ColumbiaX: Machine Learning

Lecture 11

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

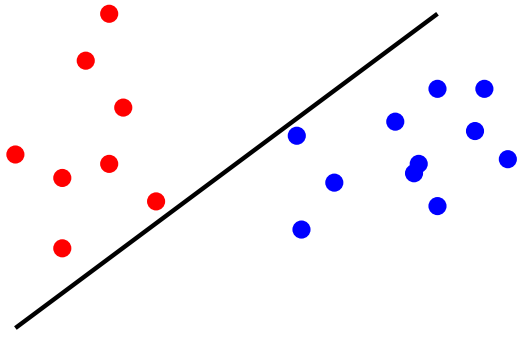
Columbia University

Maximum Margin Classifiers

Maximum margin idea

Setting

Linear classification, two linearly separable classes.



Recall Perceptron

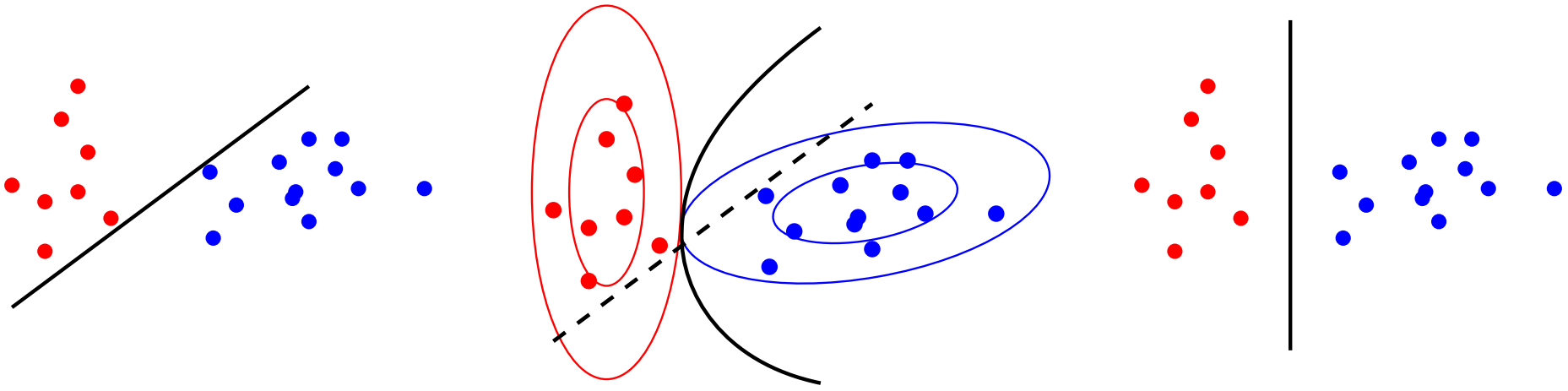
- Selects *some* hyperplane separating the classes.
- Selected hyperplane depends on several factors.

Maximum margin

To achieve good generalization (low prediction error), place the hyperplane "in the middle" between the two classes.

More precisely, choose a plane such that its distance to the closest point in each class is maximized. This distance is called the **margin**.

Generalization Error



Possible Perceptron solution

(dotted) poor generalization, (solid) better

Maximum margin solution

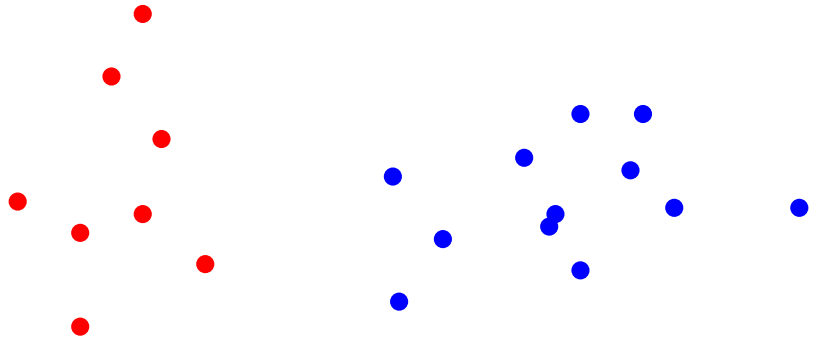
Example: Gaussian data

- Intuitively, the classifier on the left isn't good because sampling more data could lead to misclassifications.
- If we imagine the data from each class as Gaussian, we could frame the goal as to find a decision boundary that cuts into as little probability mass as possible.
- With no distribution assumptions, we can argue that max-margin is best.

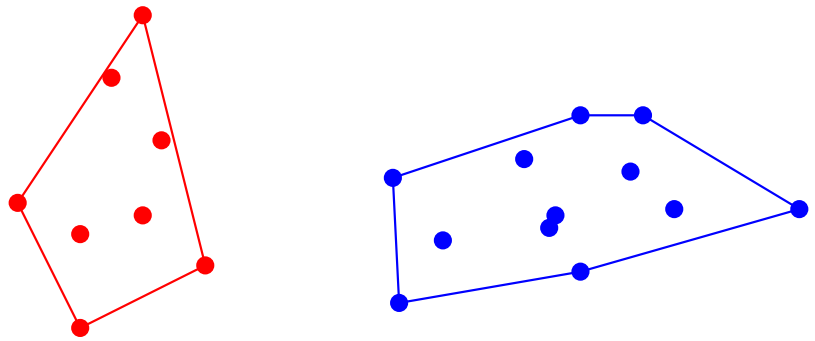
SUBSTITUTING CONVEX SETS

Observation

Where a separating hyperplane may be placed depends on the "outer" points on the sets. Points in the center do not matter.

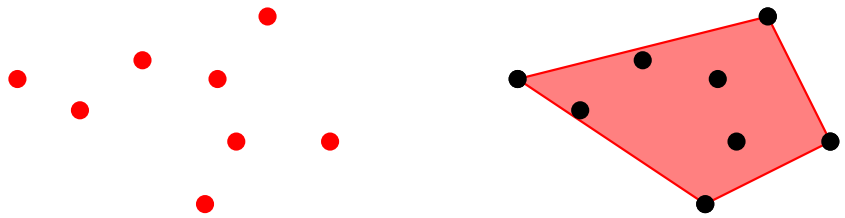


In geometric terms, we can represent each class by the smallest convex set which contains all point in the class. This is called a *convex hull*.



Convex hulls

The thing to remember for this lecture is that a convex hull is defined by all possible weighted averages of points in a set.



That is, let $x_1, \ldots, x_n$ be the above data coordinates. Every point x_0 in the shaded region – i.e., the convex hull – can be reached by setting

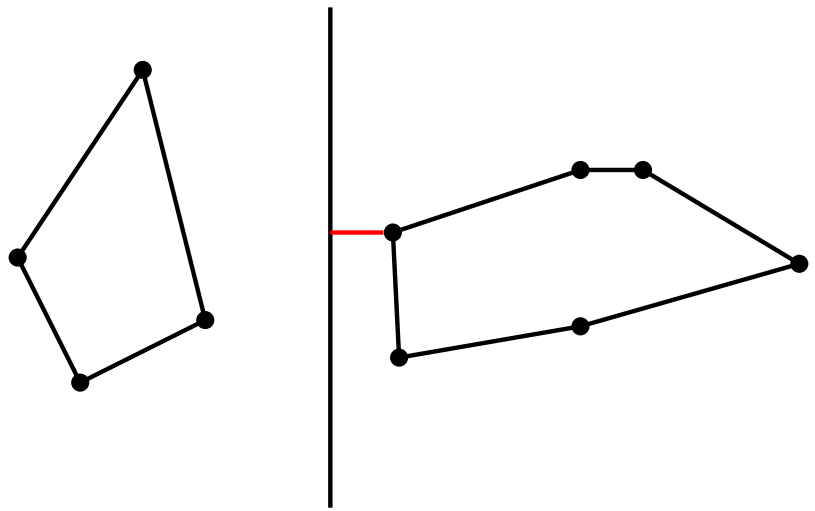
$$x_0 = \sum_{i=1}^{n} \alpha_i x_i, \quad \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i = 1,$$

for some $(\alpha_1, \ldots, \alpha_n)$. No point outside this region can be reached this way.

Margin

Definition

The *margin* of a classifying hyperplane H is the shortest distance between the plane and any point in either set (equivalently, the convex hull)



When we maximize this margin, H is "exactly in the middle" of the two convex hulls. Of course, the difficult part is how do we find this H?

SUPPORT VECTOR MACHINES

Finding the hyperplane

For n linearly separable points $(x_1, y_1), \ldots, (x_n, y_n)$ with $y_i \in \{\pm 1\}$, solve:

$$\begin{aligned} \min_{w, w_0} \quad & \tfrac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(x_i^T w + w_0) \geq 1 \quad \text{for } i = 1, \ldots, n \end{aligned}$$

Comments

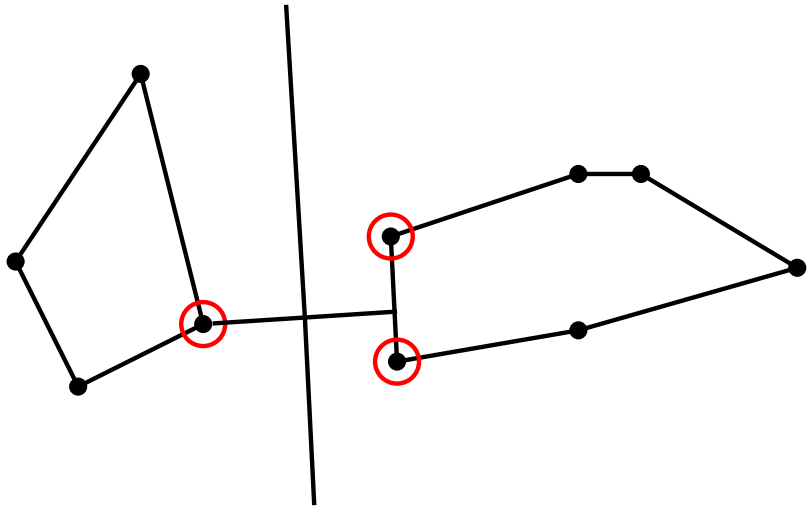
- Recall that $y_i(x_i^T w + w_0) > 0$ if $y_i = \text{sign}(x_i^T w + w_0)$.
- If there exists a hyperplane H that separates the classes, we can scale w so that $y_i(x_i^T w + w_0) > 1$ for all i (this is useful later).
- The resulting classifier is called a *support vector machine*. This formulation only has a solution when the classes are linearly separable.
- It is not at all obvious why this maximizes the margin. This will become more clear when we look at the solution.

SUPPORT VECTOR MACHINE

Skip to the end

Q: First, can we intuitively say what the solution should *look* like?

A: Yes, but we won't give the proof.



1. Find the closest two points from the convex hulls of class $+1$ and -1.

2. Connect them with a line and put a perpendicular hyperplane in the middle.

3. If S_1 and S_0 are the sets of x in class $+1$ and -1 respectively, we're looking for two *probability* vectors α_1 and α_0 such that we minimize

$$\left\| \underbrace{\left(\sum_{x_i \in S_1} \alpha_{1i} x_i\right)}_{\text{in conv. hull of } S_1} - \underbrace{\left(\sum_{x_i \in S_0} \alpha_{0i} x_i\right)}_{\text{in conv. hull of } S_0} \right\|_2$$

4. Then we define the hyperplane using the two points found with α_1 and α_0.

Primal problem

The *primal* optimization problem is the one we defined:

$$\begin{aligned} \min_{w,w_0} \quad & \tfrac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(x_i^T w + w_0) \geq 1 \quad \text{for } i = 1, \ldots, n \end{aligned}$$

This is tricky, so we use *Lagrange multipliers* to set up the "dual" problem.

Lagrange multipliers

Define Lagrange multipliers $\alpha_i > 0$ for $i = 1, \ldots, n$. The Lagrangian is

$$\begin{aligned} \mathcal{L} &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(x_i^T w + w_0) - 1) \\ &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i (x_i^T w + w_0) + \sum_{i=1}^{n} \alpha_i \end{aligned}$$

We want to minimize $\mathcal{L}$ over w and w_0 and maximize over $(\alpha_1, \ldots, \alpha_n)$.

SETTING UP THE DUAL PROBLEM

First minimize over w and w_0:

$$\mathcal{L} = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i y_i (x_i^T w + w_0) + \sum_{i=1}^{n} \alpha_i$$

$$\Downarrow$$

$$\nabla_w \mathcal{L} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \qquad \Rightarrow \qquad w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial \mathcal{L}}{\partial w_0} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \Rightarrow \qquad \sum_{i=1}^{n} \alpha_i y_i = 0$$

Therefore,

1. We can plug the solution for w back into the problem.
2. We know that $(\alpha_1, \ldots, \alpha_n)$ must satisfy $\sum_{i=1}^{n} \alpha_i y_i = 0$.

SVM DUAL PROBLEM

Lagrangian: $\mathcal{L} = \frac{1}{2}\|w\|^2 - \sum_{i=1}^n \alpha_i y_i (x_i^T w + w_0) + \sum_{i=1}^n \alpha_i$

Dual problem

Plugging these values in from the previous slide, we get the dual problem

$$\max_{\alpha_1,\ldots,\alpha_n} \quad \mathcal{L} = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

$$\text{subject to} \quad \sum_{i=1}^n \alpha_i y_i = 0, \qquad \alpha_i \geq 0 \quad \text{for } i = 1, \ldots, n$$

Comments

- Where did w_0 go? The condition $\sum_{i=1}^n \alpha_i y_i = 0$ gives $0 \cdot w_0$ in the dual.
- We now maximize over the α_i. This requires an algorithm that we won't discuss in class. Many good software implementations are available.

Solving the primal problem

Before discussing the solution of the dual, we ask:

After finding each α_i how do we predict a new $y_0 = sign(x_0^T w + w_0)$?

We have: $\mathcal{L} = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i (x_i^T w + w_0) - 1)$

With conditions: $\alpha_i \geq 0, \qquad y_i (x_i^T w + w_0) - 1 \geq 0$

Solve for w.

$$\nabla_w \mathcal{L} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad \text{(just plug in the learned } \alpha_i\text{'s)}$$

What about w_0?

- We can show that at the solution, $\alpha_i (y_i (x_i^T w + w_0) - 1) = 0$ for all i.
- Therefore, pick i for which $\alpha_i > 0$ and solve $y_i (x_i^T w + w_0) - 1 = 0$ for w_0 using the solution for w (all possible i will give the same solution).

Dual problem

We can manipulate the dual problem to find out what it's trying to do.

$$\max_{\alpha_1,\ldots,\alpha_n} \quad \mathcal{L} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \qquad \alpha_i \geq 0 \quad \text{for } i = 1, \ldots, n$$

Since $y_i \in \{-1, +1\}$

- $\sum_{i=1}^{n} \alpha_i y_i = 0 \;\Rightarrow\; C = \sum_{i \in S_1} \alpha_i = \sum_{j \in S_0} \alpha_j$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j) = \left\| \sum_{i=1}^{n} \alpha_i y_i x_i \right\|^2 = C^2 \left\| \sum_{i \in S_1} \frac{\alpha_i}{C} x_i - \sum_{j \in S_0} \frac{\alpha_j}{C} x_j \right\|^2$

Dual problem

We can change notation to write the dual as

$$\begin{aligned} \max_{\alpha_1,\ldots,\alpha_n} \quad & \mathcal{L} = 2C - \frac{1}{2}C^2 \left\| \sum_{i\in S_1} \frac{\alpha_i}{C} x_i - \sum_{j\in S_0} \frac{\alpha_j}{C} x_j \right\|^2 \\ \text{subject to} \quad & C := \sum_{i\in S_1} \alpha_i = \sum_{j\in S_0} \alpha_j, \quad \alpha_i \geq 0 \end{aligned}$$

We observe that the maximum of this function satisfies

$$\min_{\alpha_1,\ldots,\alpha_n} \left\| \underbrace{\left(\textstyle\sum_{i\in S_1} \frac{\alpha_i}{C} x_i \right)}_{\text{in conv. hull of } S_1} - \underbrace{\left(\textstyle\sum_{j\in S_0} \frac{\alpha_j}{C} x_j \right)}_{\text{in conv. hull of } S_0} \right\|^2$$

Therefore, the dual problem is trying to find the closest points in the convex hulls constructed from data in class $+1$ and -1.

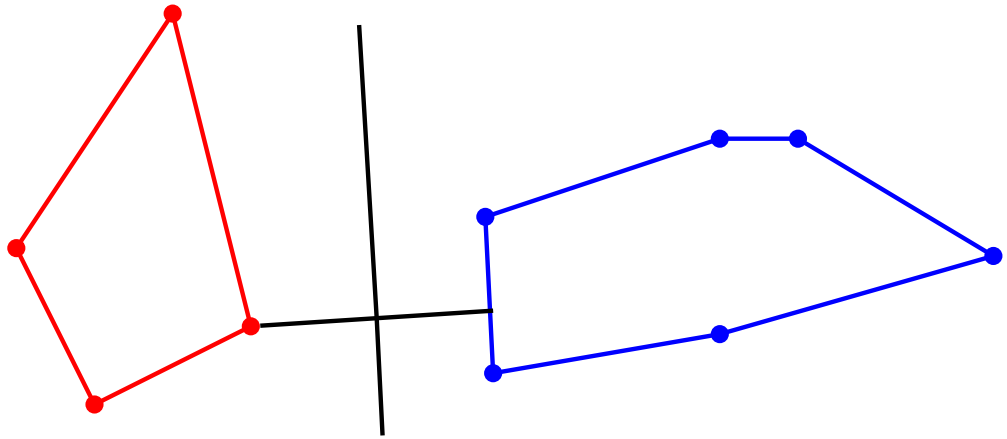
RETURNING TO THE PICTURE

Recall

We wanted to find:

$$\min_{\substack{u \in \mathcal{H}(S_1) \\ v \in \mathcal{H}(S_0)}} \|u - v\|^2$$

The direction of w is $u - v$.



We previously claimed we can find the max-margin hyperplane as follows:

1. Find shortest line connecting the convex hulls.
2. Place hyperplane orthogonal to line and exactly at the midpoint.

With the SVM we want to minimize $\|w\|^2$ and we can write this solution as

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i = C \left(\sum_{i \in S_1} \frac{\alpha_i}{C} x_i - \sum_{j \in S_0} \frac{\alpha_j}{C} x_j \right)$$

SOFT-MARGIN SVM

Question: What if the data isn't linearly separable?

Answer: Permit training data be on wrong side of hyperplane, but at a cost.

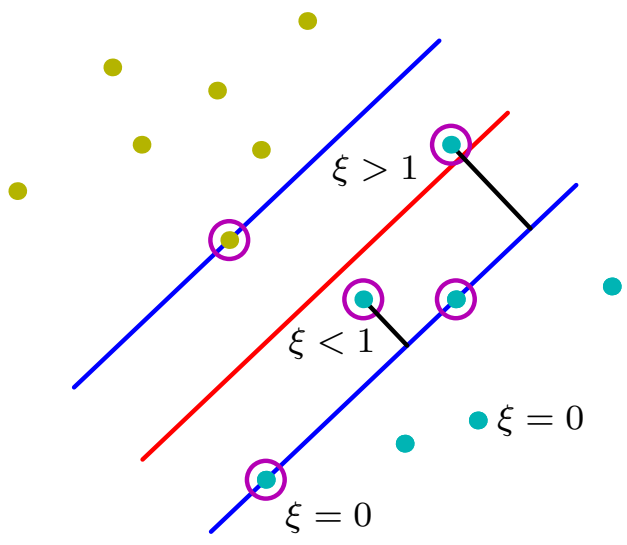
Slack variables

Replace the training rule $y_i(x_i^T w + w_0) \geq 1$ with

$$y_i(x_i^T w + w_0) \geq 1 - \xi_i,$$

with $\xi_i \geq 0$.

The ξ_i are called *slack variables*.



Soft-margin objective function

Adding the slack variables gives a new objective to optimize

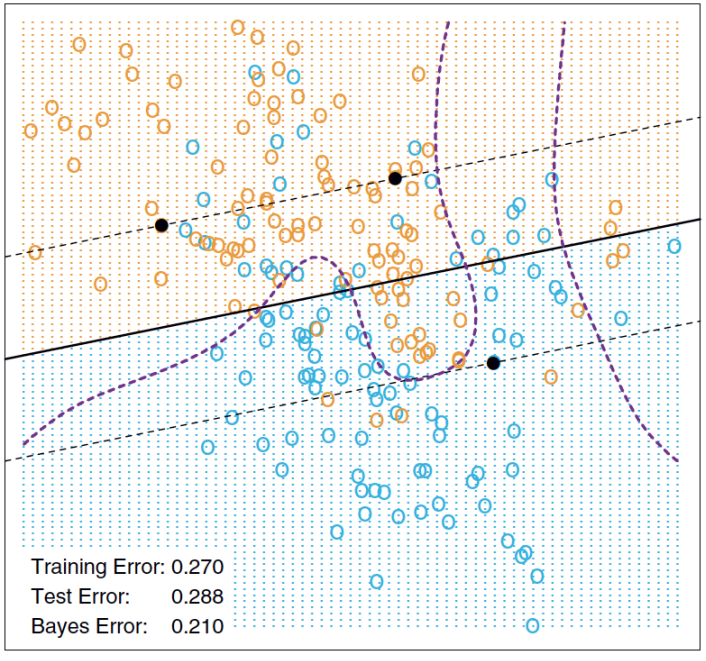
$$\begin{aligned} \min_{w, w_0, \xi_1, \ldots, \xi_n} \quad & \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(x_i^T w + w_0) \geq 1 - \xi_i \quad \text{for} \quad i = 1, \ldots, n \\ & \xi_i \geq 0 \quad \text{for} \quad i = 1, \ldots, n \end{aligned}$$

We also have to choose the parameter $\lambda > 0$. We solve the dual as before.

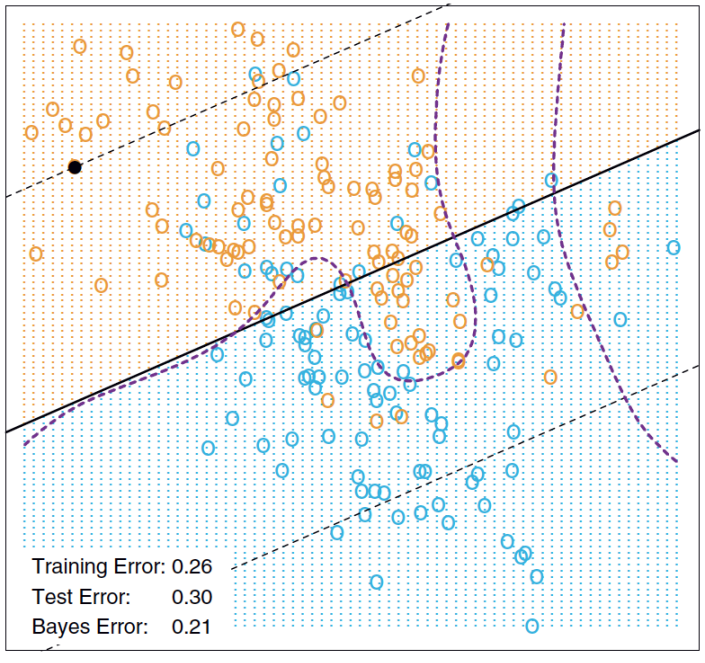
Role of λ

- Specifies the "cost" of allowing a point on the wrong side.
- If λ is very small, we're happy to misclassify.
- For $\lambda \to \infty$, we recover the original SVM because we want $\xi_i = 0$.
- We can use cross-validation to choose it.

Influence of Margin Parameter



$\lambda = 100000$



$\lambda = 0.01$

Hyperplane is sensitive to λ. Either way, a linear classifier isn't ideal . . .

Primal problem with slack variables

Let's map the data into higher dimensions using the function $\phi(x_i)$,

$$\begin{aligned} \min_{w,w_0,\xi_1,\ldots,\xi_n} \quad & \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(\phi(x_i)^T w + w_0) \geq 1 - \xi_i \quad \text{for} \quad i = 1, \ldots, n \\ & \xi_i \geq 0 \quad \text{for} \quad i = 1, \ldots, n \end{aligned}$$

Dual problem

Maximize over each (α_i, μ_i) and minimize over $w, w_0, \xi_1, \ldots, \xi_n$

$$\mathcal{L} = \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i (y_i(\phi(x_i)^T w + w_0) - 1 + \xi_i) - \sum_{i=1}^{n} \mu_i \xi_i$$

subject to $\alpha_i \geq 0, \quad \mu_i \geq 0, \quad y_i(\phi(x_i)^T w + w_0) - 1 + \xi_i \geq 0$

KERNELIZING THE SVM

Dual problem

Minimizing for w, w_0 and each ξ_i, we find

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i, \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \lambda - \alpha_i - \mu_i = 0$$

If we plug w and $\mu_i = \lambda - \alpha_i$ back into the $\mathcal{L}$, we have the dual problem

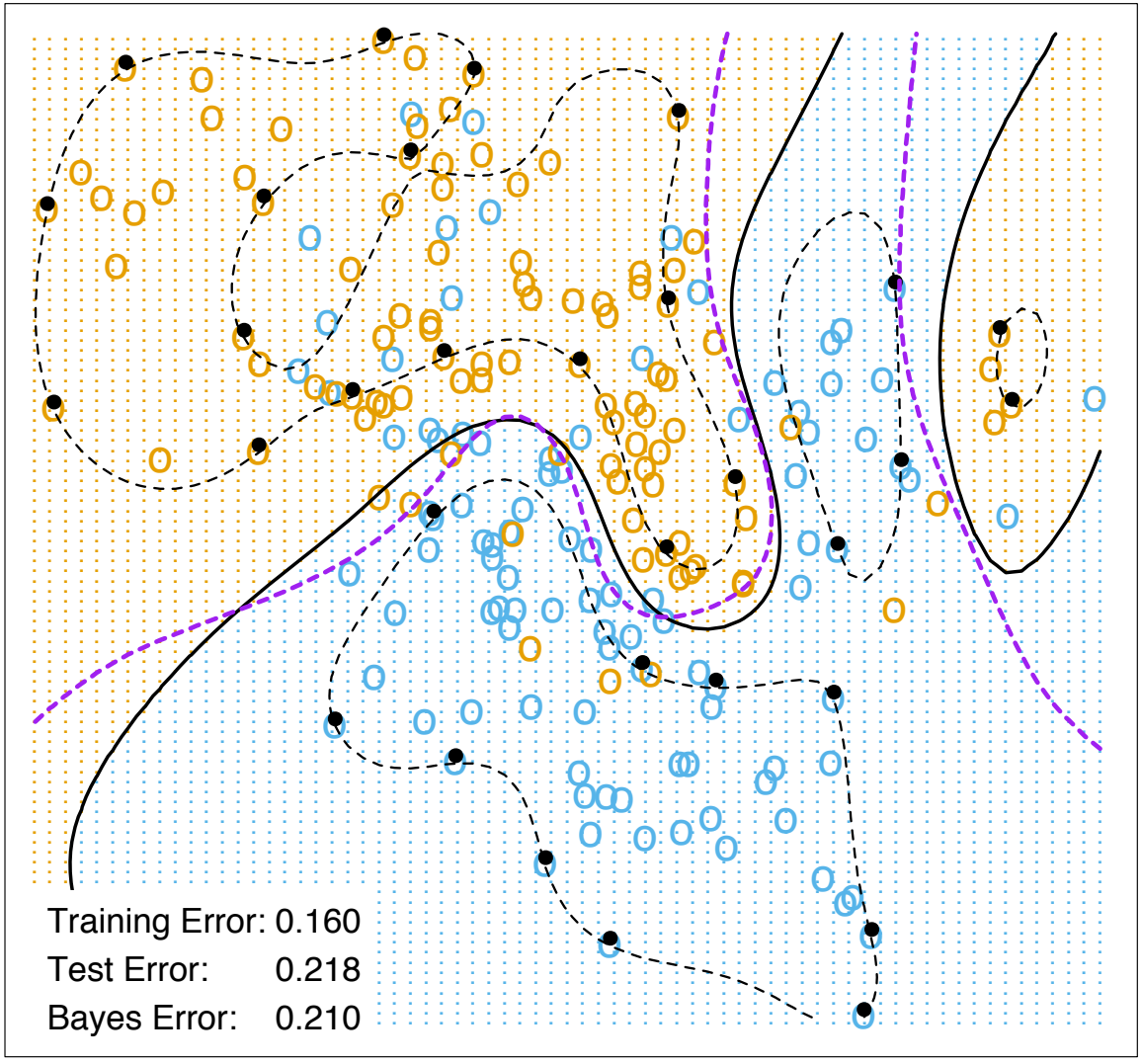
$$\max_{\alpha_1,\ldots,\alpha_n} \quad \mathcal{L} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \underbrace{\phi(x_i)^T \phi(x_j)}_{\boldsymbol{K(x_i, x_j)}}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq \lambda$$

Classification: Using the solution $w = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i)$, declare

$$y_0 = \text{sign}\Big(\sum_{i=1}^{n} \alpha_i y_i \phi(x_0)^T \phi(x_i) + w_0 \Big) = \text{sign}\Big(\sum_{i=1}^{n} \alpha_i y_i K(x_0, x_i) + w_0 \Big)$$

KERNELIZING THE SVM



Black solid line

SVM decision boundary

Classification rule

$$\text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_0, x_i) + w_0\right)$$

Dots

Support vectors ($\alpha_i > 0$)

Purple line

A Bayes classifier.

Summary: Support Vector Machine

Basic SVM

- Linear classifier for linearly separable data.
- Position of affine hyperplane is determined to maximize the margin.
- The dual is a convex, so we can find exact solution with optimization.

Full-fledged SVM

Ingredient	Purpose
Maximum margin	Good generalization properties
Slack variables	Overlapping classes, robust against outliers
Kernel	Nonlinear decision boundary

Use in practice

- Software packages (many options)
- Choose a kernel function (e.g., RBF)
- Cross-validate λ parameter and RBF kernel width

ColumbiaX: Machine Learning

Lecture 12

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

Decision Trees

Decision Trees

A *decision tree* maps input $x \in \mathbb{R}^d$ to output y using binary decision rules:

- Each node in the tree has a *splitting rule*.
- Each leaf node is associated with an output value (outputs can repeat).

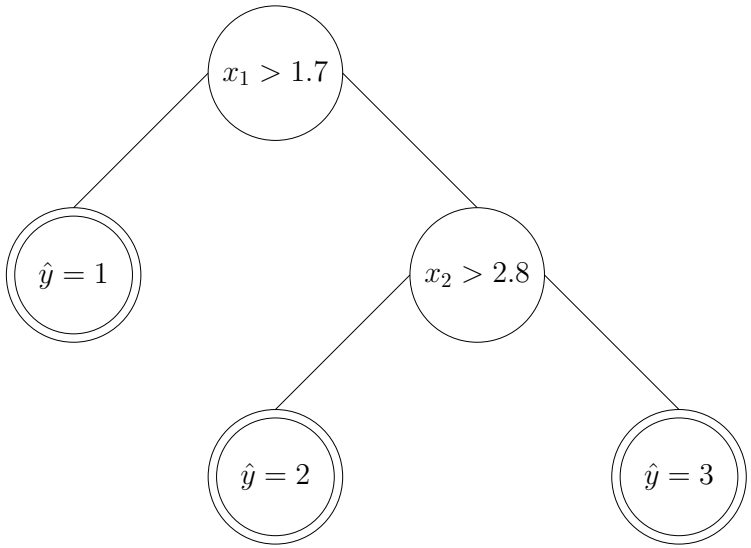
Each splitting rule is of the form

$$h(x) = \mathbb{1}\{x_j > t\}$$

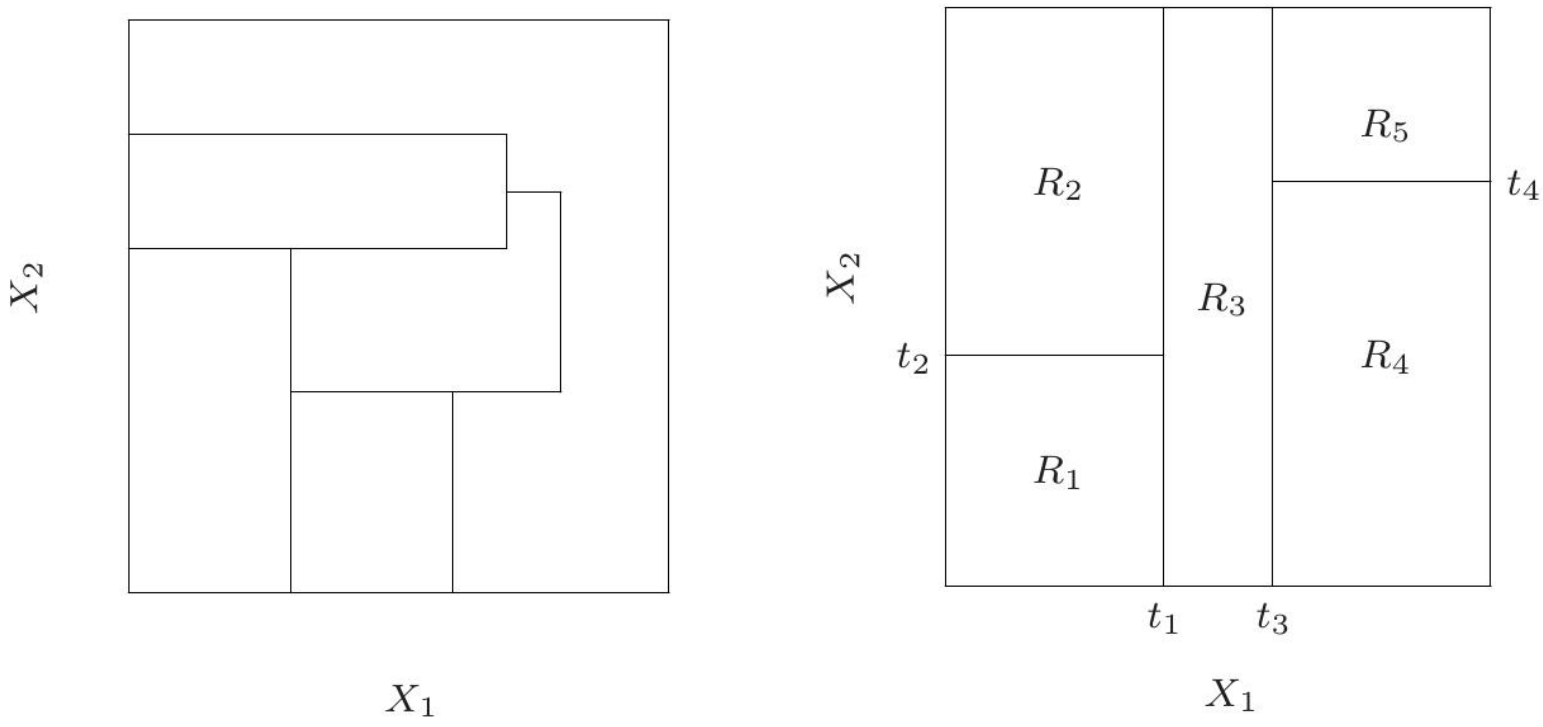
for some dimension j of x and $t \in \mathbb{R}$.

Using these transition rules, a path to a *leaf node* gives the prediction.

(One-level tree = *decision stump*)



Regression Trees

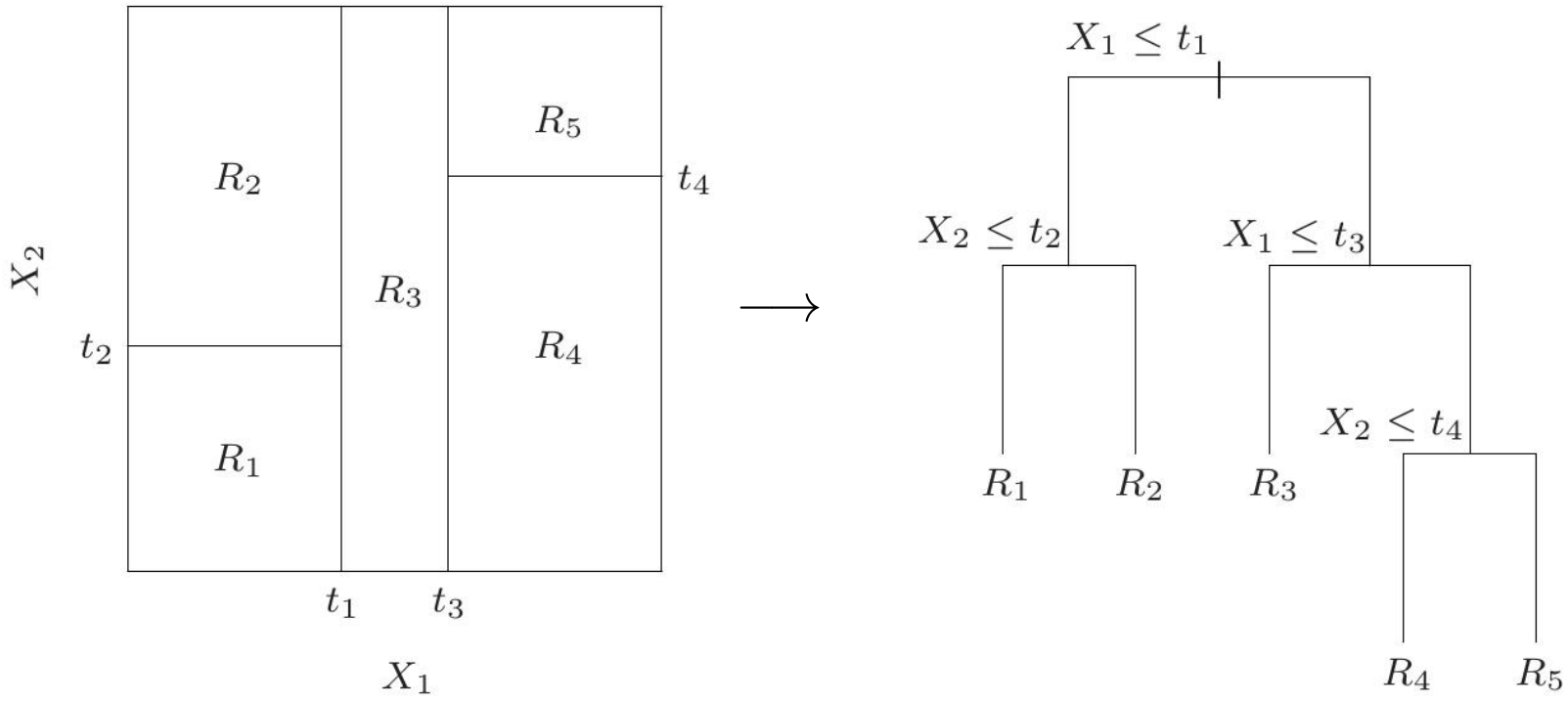


Motivation: Partition the space so that data in a region have same prediction

Left: Difficult to define a "rule".

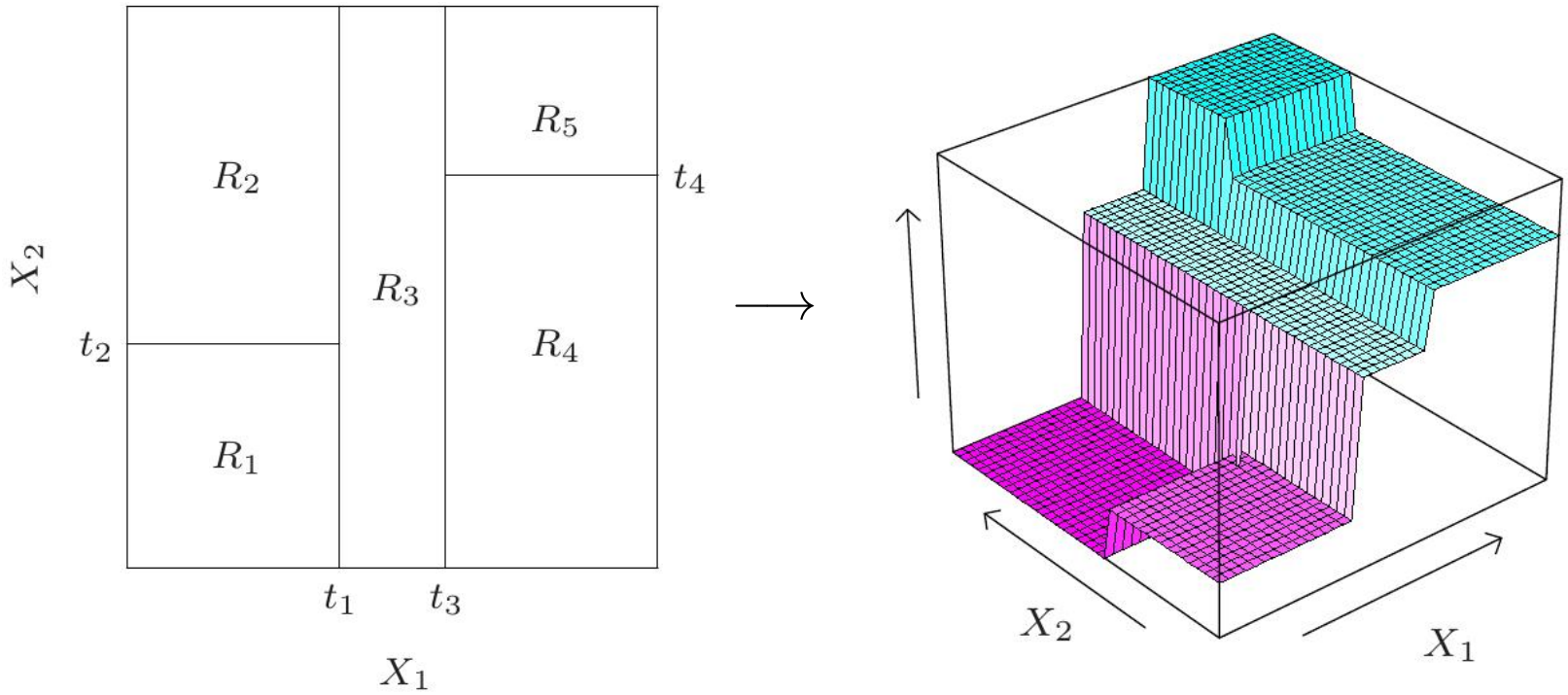
Right: Easy to define a recursive splitting rule.

Regression Trees



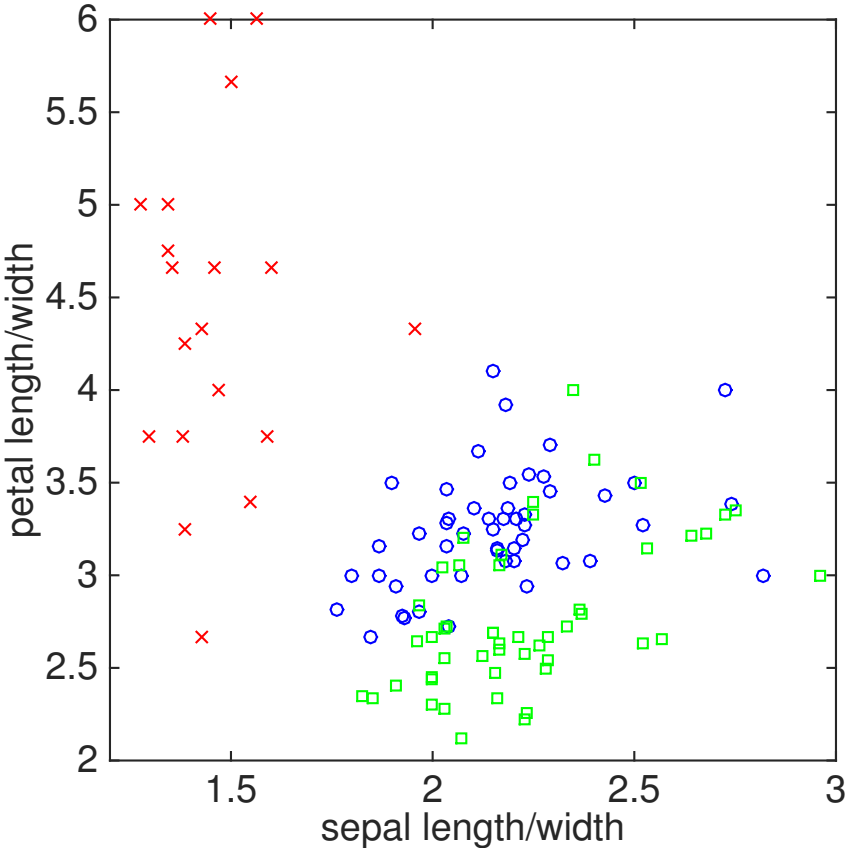
If we think in terms of trees, we can define a simple rule for partitioning the space. The left and right figures represent the same regression function.

Regression Trees



Adding an output dimension to the figure (right), we can see how regression trees can learn a step function approximation to the data.

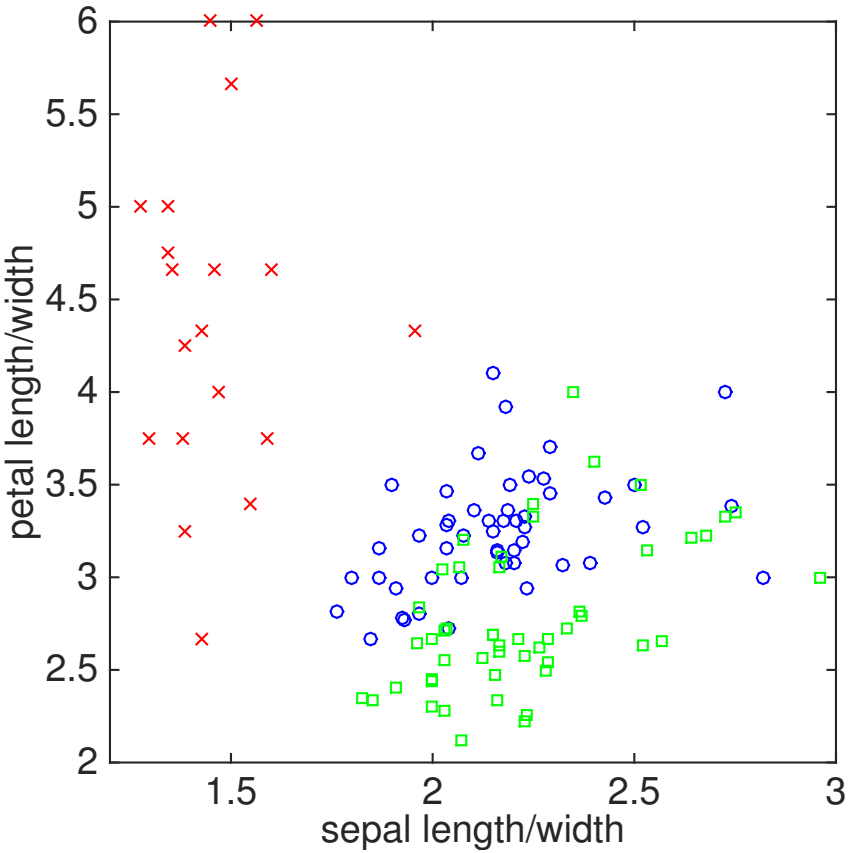
Classification Trees (Example)



Classifying irises using sepal and petal measurements:

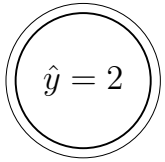
- $x \in \mathbb{R}^2, y \in \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width

Classification Trees (Example)

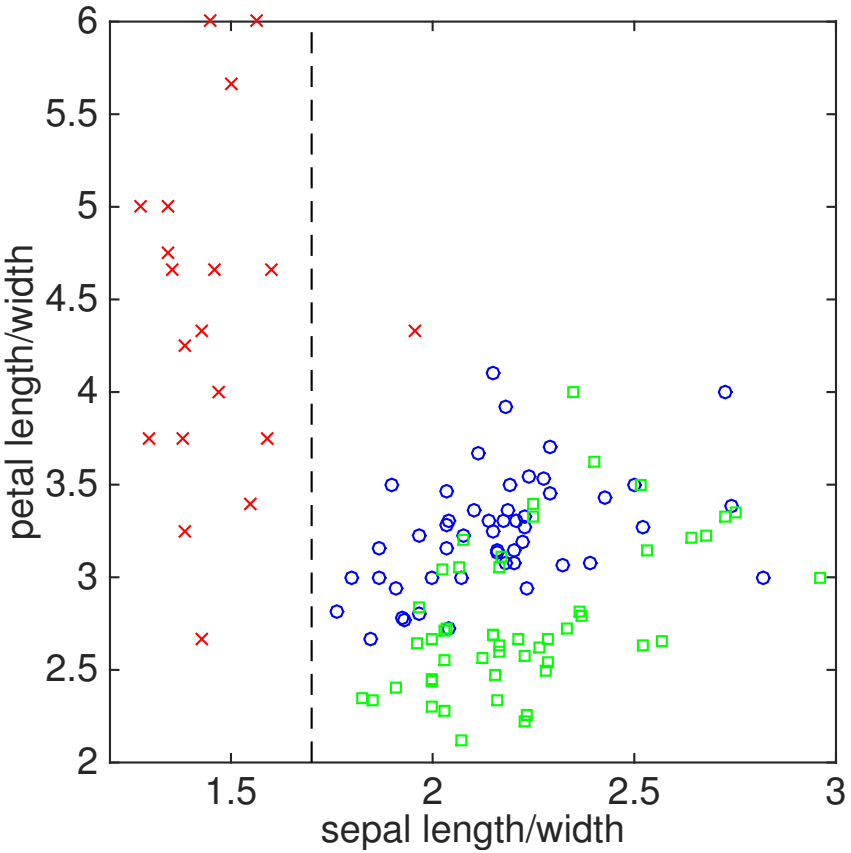


Classifying irises using sepal and petal measurements:

- $x \in \mathbb{R}^2, y \in \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width

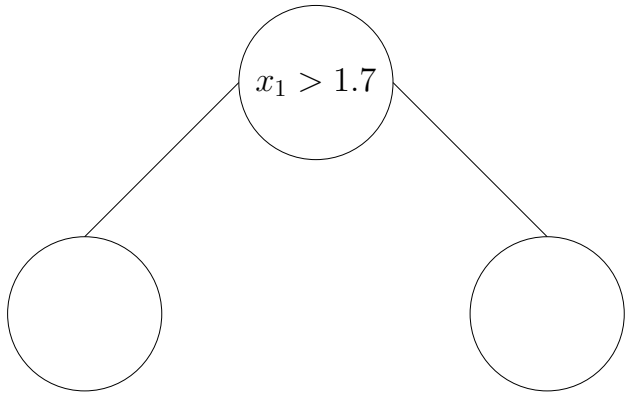


CLASSIFICATION TREES (EXAMPLE)

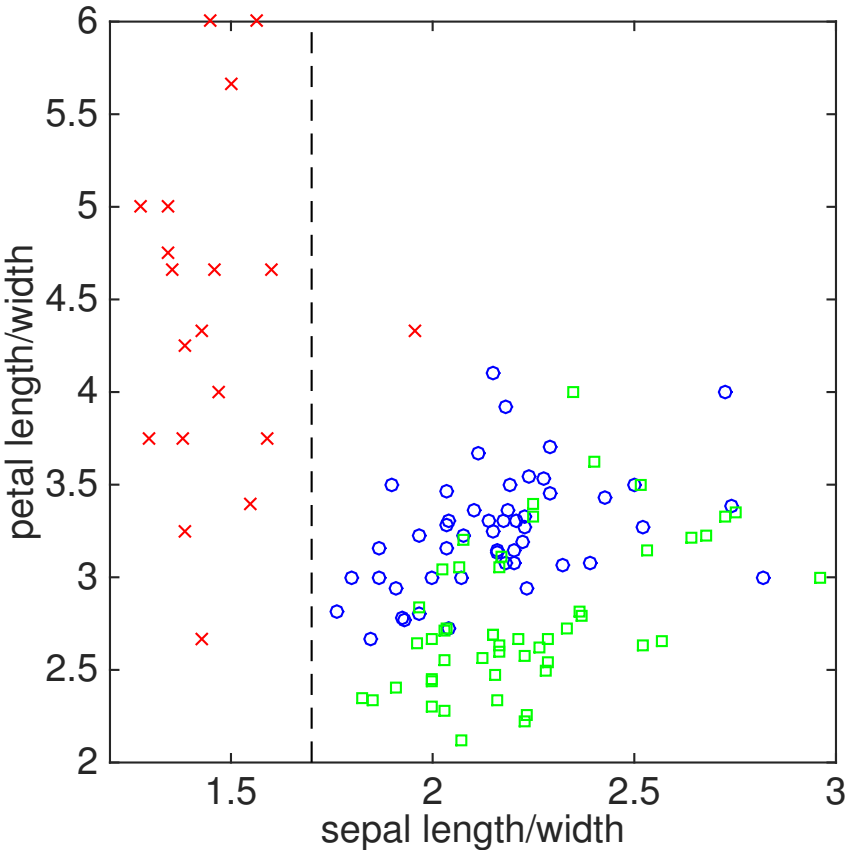


Classifying irises using sepal and petal measurements:

- $x \in \mathbb{R}^2, y \in \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width

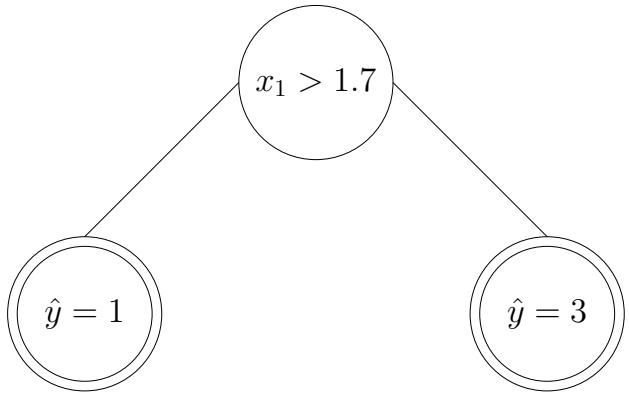


CLASSIFICATION TREES (EXAMPLE)

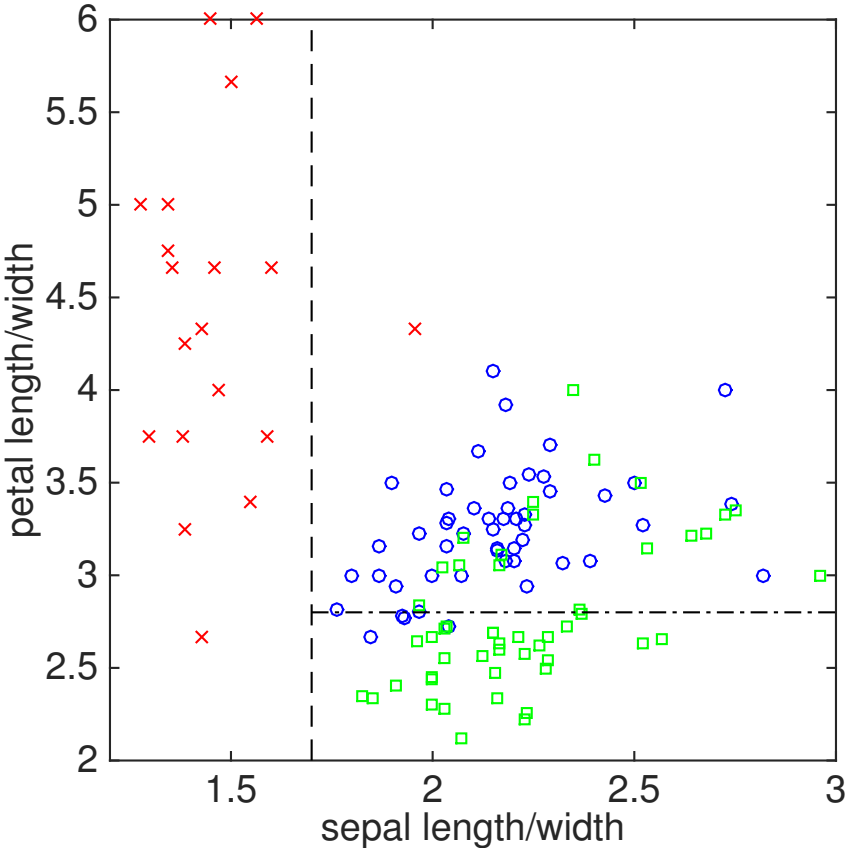


Classifying irises using sepal and petal measurements:

- $x \in \mathbb{R}^2, y \in \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width

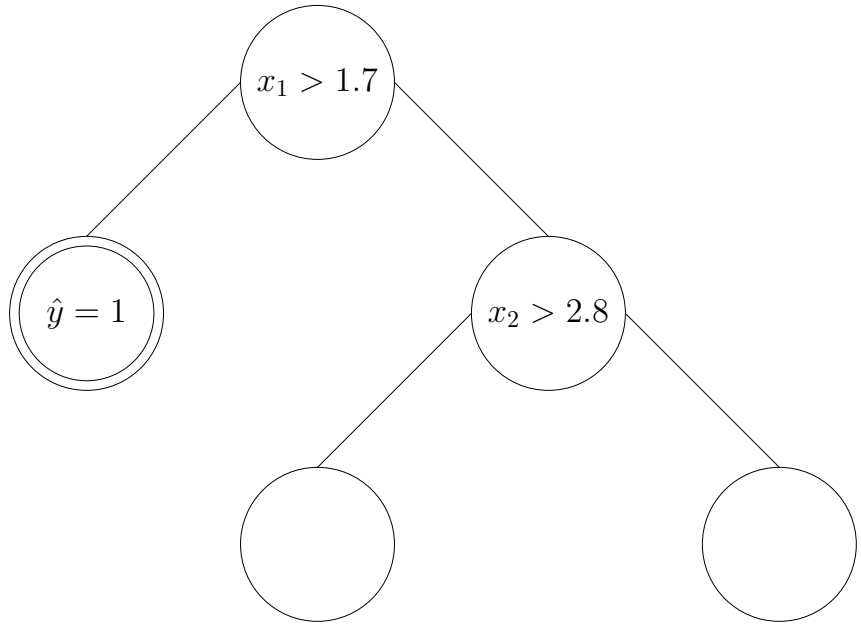


CLASSIFICATION TREES (EXAMPLE)

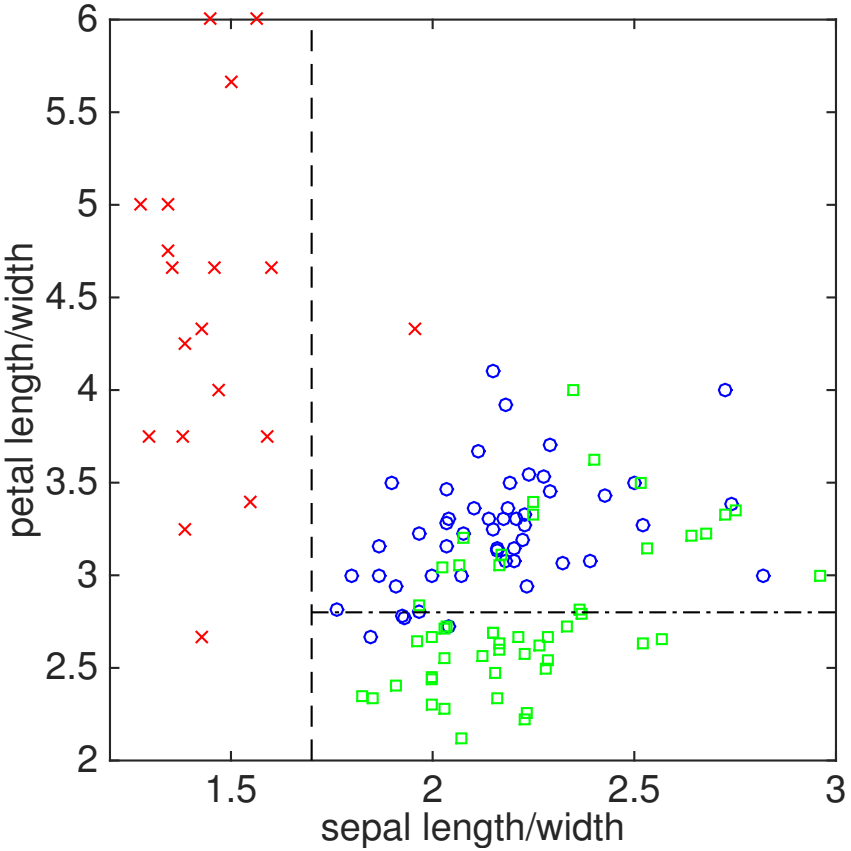


Classifying irises using sepal and petal measurements:

- $x \in \mathbb{R}^2, y \in \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width

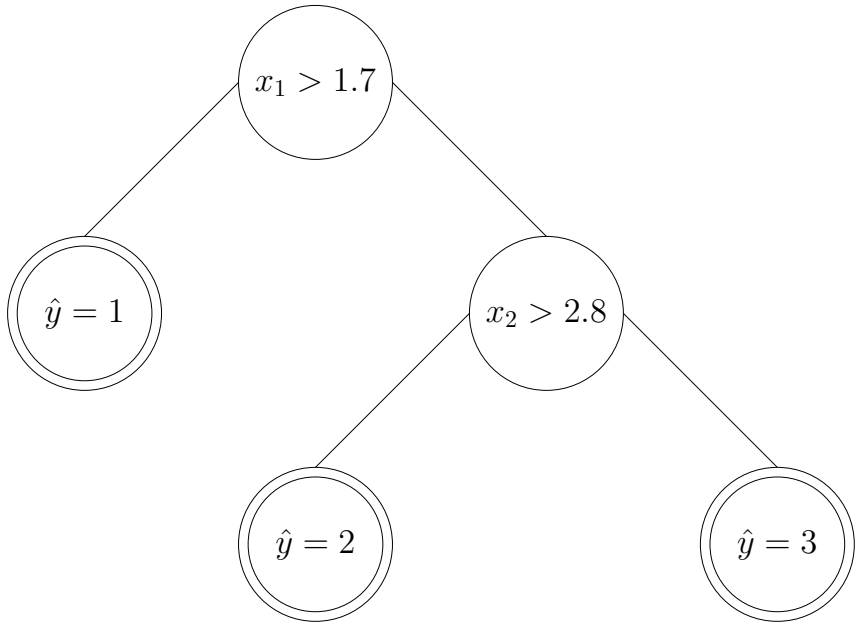


CLASSIFICATION TREES (EXAMPLE)

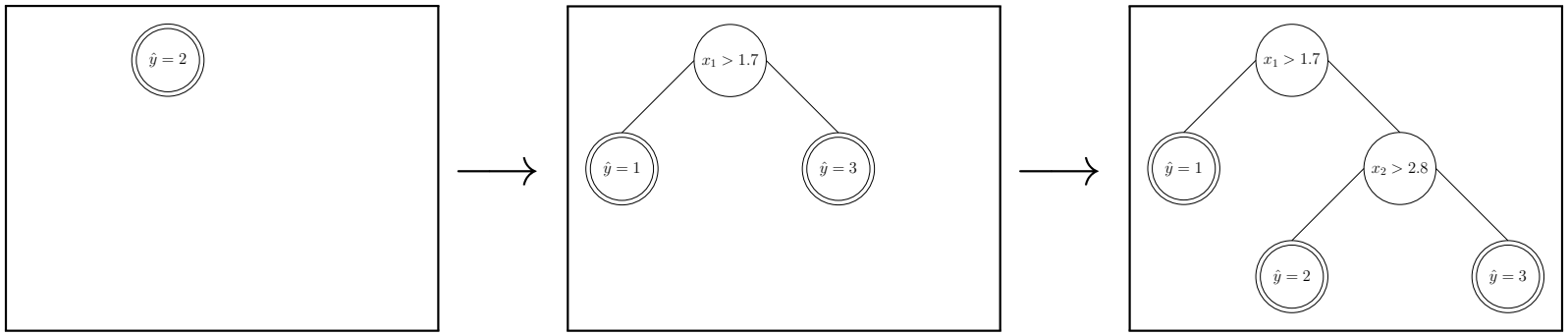


Classifying irises using sepal and petal measurements:

- $x \in \mathbb{R}^2, y \in \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width



BASIC DECISION TREE LEARNING ALGORITHM



The basic method for learning trees is with a top-down greedy algorithm.

- Start with a single leaf node containing all data
- Loop through the following steps:
  - Pick the leaf to split that reduces uncertainty the most.
  - Figure out the $\lessgtr$ decision rule on one of the dimensions.
- Stopping rule discussed later.

Label/response of the leaf is majority-vote/average of data assigned to it.

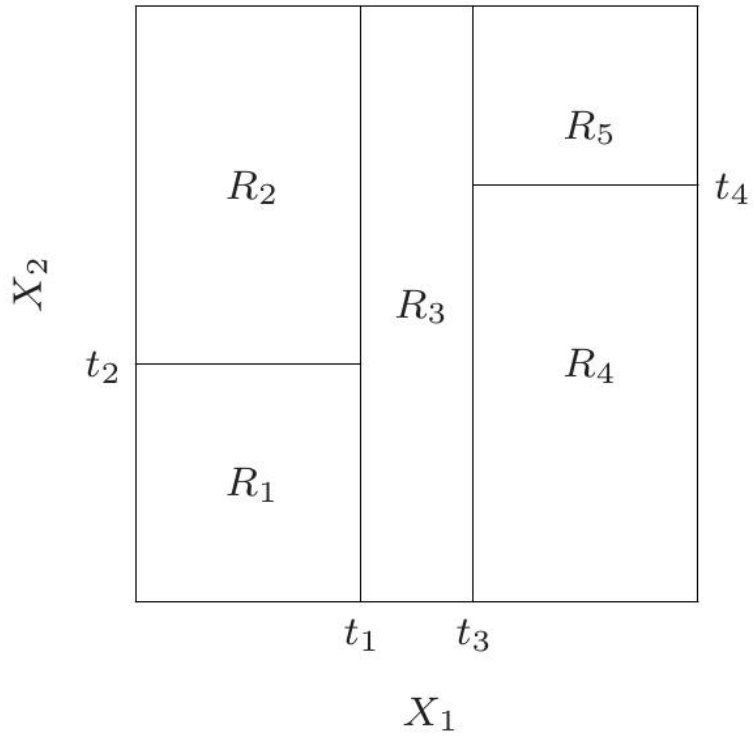
GROWING A REGRESSION TREE

How do we grow a regression tree?

- For M regions of the space, $R_1, \ldots, R_M$, the prediction function is

$$f(x) = \sum_{m=1}^{M} c_m \mathbb{1}\{x \in R_m\}.$$

So for a fixed M, we need R_m and c_m.



Goal: Try to minimize $\sum_i (y_i - f(x_i))^2$.

1. Find c_m given R_m: Simply the average of all y_i for which $x_i \in R_m$.
2. How do we find regions? Consider splitting region R at value s of dim j:
   - Define $R^-(j,s) = \{x_i \in R | x_i(j) \leq s\}$ and $R^+(j,s) = \{x_i \in R | x_i(j) > s\}$
   - For each dimension j, calculate the best splitting point s for that dimension.
   - Do this for each region (leaf node). Pick the one that reduces the objective most.

GROWING A CLASSIFICATION TREE

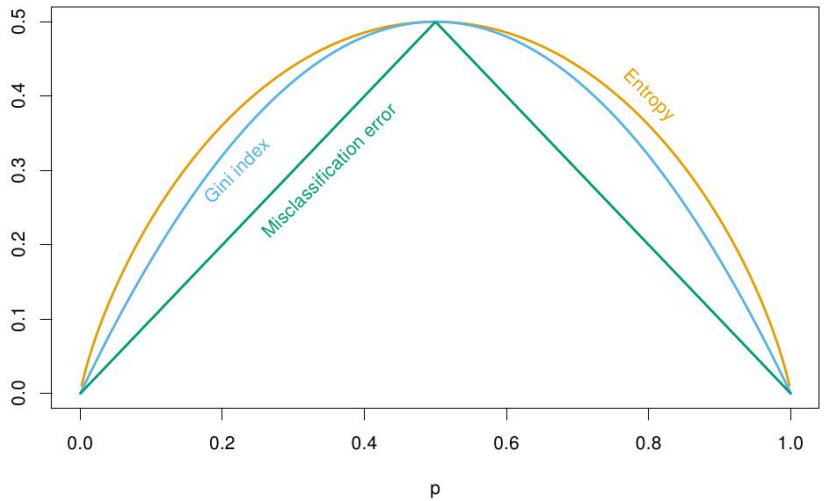
For regression: Squared error is a natural way to define the splitting rule.

For classification: Need some measure of how badly a region classifies data and how much it can improve if it's split.

K-class problem: For all $x \in R_m$, let p_k be empirical fraction labeled k.

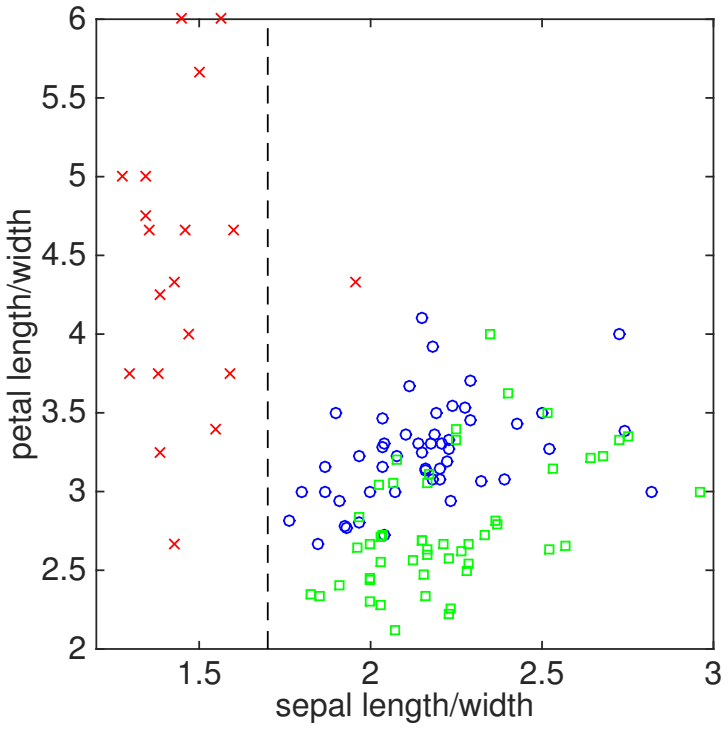
Measures of quality of R_m include

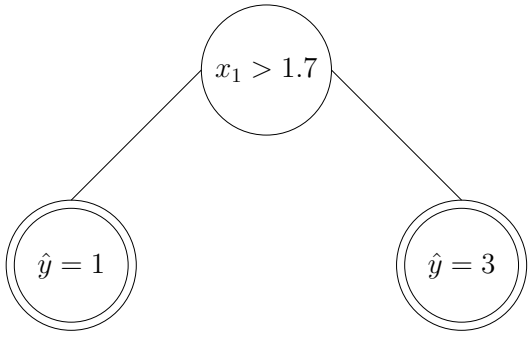
1. Classification error: $1 - \max_k p_k$
2. Gini index: $1 - \sum_k p_k^2$
3. Entropy: $-\sum_k p_k \ln p_k$



- These are all *maximized* when p_k is uniform on the K classes in R_m.
- These are *minimized* when $p_k = 1$ for some k (R_m only contains one class)

GROWING A CLASSIFICATION TREE





Search R_1 and R_2 for splitting options.

1. R_1: $y = 1$ leaf classifies perfectly
2. R_2: $y = 3$ leaf has Gini index

$$u(R_2) = 1 - \left(\frac{1}{101}\right)^2 - \left(\frac{50}{101}\right)^2 - \left(\frac{50}{101}\right)^2$$
$$= 0.5098$$

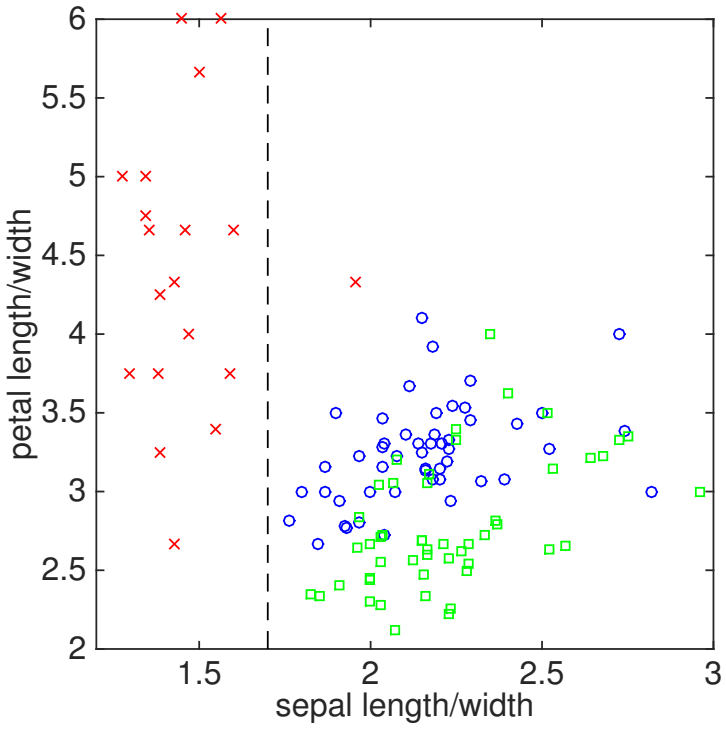
Gini improvement from split R_m to R_m^- & R_m^+:

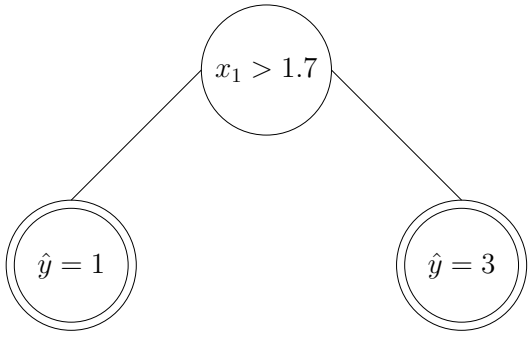
$$u(R_m) - \left(p_{R_m^-} \cdot u(R_m^-) + p_{R_m^+} \cdot u(R_m^+)\right)$$

$p_{R_m^+}$: Fraction of data in R_m split into R_m^+.

$u(R_m^+)$: New quality measure in region R_m^+.

GROWING A CLASSIFICATION TREE



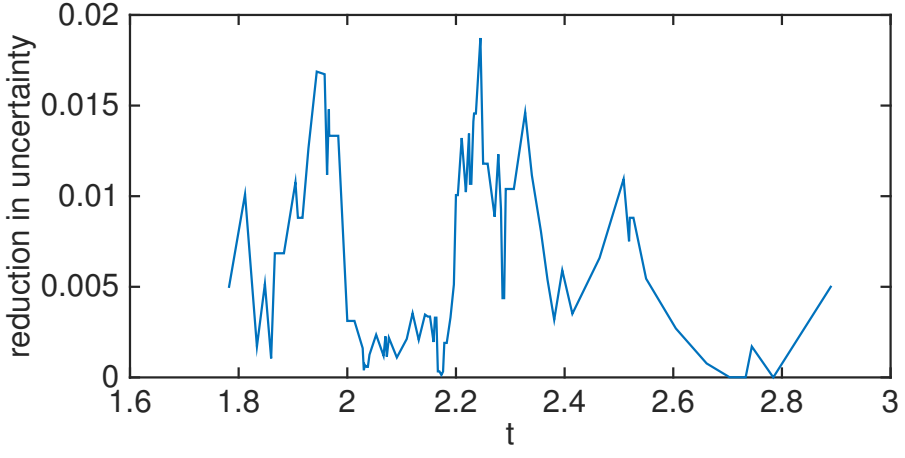


Search R_1 and R_2 for splitting options.

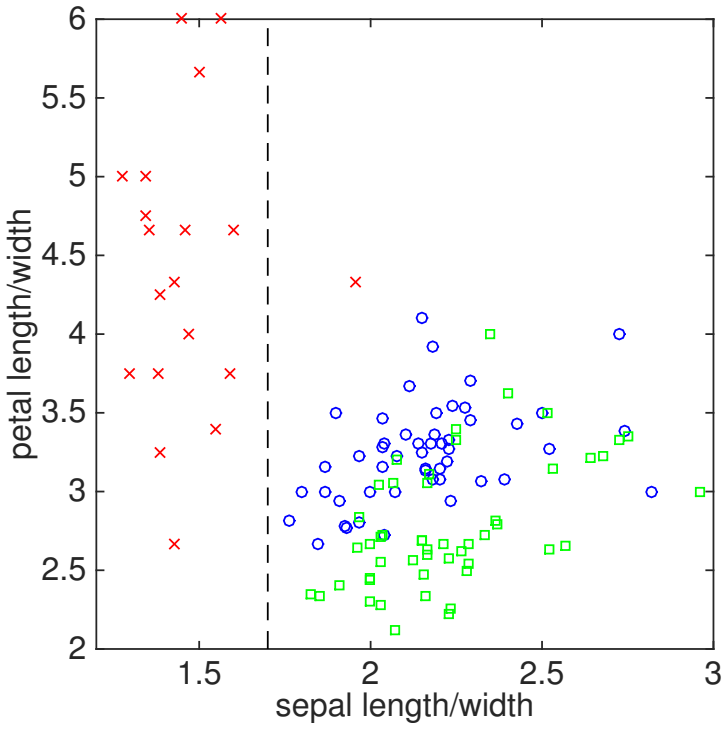
1. R_1: $y = 1$ leaf classifies perfectly
2. R_2: $y = 3$ leaf has Gini index

$$u(R_2) = 1 - \left(\frac{1}{101}\right)^2 - \left(\frac{50}{101}\right)^2 - \left(\frac{50}{101}\right)^2$$
$$= 0.5098$$

Check split R_2 with $\mathbb{1}\{x_1 > t\}$



GROWING A CLASSIFICATION TREE

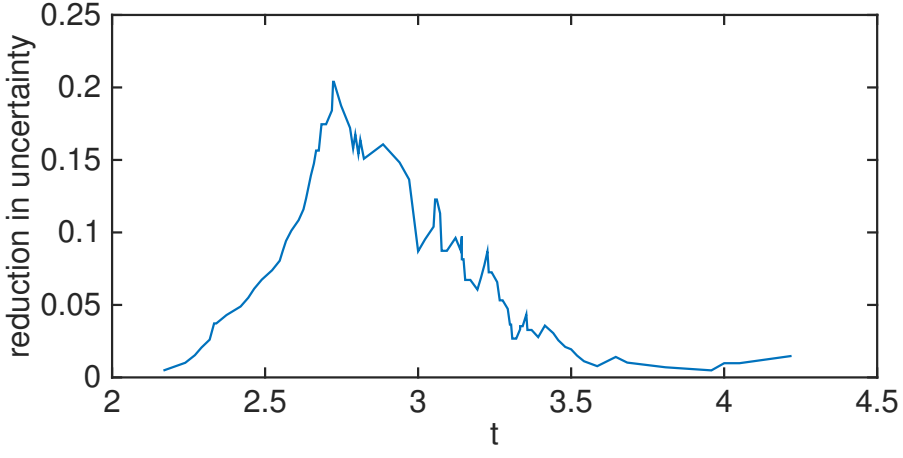


Search R_1 and R_2 for splitting options.

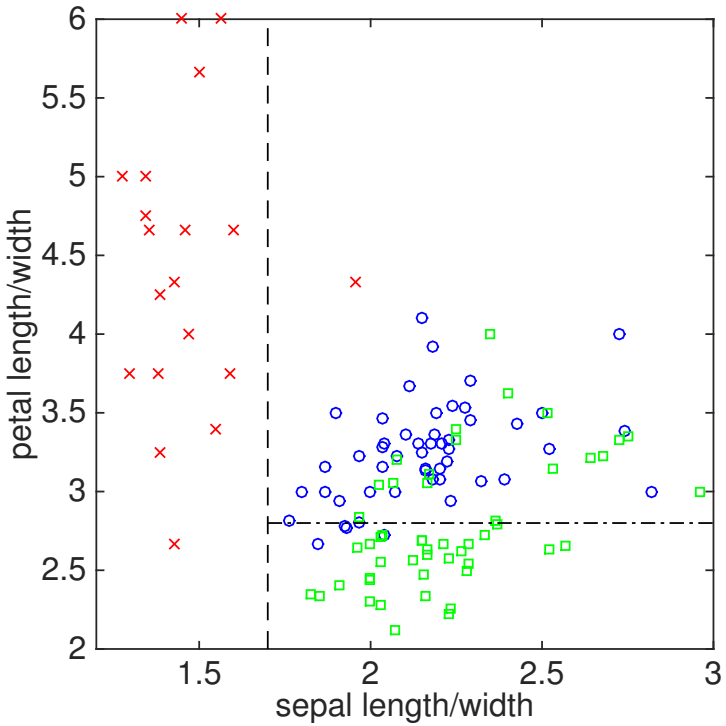
1. R_1: $y = 1$ leaf classifies perfectly
2. R_2: $y = 3$ leaf has Gini index

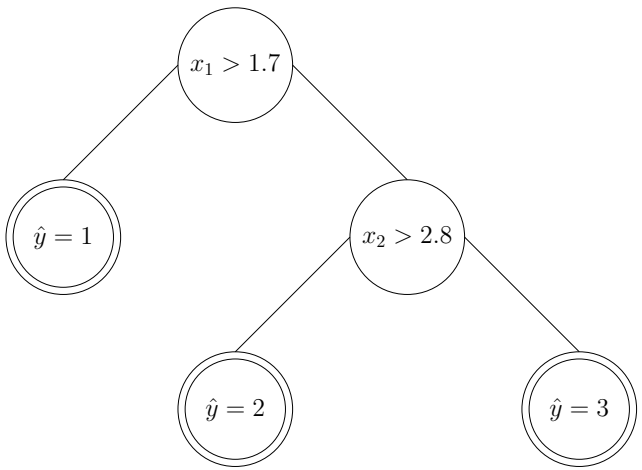
$$u(R_2) = 1 - \left(\frac{1}{101}\right)^2 - \left(\frac{50}{101}\right)^2 - \left(\frac{50}{101}\right)^2$$
$$= 0.5098$$

Check split R_2 with $\mathbb{1}\{x_2 > t\}$



GROWING A CLASSIFICATION TREE



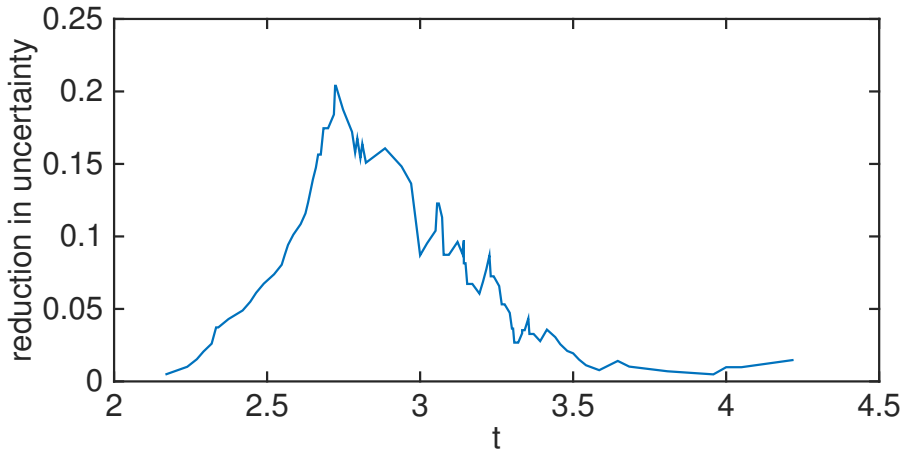


Search R_1 and R_2 for splitting options.

1. R_1: $y = 1$ leaf classifies perfectly
2. R_2: $y = 3$ leaf has Gini index

$$u(R_2) = 1 - \left(\frac{1}{101}\right)^2 - \left(\frac{50}{101}\right)^2 - \left(\frac{50}{101}\right)^2$$
$$= 0.5098$$

Check split R_2 with $\mathbb{1}\{x_2 > t\}$



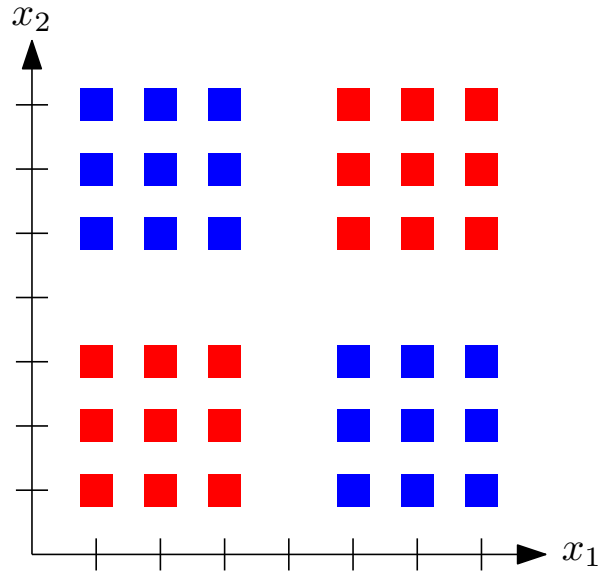
PRUNING A TREE

Q: When should we stop growing a tree?

A: Uncertainty reduction is not best way.

Example: Any split of x_1 or x_2 at right will show *zero* reduction in uncertainty.

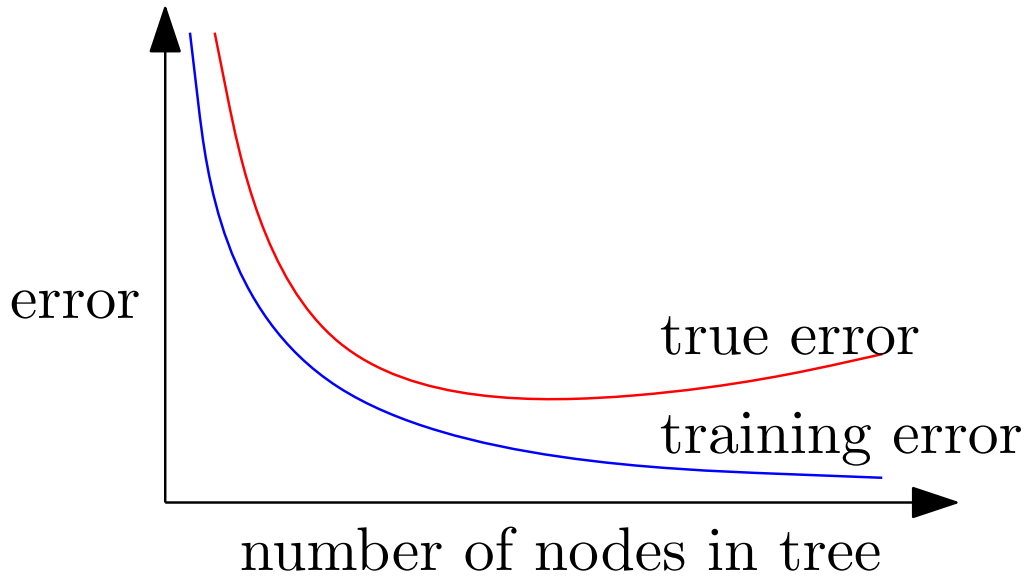
However, we can learn a perfect tree on this data by partitioning in quadrants.



Pruning is the method most often used. Grow the tree to a very large size. Then use an algorithm to trim it back.

(We won't cover the algorithm, but mention that it's non-trivial.)

OVERFITTING



- *Training error* goes to zero as size of tree increases.
- *Testing error* decreases, but then increases because of *overfitting*.

THE BOOTSTRAP

The Bootstrap: A Resampling Technique

We briefly present a technique called the *bootstrap*. This statistical technique is used as the basis for learning *ensemble classifiers*.

Bootstrap

Bootstrap (i.e., resampling) is a technique for improving estimators.

> Resampling = Sampling from the empirical distribution of the data

Application to ensemble methods

- We will use resampling to generate many "mediocre" classifiers.
- We then discuss how "bagging" these classifiers improves performance.
- First, we cover the bootstrap in a simpler context.

Bootstrap: Basic algorithm

Input

- A sample of data $x_1, \ldots, x_n$.
- An estimation rule $\hat{S}$ of a statistic S. For example, $\hat{S} = \text{med}(x_{1:n})$ estimates the true median S of the unknown distribution on x.

Bootstrap algorithm

1. Generate *bootstrap samples* $\mathcal{B}_1, \ldots, \mathcal{B}_B$.
   - Create $\mathcal{B}_b$ by picking points from $\{x_1, \ldots, x_n\}$ randomly n times.
   - A particular x_i can appear in $\mathcal{B}_b$ many times (it's simply duplicated).
2. Evaluate the estimator on each $\mathcal{B}_b$ by pretending it's the data set:

$$\hat{S}_b := \hat{S}(\mathcal{B}_b)$$

3. Estimate the mean and variance of $\hat{S}$:

$$\mu_B = \frac{1}{B}\sum_{b=1}^{B} \hat{S}_b, \quad \sigma_B^2 = \frac{1}{B}\sum_{b=1}^{B} (\hat{S}_b - \mu_B)^2$$

EXAMPLE: VARIANCE ESTIMATION OF THE MEDIAN

- The median of $x_1, \ldots, x_n$ (for $x \in \mathbb{R}$) is found by simply sorting them and taking the middle one, or the average of the two middle ones.

- How confident can we be in the estimate median$(x_1, \ldots, x_n)$?
  - Find it's variance.
  - But how? Answer: By bootstrapping the data.

1. Generate bootstrap data sets $\mathcal{B}_1, \ldots, \mathcal{B}_B$.

2. Calculate: (notice that $\hat{S}_{mean}$ is the mean of the median)

$$\hat{S}_{mean} = \frac{1}{B}\sum_{b=1}^{B} \text{median}(\mathcal{B}_b), \quad \hat{S}_{var} = \frac{1}{B}\sum_{b=1}^{B} \Big(\text{median}(\mathcal{B}_b) - \hat{S}_{mean}\Big)^2$$

- The procedure is remarkably simple, but has a lot of theory behind it.

Bagging and Random Forests

Bagging

Bagging uses the bootstrap for regression or classification:

$$\textbf{Bagging} \;=\; \textbf{B}\text{ootstrap } \textbf{agg}\text{regation}$$

Algorithm

For $b = 1, \ldots, B$:

1. Draw a bootstrap sample $\mathcal{B}_b$ of size n from training data.
2. Train a classifier or regression model f_b on $\mathcal{B}_b$.

- For a new point x_0, compute:

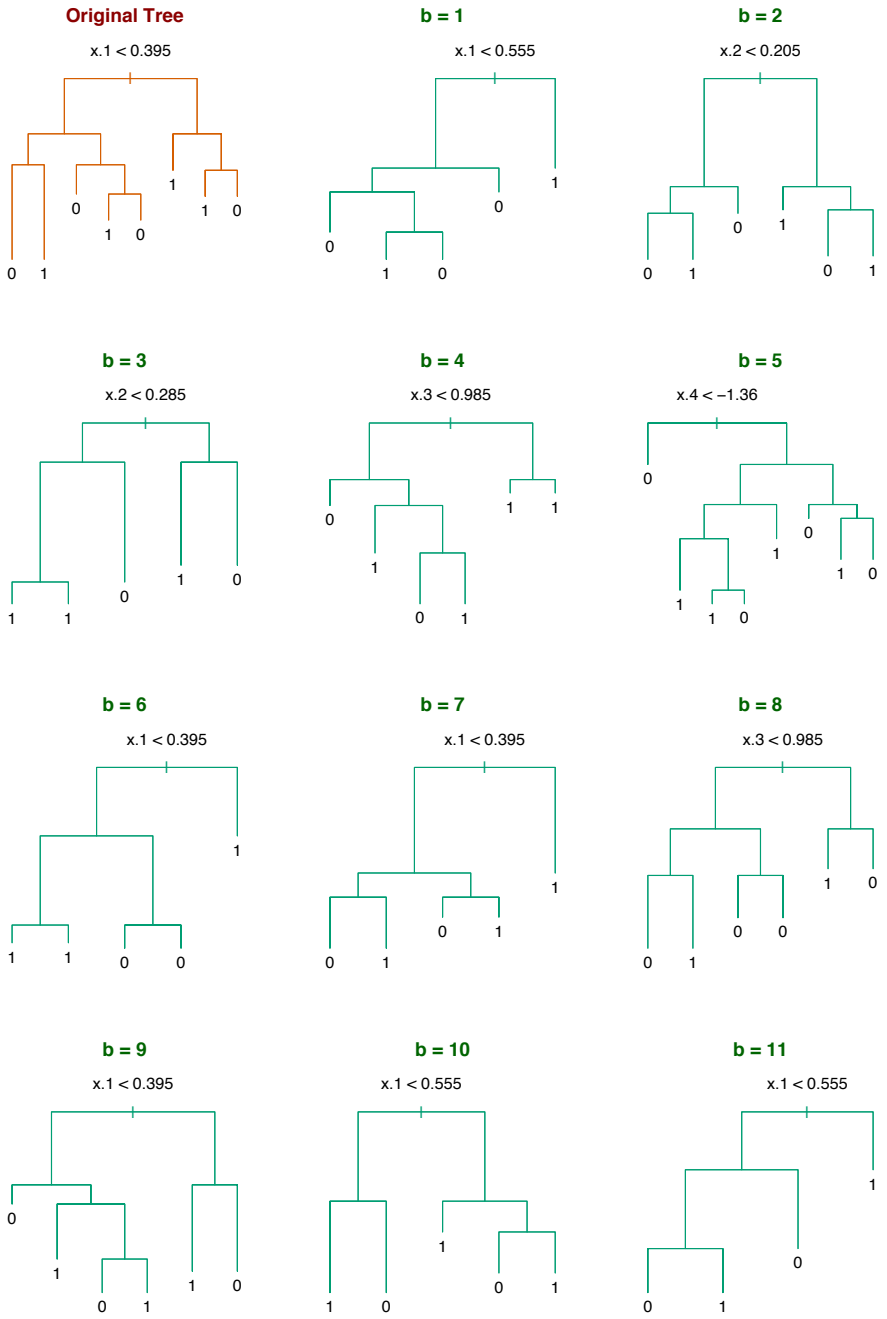
$$f_{\text{avg}}(x_0) = \frac{1}{B}\sum_{b=1}^{B} f_b(x_0)$$

  - For regression, $f_{\text{avg}}(x_0)$ is the prediction.
  - For classification, view $f_{\text{avg}}(x_0)$ as an average over B votes. Pick the majority.

Example: Bagging Trees

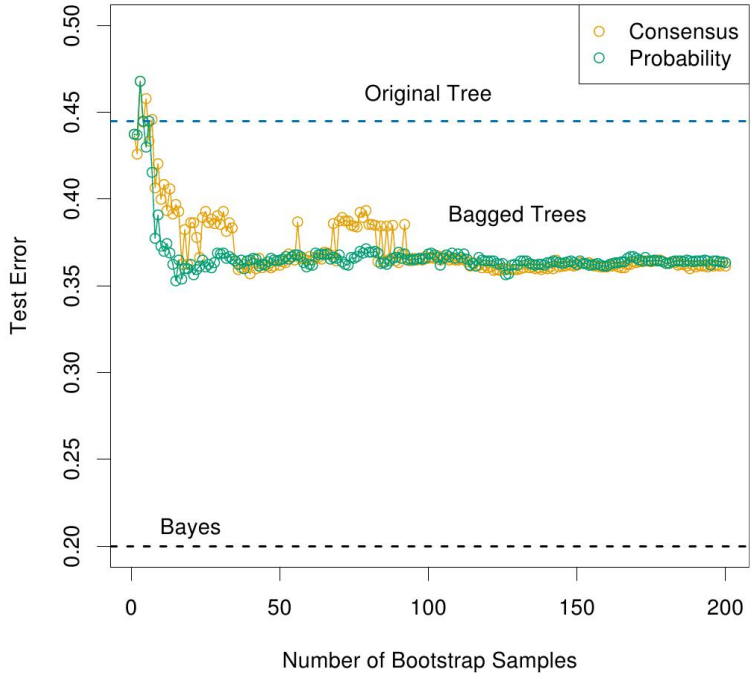
- Binary classification, $x \in \mathbb{R}^5$.
- Note the variation among bootstrapped trees.
- Take-home message:

  With bagging, each tree doesn't have to be great, just "ok".
- Bagging often improves results *when the function is non-linear*.



Drawbacks of Bagging

- Bagging works on trees because of the bias-variance tradeoff ($\uparrow$ bias, $\downarrow$ variance).
- However, the bagged trees are correlated.
- In general, when bootstrap samples are correlated, the benefit of bagging decreases.



Random Forests

Modification of bagging where trees are designed to reduce correlation.

- A very simple modification.
- Still learn a tree on each bootstrap set, $\mathcal{B}_b$.
- To split a region, only consider random subset of dimensions of $x \in \mathbb{R}^d$.

Random Forests: Algorithm

Training

Input parameter: m — a positive integer with $m < d$, often $m \approx \sqrt{d}$

For $b = 1, \ldots, B$:

1. Draw a bootstrap sample $\mathcal{B}_b$ of size n from the training data.
2. Train a tree classifier on $\mathcal{B}_b$, where each split is computed as follows:
   - Randomly select m dimensions of $x \in \mathbb{R}^d$, newly chosen for each b.
   - Make the best split restricted to that subset of dimensions.

- Bagging for trees: Bag trees learned using the original algorithm.
- Random forests: Bag trees learned using algorithm on this slide.

Random Forests

Example problem

- Random forest classification.
- Forest size: A few hundred trees.
- Notice there is a tendency to align decision boundary with the axis.