Classification and Regression Trees

Note: Unless otherwise noted all references including images are from the required textbook, Machine Learning: A Probabilistic Perspective by Kevin P. Murphy.

Adaptive Basis Function Models

Kernel methods rely on having a good kernel function to measure the similarity between data vectors. Often coming up with a good kernel function is quite difficult.

An alternative approach is to try to learn useful features $\phi(\mathbf{x})$ directly from the input data. That is achieved by creating an **adaptive basis-function model** (**ABM**).

Adaptive Basis Function Models

An adaptive basis-function model has the form

$$f(\mathbf{x}) = w_0 + \sum_{m=1}^{M} w_m \phi_m(\mathbf{x})$$

where $\phi_m(\mathbf{x})$ is the m^{th} basis function, which is learned from data.

Adaptive Basis Function Models

Typically the basis functions are parametric, so we can write $\phi_m(\mathbf{x}) = \phi(\mathbf{x}, \mathbf{v}_m)$, where $\mathbf{v}_m$ are the parameters of the m^{th} basis function.

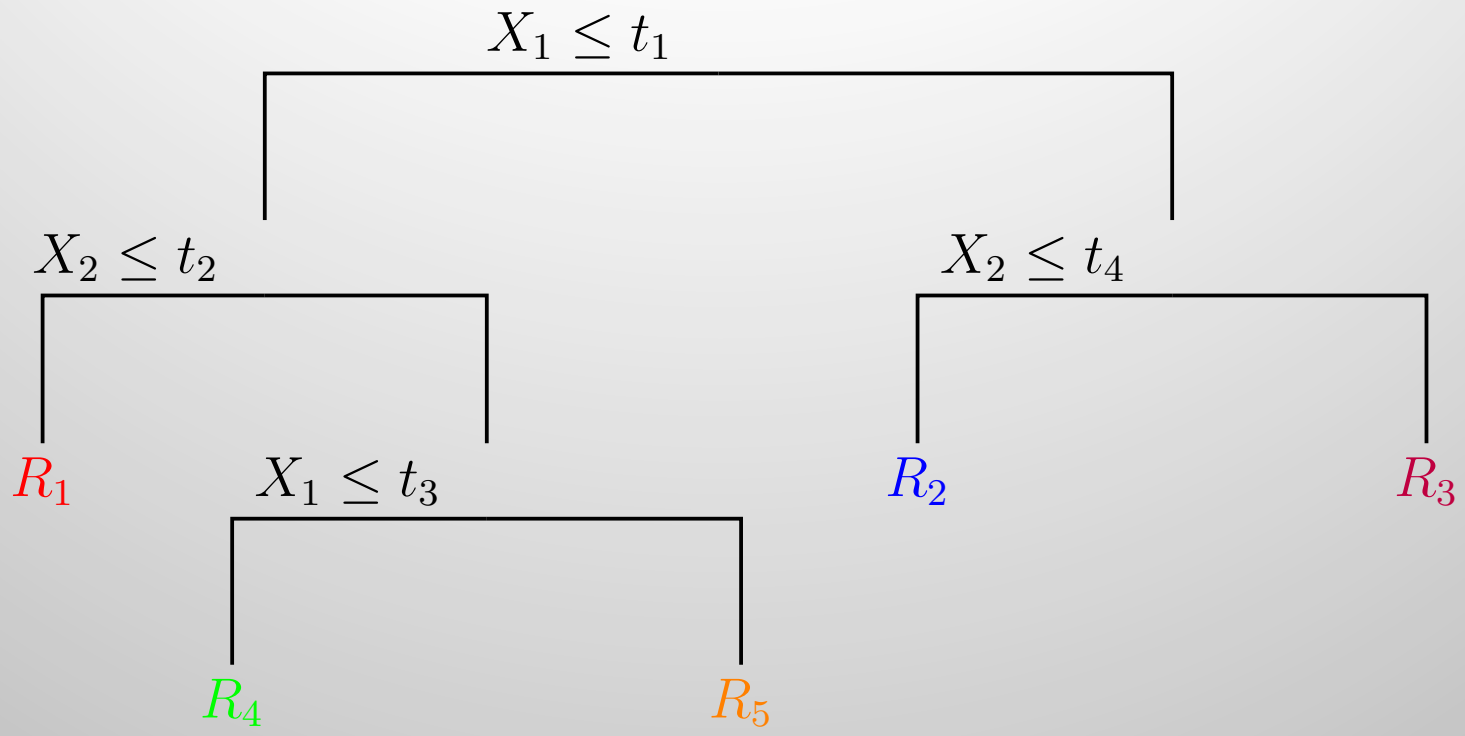
We will use $\boldsymbol{\theta} = (w_0, \mathbf{w}_{1:M}, \{\mathbf{v}_m\}_{m=1}^M)$ to denote the entire parameter set.

The resulting model is not linear-in-the-parameters anymore and often significantly outperforms linear models.

Classification and Regression Trees (CART)

Classification and regression trees or **CART** models, also called decision trees are defined by recursively partitioning the input space, and defining a local model in each resulting region of input space. This can be represented by a tree, with one leaf per region.

Classification and Regression Trees (CART)



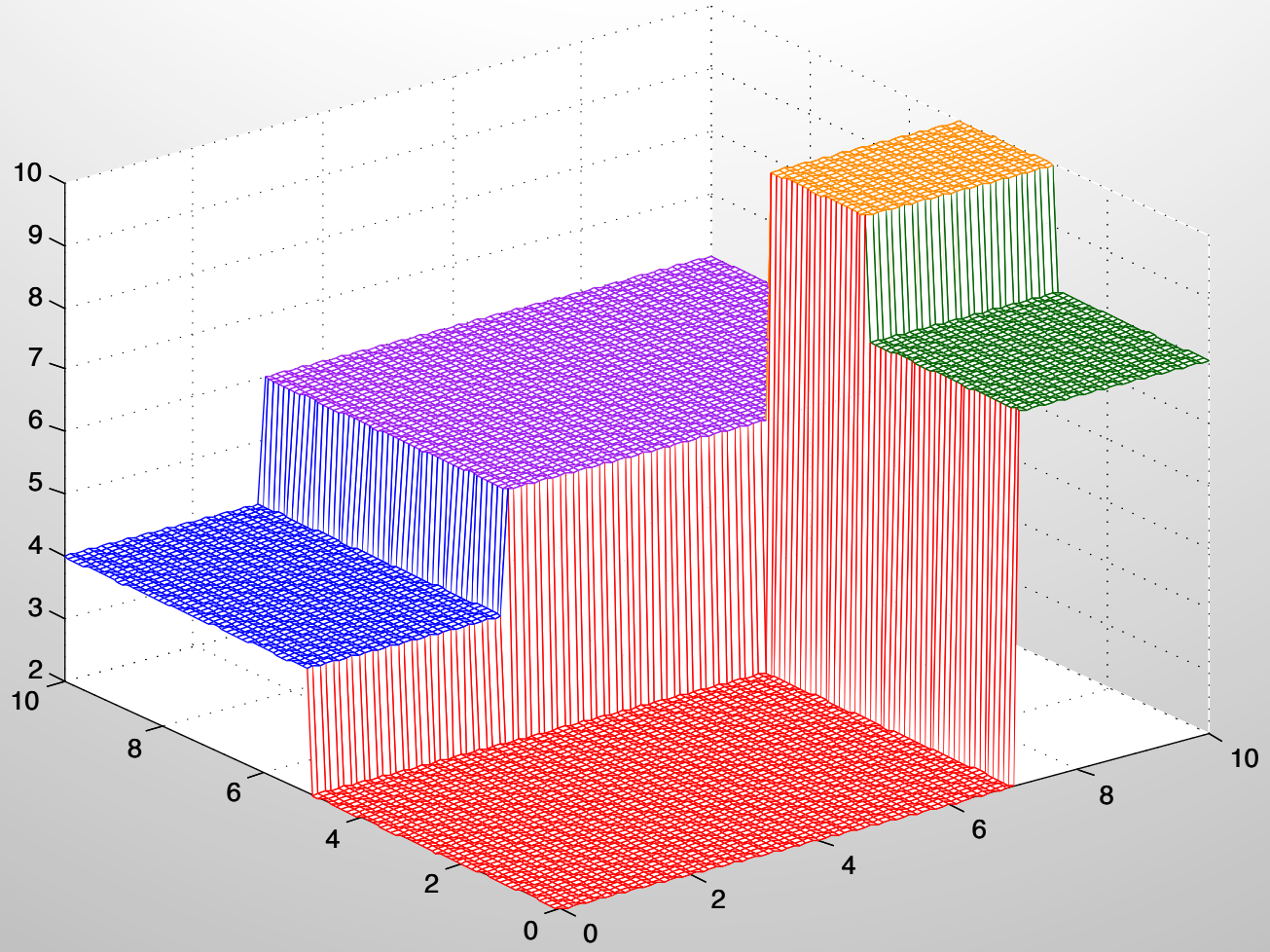
A simple regression tree on two inputs

Classification and Regression Trees (CART)

The first node asks if x_1 is less than some threshold t_1. If yes, we then ask if x_2 is less than some other threshold t_2. If yes, we are in region R_1. If no, we ask if x_1 is less than t_3. And so on.

The result of these **axis parallel splits** is to partition 2d space into 5 regions. We can now associate a mean response with each of these regions, resulting in a piecewise constant surface.

Classification and Regression Trees (CART)



A simple regression tree on two inputs

Classification and Regression Trees (CART)

We can write the model in the following form

$$f(\mathbf{x}) = \mathbb{E}\left[y|\mathbf{x}\right] = \sum_{m=1}^{M} w_m \mathbb{I}(\mathbf{x} \in R_m) = \sum_{m=1}^{M} w_m \phi(\mathbf{x}; \mathbf{v}_m)$$

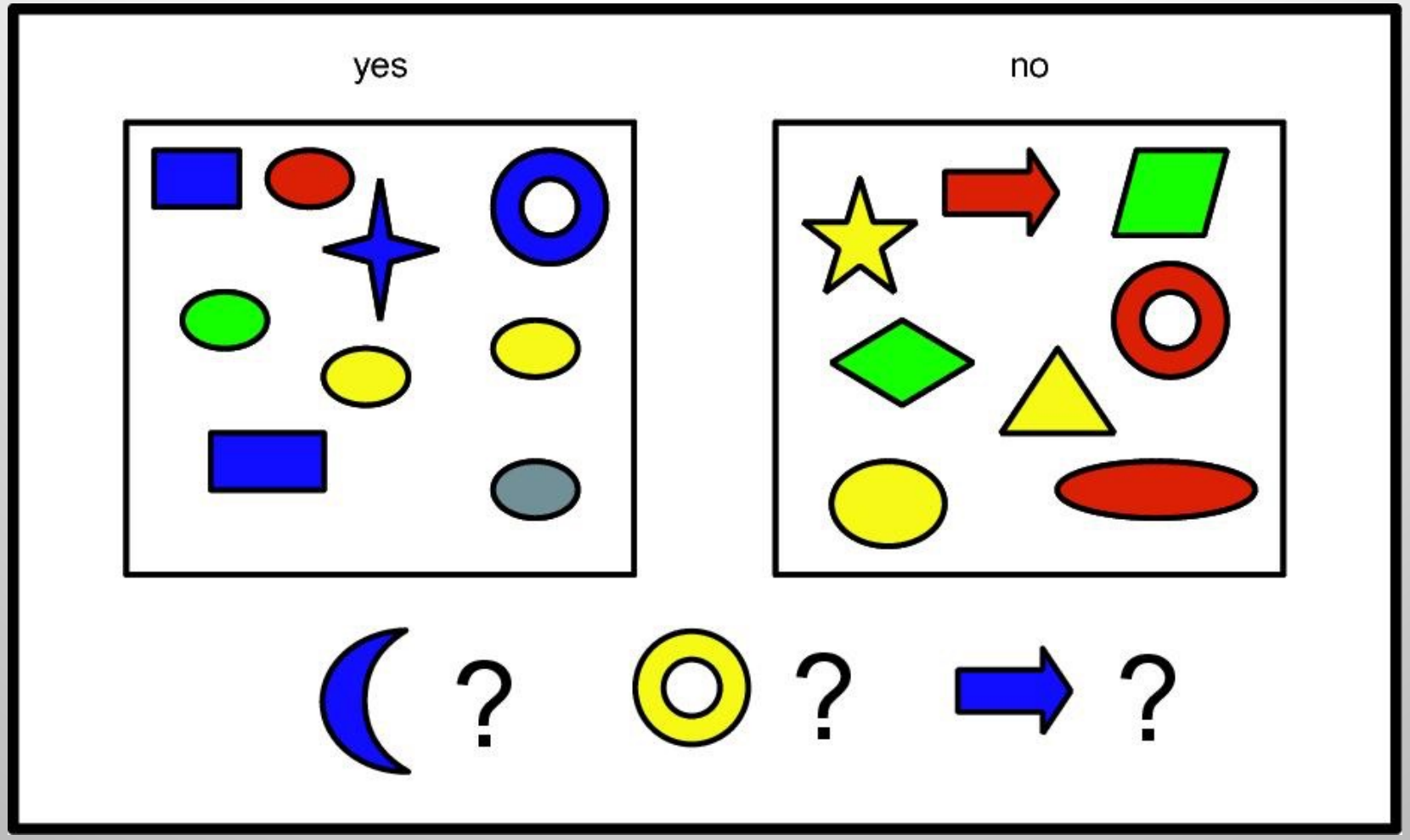
where R_m is the m^{th} region, w_m is the mean response in this region, and $\mathbf{v}_m$ encodes the choice of variable to split on, and the threshold value, on the path from the root to the m^{th} leaf.

Classification and Regression Trees (CART)

It is clear that a CART model is just a an adaptive basis-function model, where the basis functions define the regions, and the weights specify the response value in each region.

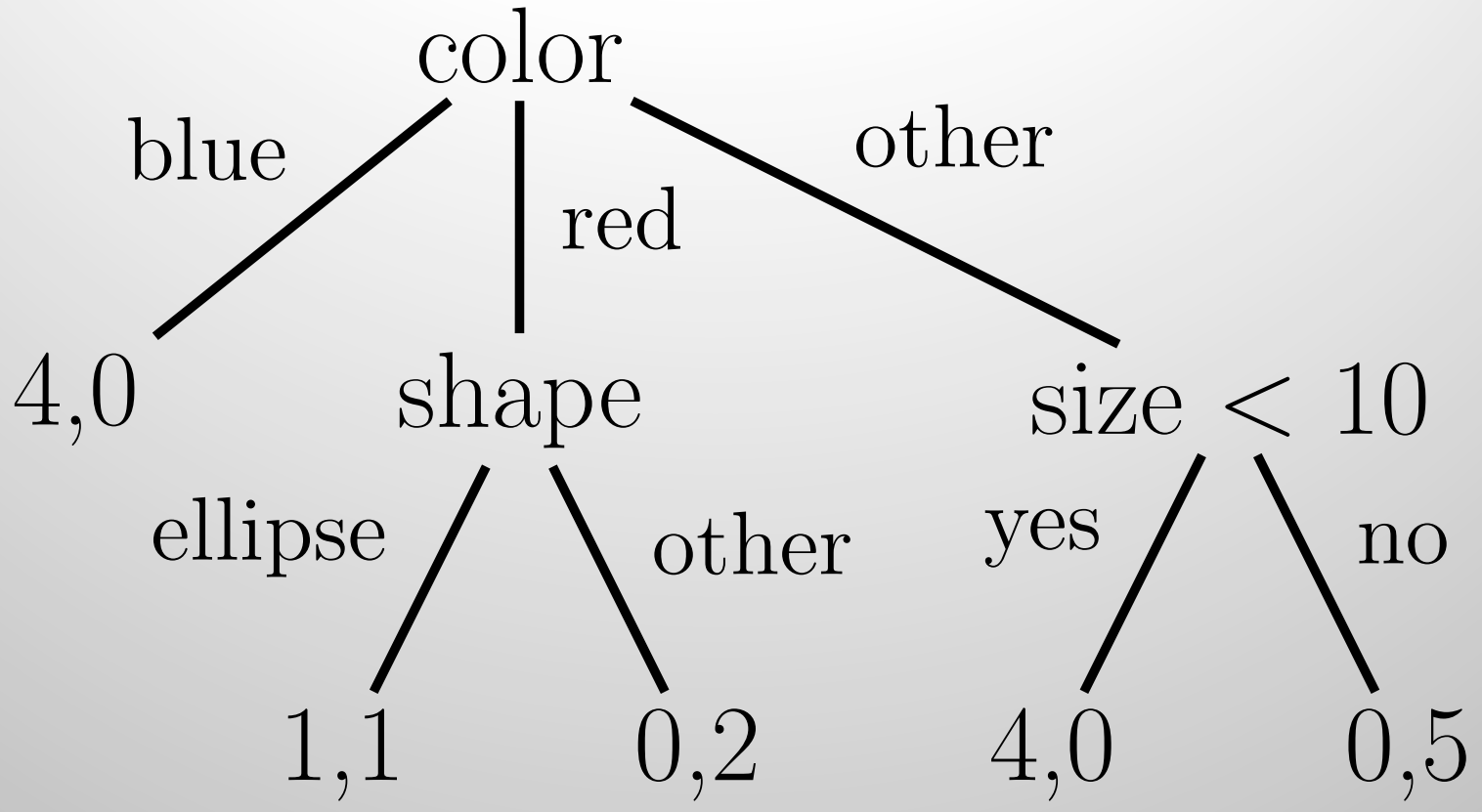
We can generalize this to the classification setting by storing the distribution over class labels in each leaf, instead of the mean response.

Classification and Regression Trees (CART)



Some labeled training examples of colored shapes, along with 3 unlabeled test cases.

Classification and Regression Trees (CART)



A simple decision tree for the data in the previous slide

Classification and Regression Trees (CART)

We first check the color of the object. If it is blue, we follow the left branch and end up in a leaf labeled “4,0”, which means we have 4 positive examples and 0 negative examples which match this criterion. Hence we predict $p(y = 1|x) = 4/4$ if **x** is blue.

Classification and Regression Trees (CART)

If it is red, we then check the shape: if it is an ellipse, we end up in a leaf labeled “1,1”, so we predict $p(y = 1|x) = 1/2$.

If it is red but not an ellipse, we predict $p(y = 1|x) = 0/2$.

Classification and Regression Trees (CART)

If it is some other color, we check the size: if less than 10, we predict $p(y = 1|x) = 4/4$, otherwise $p(y = 1|x) = 0/5$.

These probabilities are just the empirical fraction of positive examples that satisfy each conjunction of feature values, which defines a path from the root to a leaf.

Growing a Tree

Finding the optimal partitioning of the data is NP-complete, so it is common to use the greedy procedure.

The split function chooses the best feature, and the best value for that feature as follows:

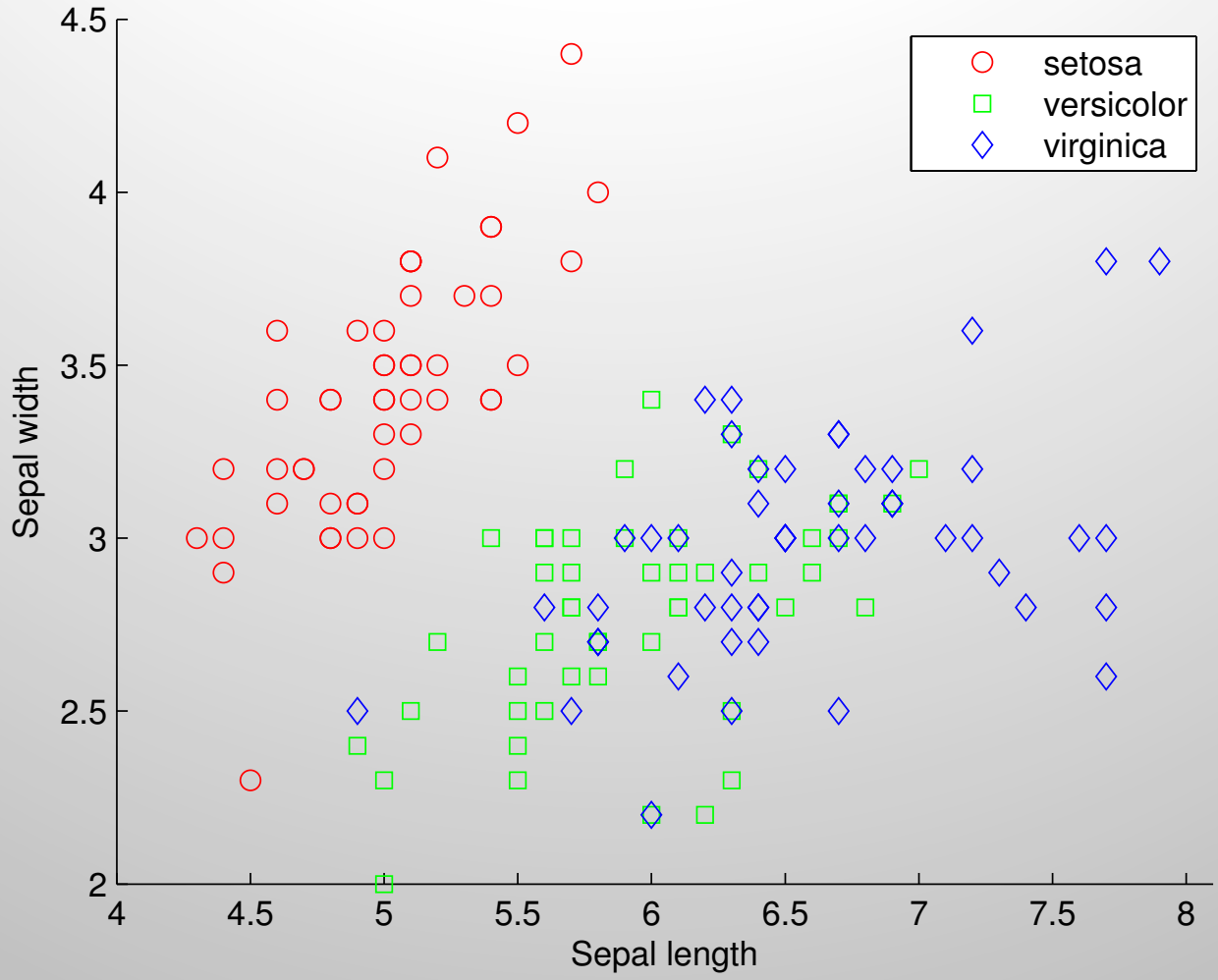
$$(j^*, t^*) = \arg \min_{j \in \{1,\ldots,D\}} \min_{t \in \mathcal{T}_j} \text{cost}(\{\mathbf{x}_i, y_i : x_{ij} \leq t\}) + \text{cost}(\{\mathbf{x}_i, y_i : x_{ij} > t\})$$

Growing a Tree

The function that checks if a node is worth splitting can use several stopping heuristics:

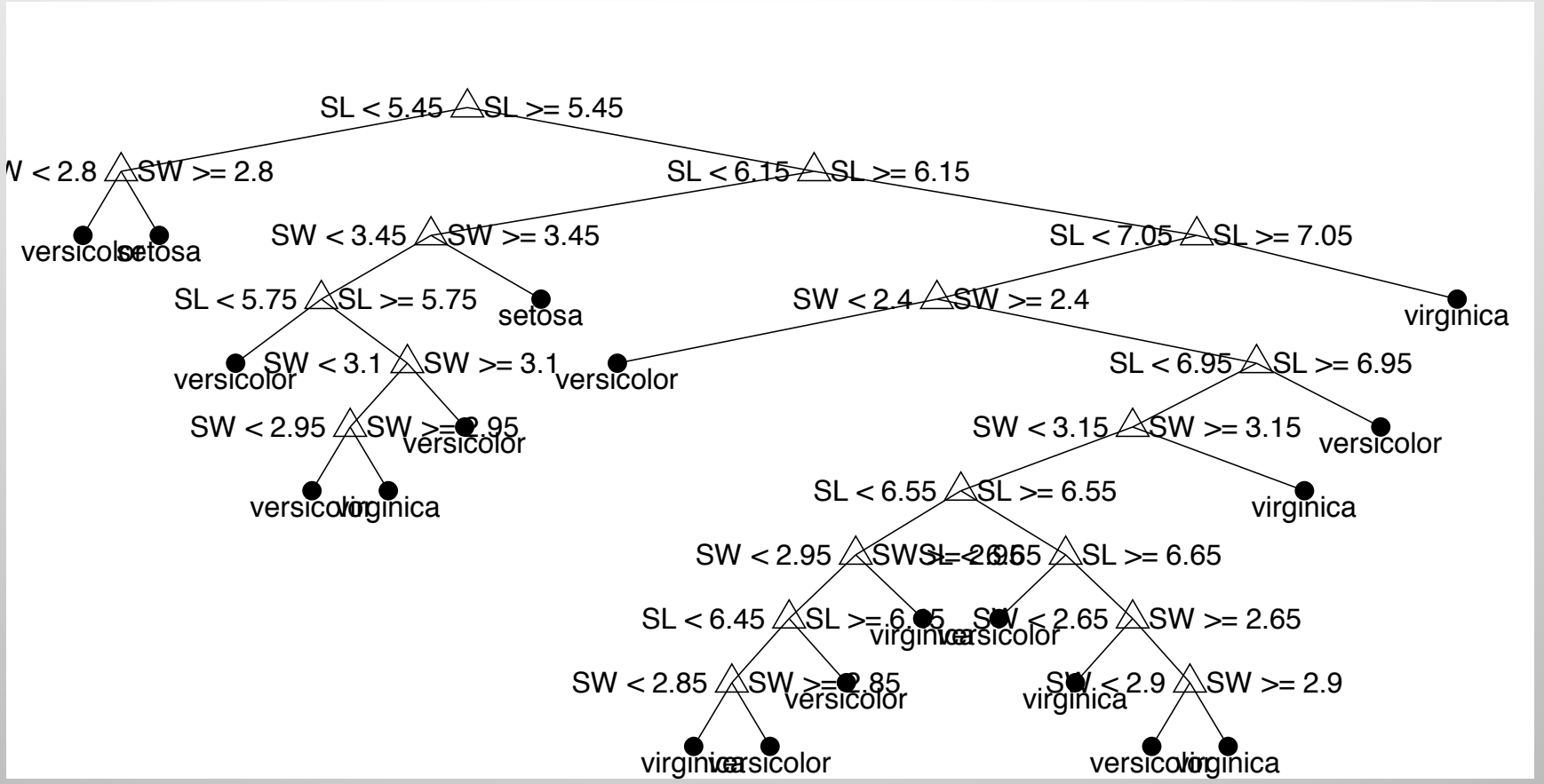
- is the reduction in cost too small?
- has the tree exceeded the maximum desired depth?
- is the distribution of the response in either $\mathcal{D}_L$ or $\mathcal{D}_R$ sufficiently homogeneous?
- is the number of examples in either $\mathcal{D}_L$ or $\mathcal{D}_R$ too small?

Iris Flower Dataset



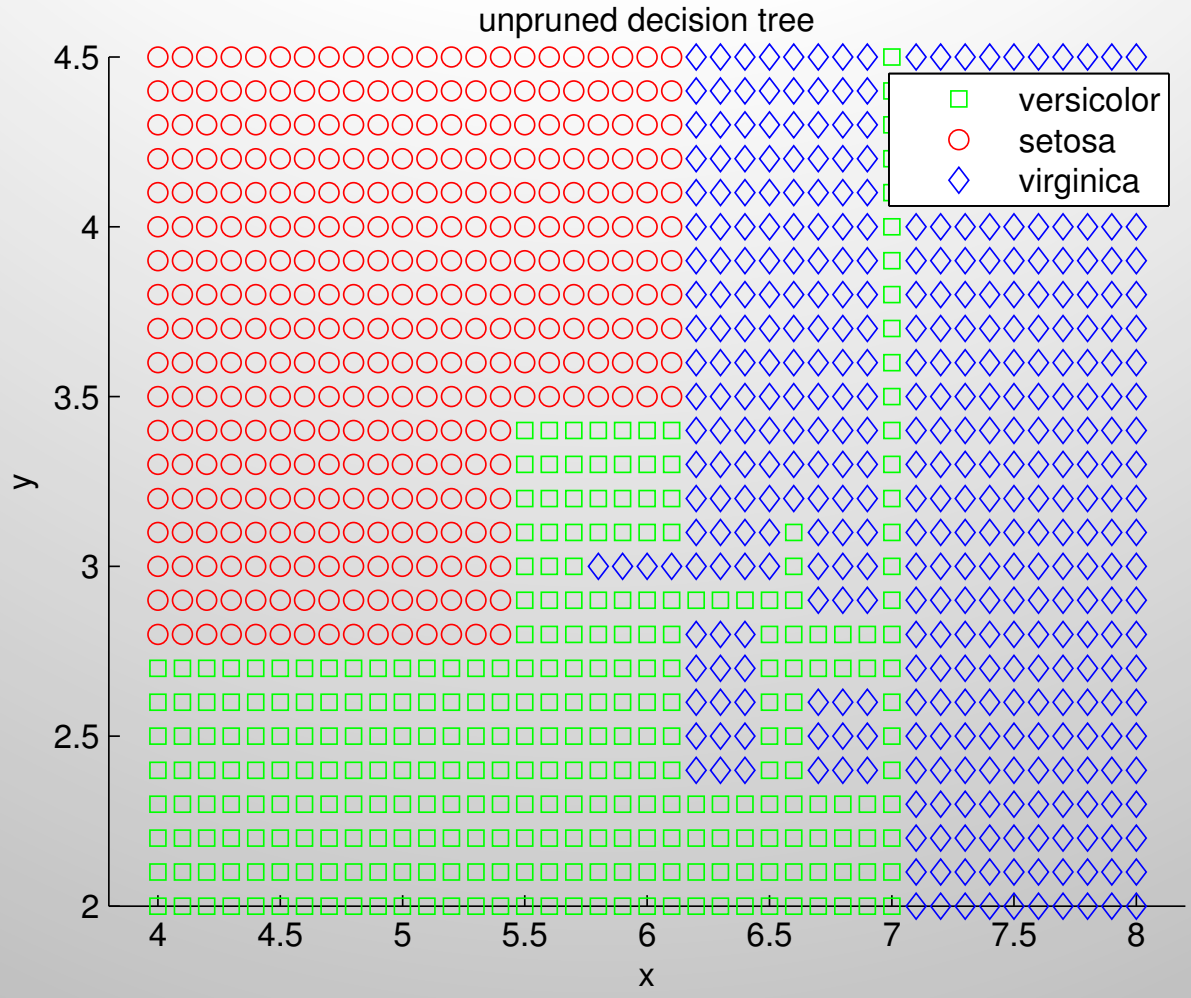
Iris data shown with two features: sepal length and sepal width

Iris Flower Dataset



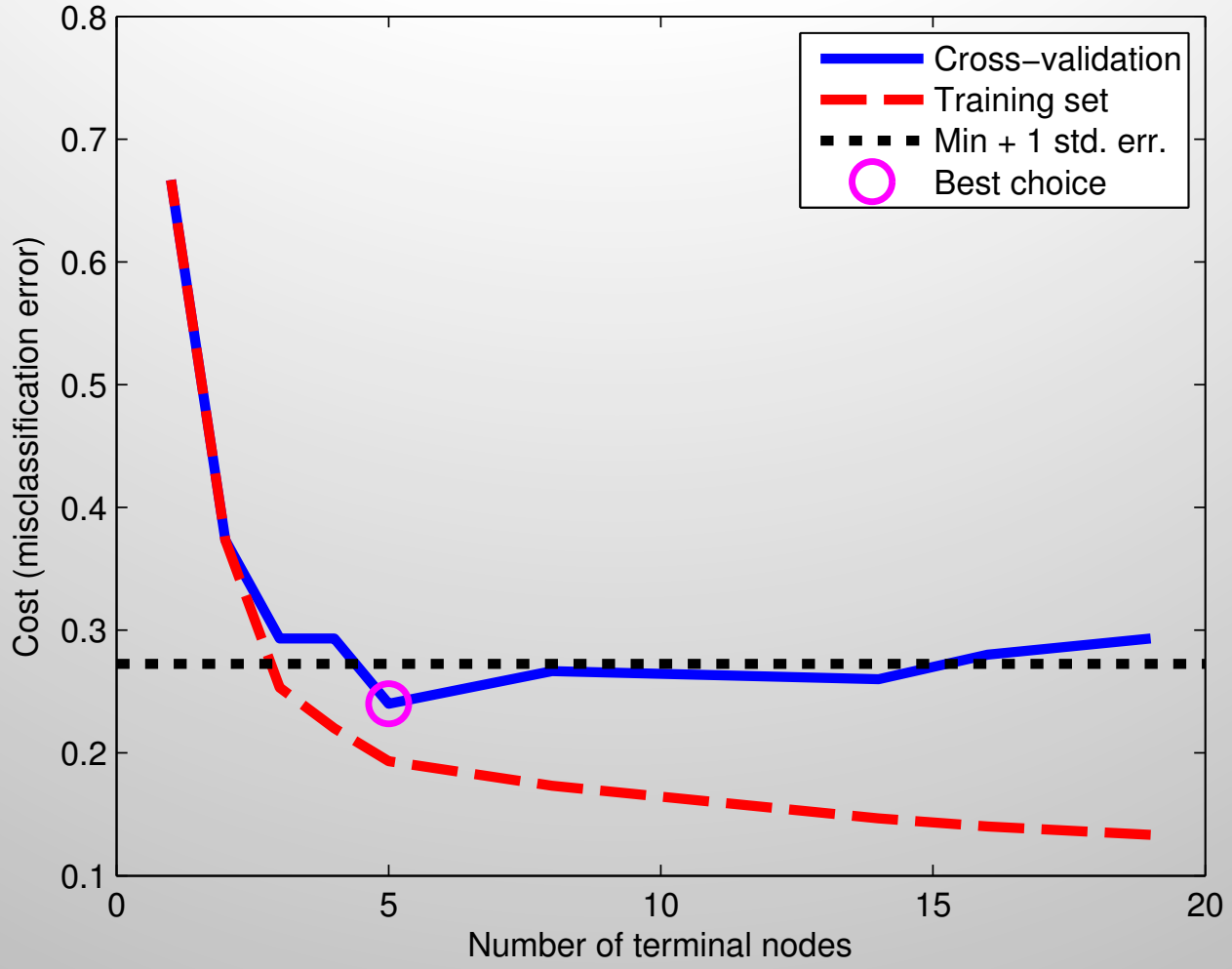
Unpruned decision tree for Iris data

Iris Flower Dataset



Decision boundaries induced by the decision tree on the previous slide

Iris Flower Dataset



Plot of misclassification error rate vs depth of tree