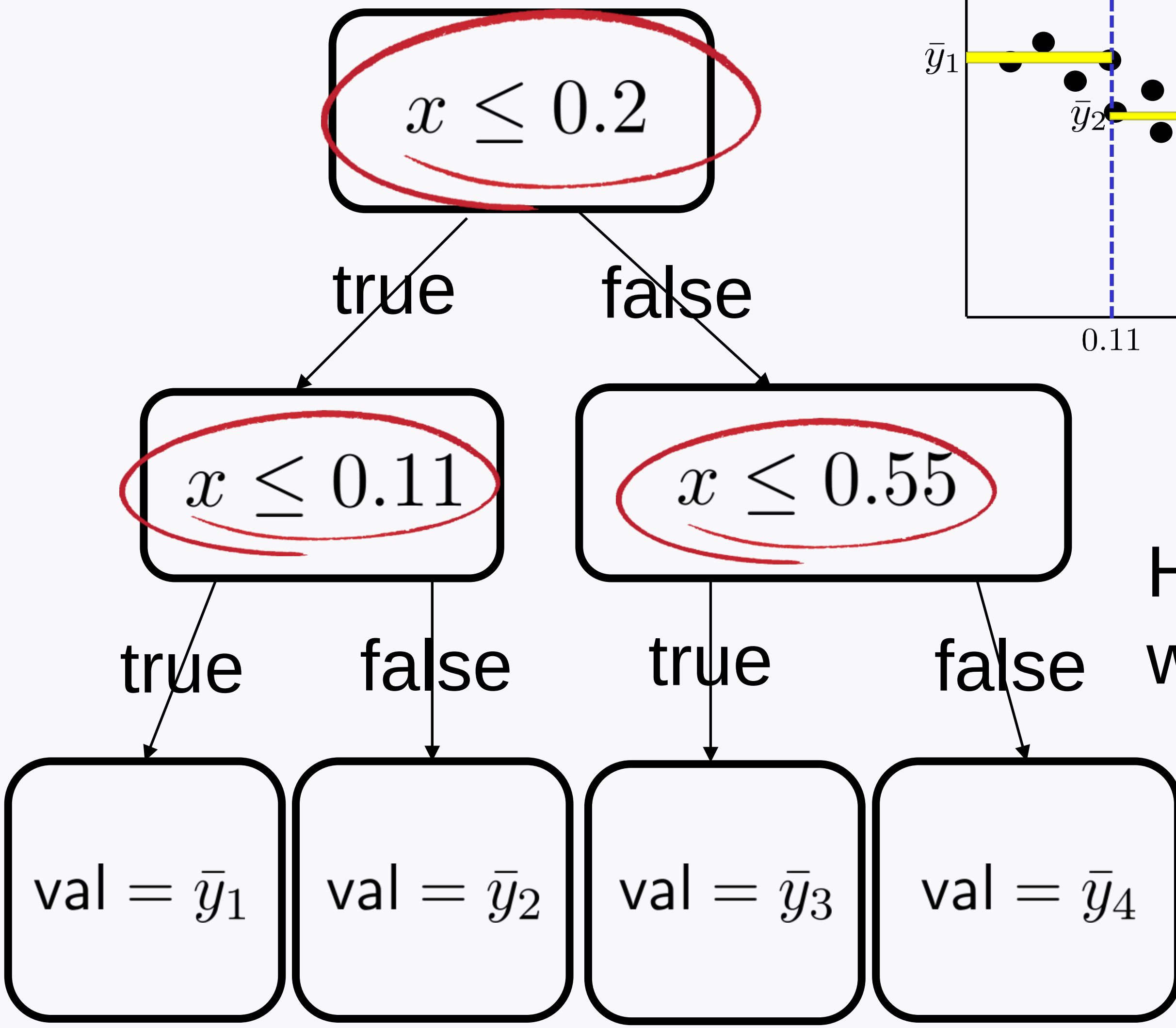
Observation in each split

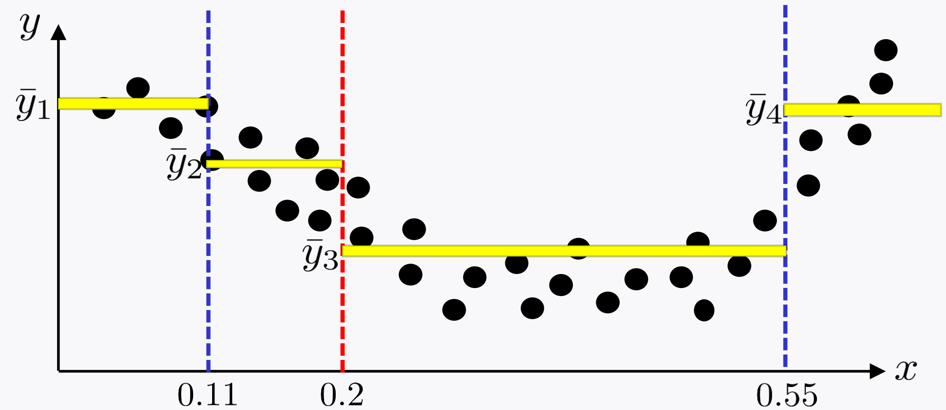


A follow-up natural attempt



Decision tree



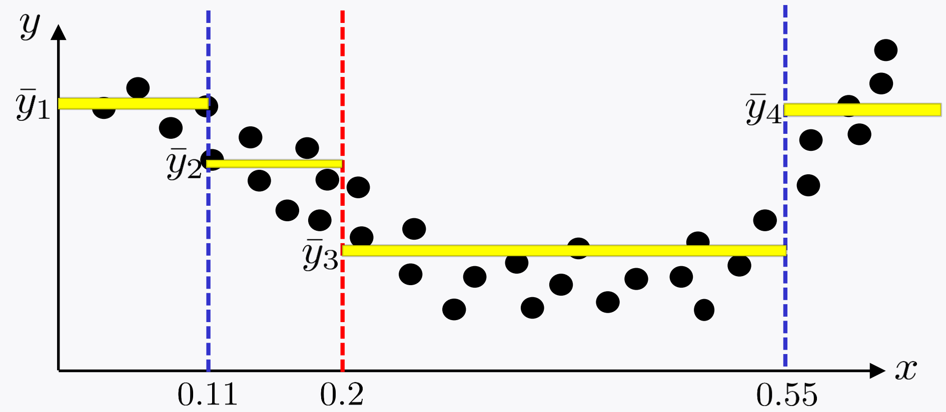


How to come up with the conditions?

CART algorithm

k : feature index

t_k : threshold



Step 1: Find (k, t_k) such that $J(k, t_k)$ is minimized.

$$J(k, t_k) = \frac{m_{\text{left}}}{m} \text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m} \text{MSE}_{\text{right}}$$

$$\text{MSE}_{\text{left}} := \frac{1}{m_{\text{left}}} \sum_{i \in \text{left}} \left(y^{(i)} - \bar{y}_{\text{left}}\right)^2 \quad \bar{y}_{\text{left}} = \frac{1}{m_{\text{left}}} \sum_{i \in \text{left}} y^{(i)}$$

CART algorithm

Step 1: Find (k, t_k) such that $J(k, t_k)$ is minimized.

$$J(k, t_k) = \frac{m_{\text{left}}}{m} \text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m} \text{MSE}_{\text{right}}$$

Step 2: Repeat Step 1 for each split:

$$\text{MSE}_{\text{left}} \begin{cases} \text{MSE}_{\text{left,left}} \\ \text{MSE}_{\text{left,right}} \end{cases} \qquad \text{MSE}_{\text{right}} \begin{cases} \text{MSE}_{\text{right,left}} \\ \text{MSE}_{\text{right,right}} \end{cases}$$

Stopping criteria & hyperparameters are the same as those of classification.

Look ahead

1. Investigate a challenge that arises in DTs.

2. Explore a way to address the challenge:

Ensemble learning