Machine Learning

15.S60 - Computing in Optimization and Statistics

Clark Pixton, Colin Pawlowski

MIT Operations Research Center

Overview

Outline

Machine Learning: Intro

The goals of **Machine Learning** are:

1. to discover mathematical relationships in the world, and
2. to make predictions for the future,

based upon data.

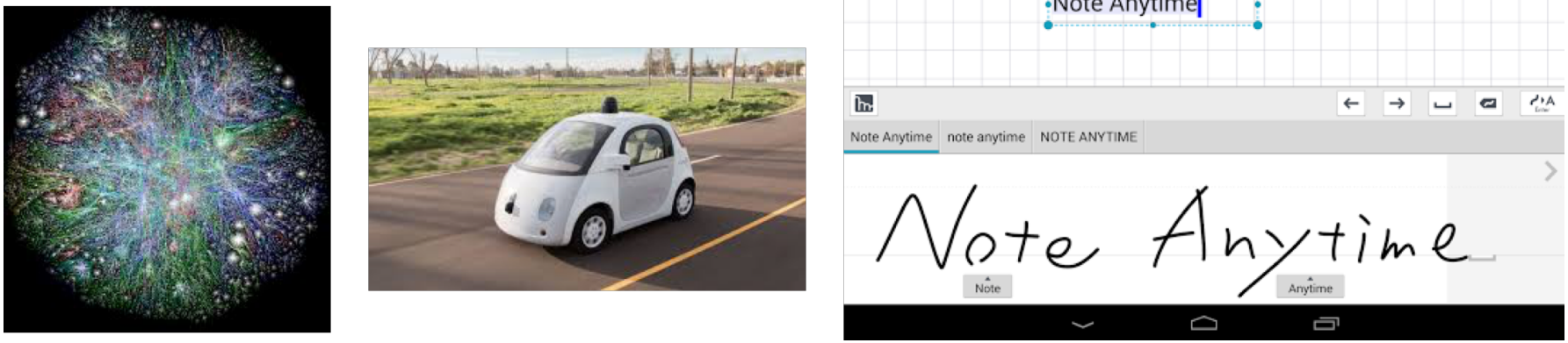


Figure: (From left to right) Graph of the internet, Google's self-driving car, and handwriting recognition software.

Machine Learning: Intro

Given i.i.d. data $\mathbf{x}_i \in \mathbb{R}^p$, $i = 1, \ldots, n$, there are two general classes of machine learning problems:

- **Supervised Learning**
  - We have data labels $y_i \in \mathbb{R}$, $i = 1, \ldots, n$.
  - **Task:** Find the function $f : \mathbb{R}^p \to \mathbb{R}$ which predicts the *label* value y_i based on the *feature* vector $\mathbf{x}_i$, i.e: $f(\mathbf{x}_i) \approx y_i$.
  - Typically, we find f by solving some minimization problem:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \ell(y_i - f(\mathbf{x}_i)).$$

  - **Examples:** Linear regression, LASSO, logistic regression, CART, random forest, SVM, neural networks
- **Unsupervised Learning**
  - We have unlabelled data.
  - **Task:** Find patterns and relationships present in the data.
  - **Examples:** k-means, hierarchical clustering, anomaly detection methods

Outline

Linear Regression

- We wish to find a linear function of the features $\mathbf{x}_i$ which best approximates the label y_i.
- **Predict Function:** $f(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$
- In Ordinary Least Squares, we minimize the sum-of-squared error:

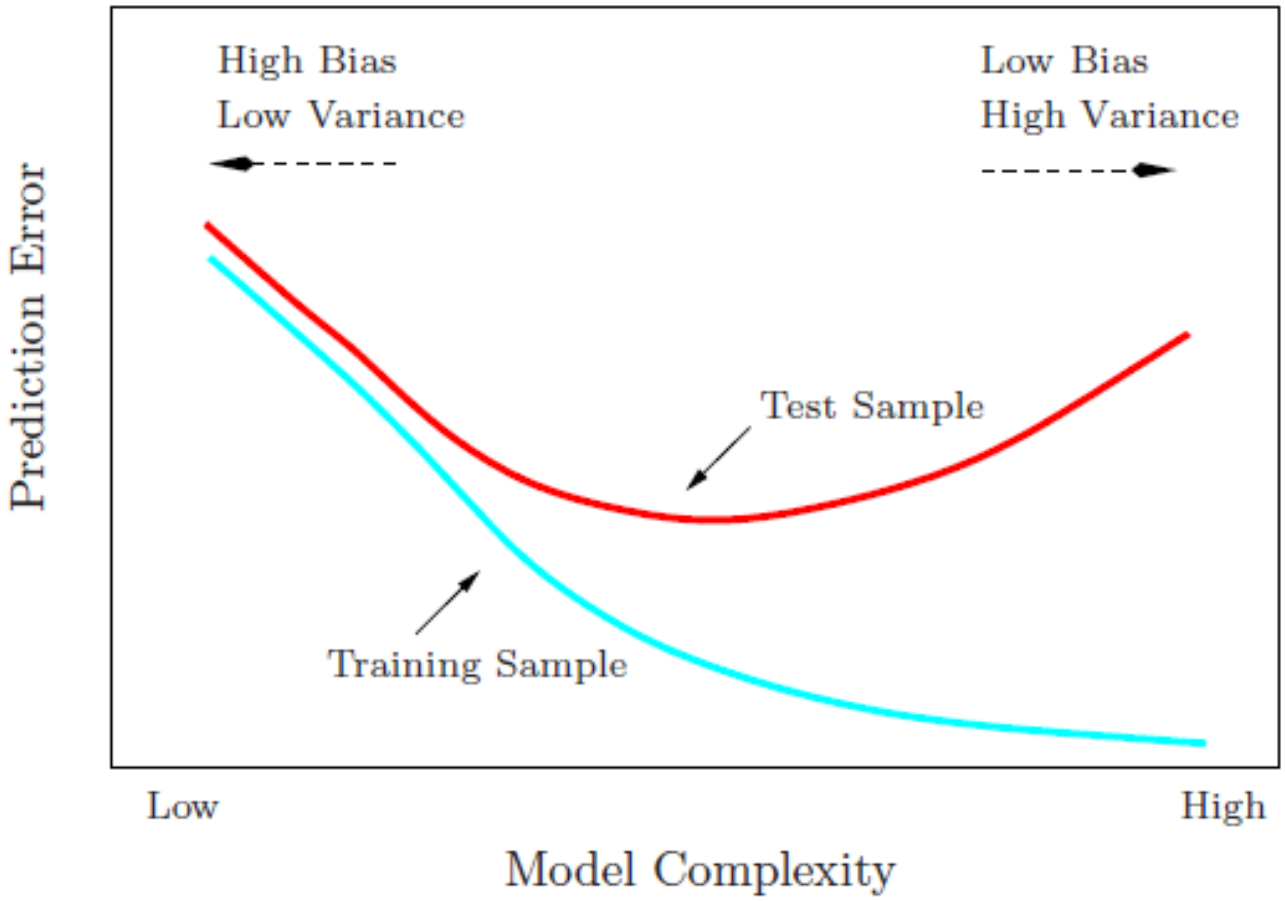
$$\min_{\beta} \sum_{i=1}^{n} (y_i - \beta^T \mathbf{x}_i)^2.$$

- We can measure how well a linear model explains the data using its R^2 value, which ranges from 0 (no fit) to 1 (perfect linear fit).
  - R^2 indicates the % variation in y that is explained by variation in $\mathbf{x}$.
  - Beware of overfitting!
- **Major issue:** Just because a model performs well on the data set used for training does not necessarily mean that it is an accurate model in general.

Model Selection

- In general, models that are too complex (i.e. linear regressions with too many variables) are prone to overfitting.
- However, models that are too simple (i.e. linear regressions with too few variables) will give poor predictions on both the training and testing data sets.
- There is a "sweet spot" in the middle, where a model is neither too complex nor too simple.
- **Regularization** is a popular method for tuning model complexity.
  - We add a penalty term to the objective function of the ML method with constant coefficient λ.
  - By varying λ, we can vary the complexity of the ML method.
  - In **Ridge Regression**, we add the regularization term $+\lambda\|\beta\|_2$.
  - In **LASSO**, we add the regularization term $+\lambda\|\beta\|_1$.

Bias-Variance Tradeoff



*From *The Elements of Statistical Learning* (2001)

Selecting the Model Parameters

- How do we determine the correct values for the model parameters?
  - For example, how do we select λ in LASSO?
- We follow a systematic procedure called **cross-validation** to select model parameters. Steps:
  1. Split the data into three groups: training, validation, and testing.
  2. For each unique combination of model parameters, build a separate ML model on the training data set.
  3. Evaluate the performance of each ML model on the validation set.
  4. Select the combination of model parameters which yields the best performance on the validation set. Use this set of parameters used to build the final model.
  5. Evaluate the performance of the final model on the testing set.

Logistic Regression

- Similar to linear regression, but used for binary classification. (e.g. predict whether or not a passenger on *Titanic* survived)
- Uses the logistic function to predict the probability of a class $\mathbb{P}(y = 1)$:

$$g(t) = \frac{1}{1 + e^{-t}}.$$

  - The output of the function $g : \mathbb{R} \to (0, 1]$ can be thought of as a probability.
- **Predict function:** $f(\mathbf{x}) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\ldots+\beta_p x_p)}}$
- We then use a threshold value on the output of the logistic regression.
  - Often this value is 0.5, indicating that we predict the class with higher probability.
  - Sometimes, other threshold values may be more appropriate. (Can you think of any scenarios where this might be the case?)

Logistic Regression

- To find the coefficients β, we solve the Maximum Likelihood Estimation problem:

$$\max_{\beta} -\sum_{i=1}^{n} \log\left(1 + e^{-y_i(\beta^T \mathbf{x}_i + \beta_0)}\right).$$

- In addition, to avoid overfitting, we add a regularization term to the objective.
  - L_1-regularized logistic regression: $-\lambda\|\beta\|_1$.
  - L_2-regularized logistic regression: $-\lambda\|\beta\|_2$.
- As in LASSO and ridge regression, we use **cross-validation** to select the regularization parameter λ.

CART

- Classification And Regression Trees is a heuristic method for constructing decision trees.
  - *Classification Trees* are used to predict binary or multi-class outcomes.
  - *Regression Trees* are used to predict continuous outcomes, although these tend to have poor performance.
- **Predict function:** A decision tree, such as:

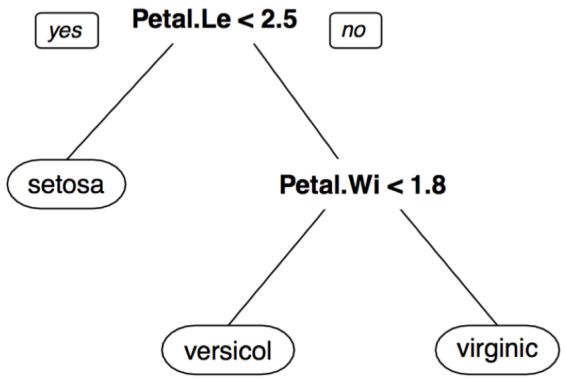


Figure: Decision tree obtained from running CART on the `iris` data set

- Machine learning methods for Optimal Decision Trees is an active research area (see recent works by Bertsimas and Dunn).

How does CART work?

- The algorithm makes sequential splits on the features.
  - For example: Did the Airbnb have ≥ 4 bedrooms? If yes, then consider price, if no, then consider # of bathrooms, and so on.
- Splits are made to make the "buckets" as "pure" as possible, in a greedy fashion.

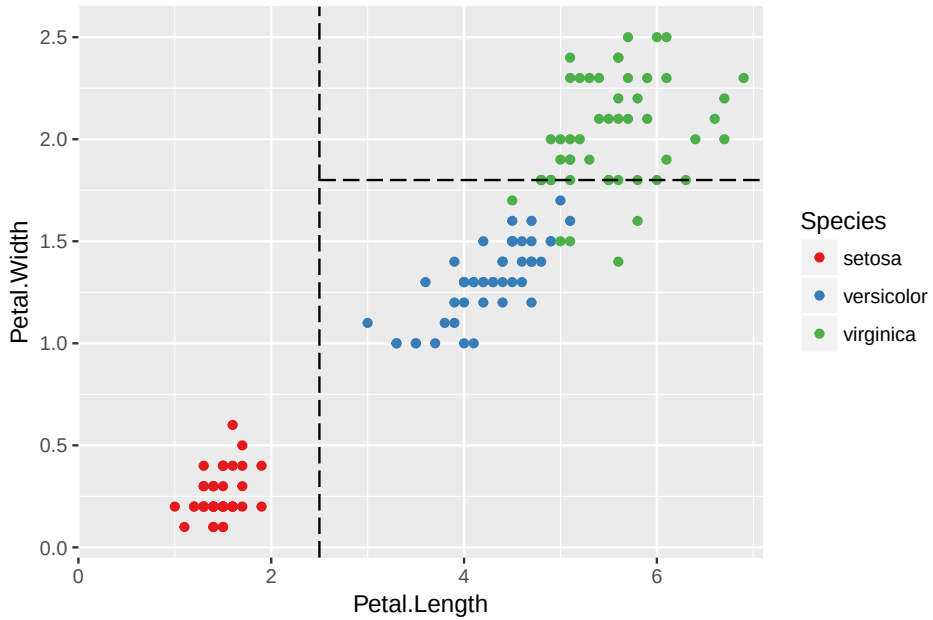
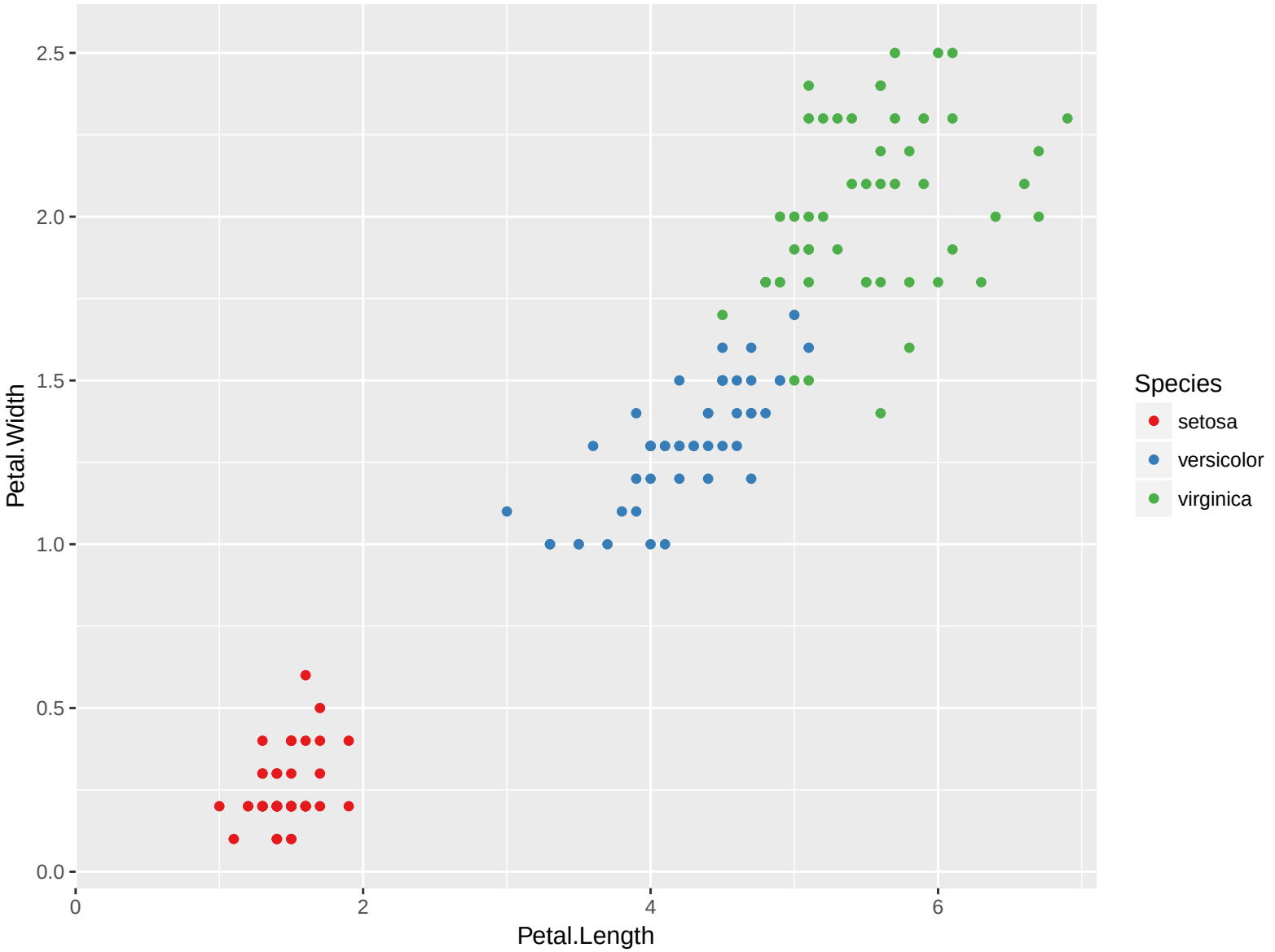
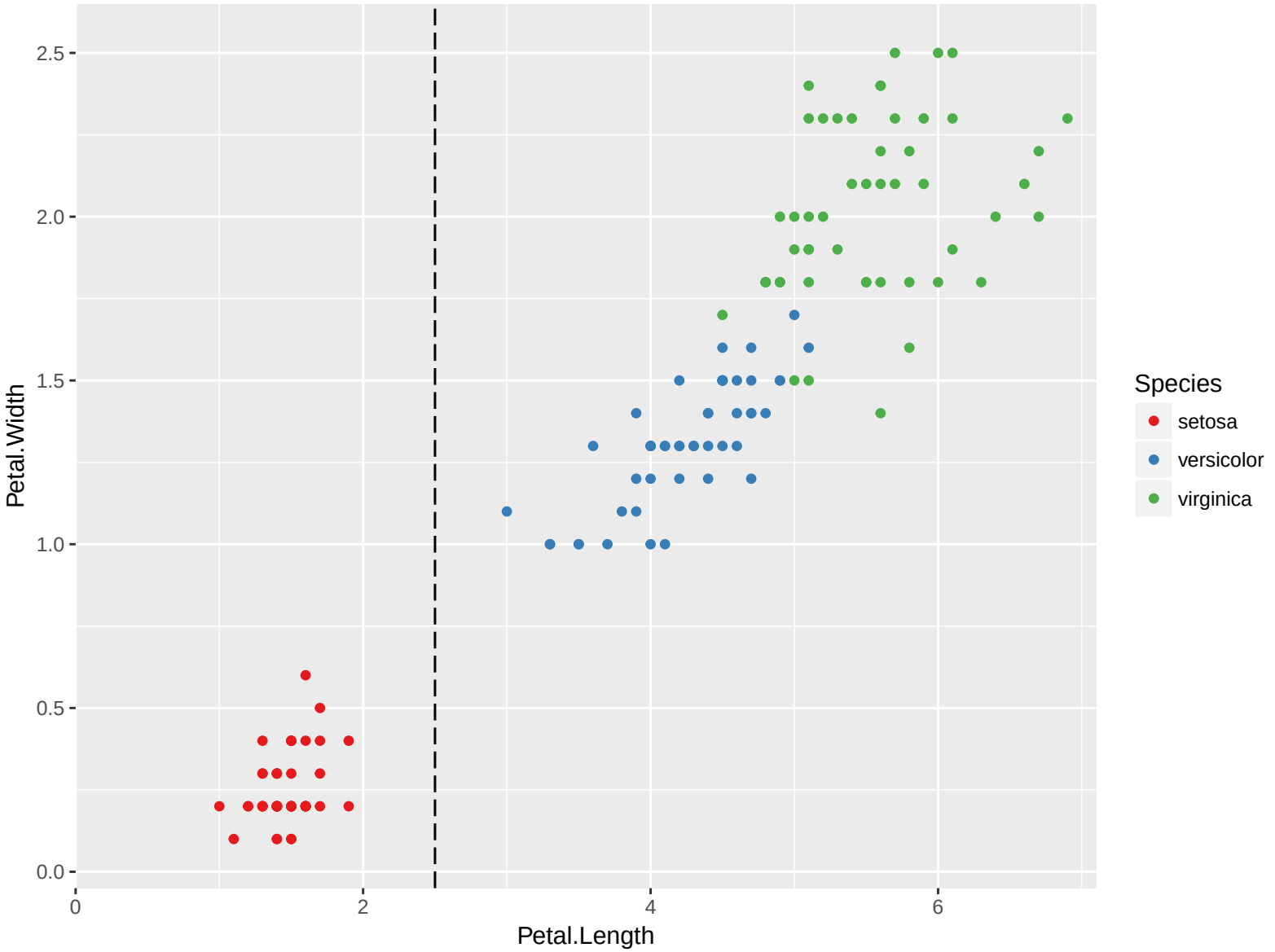


Figure: CART applied to the `iris` data set

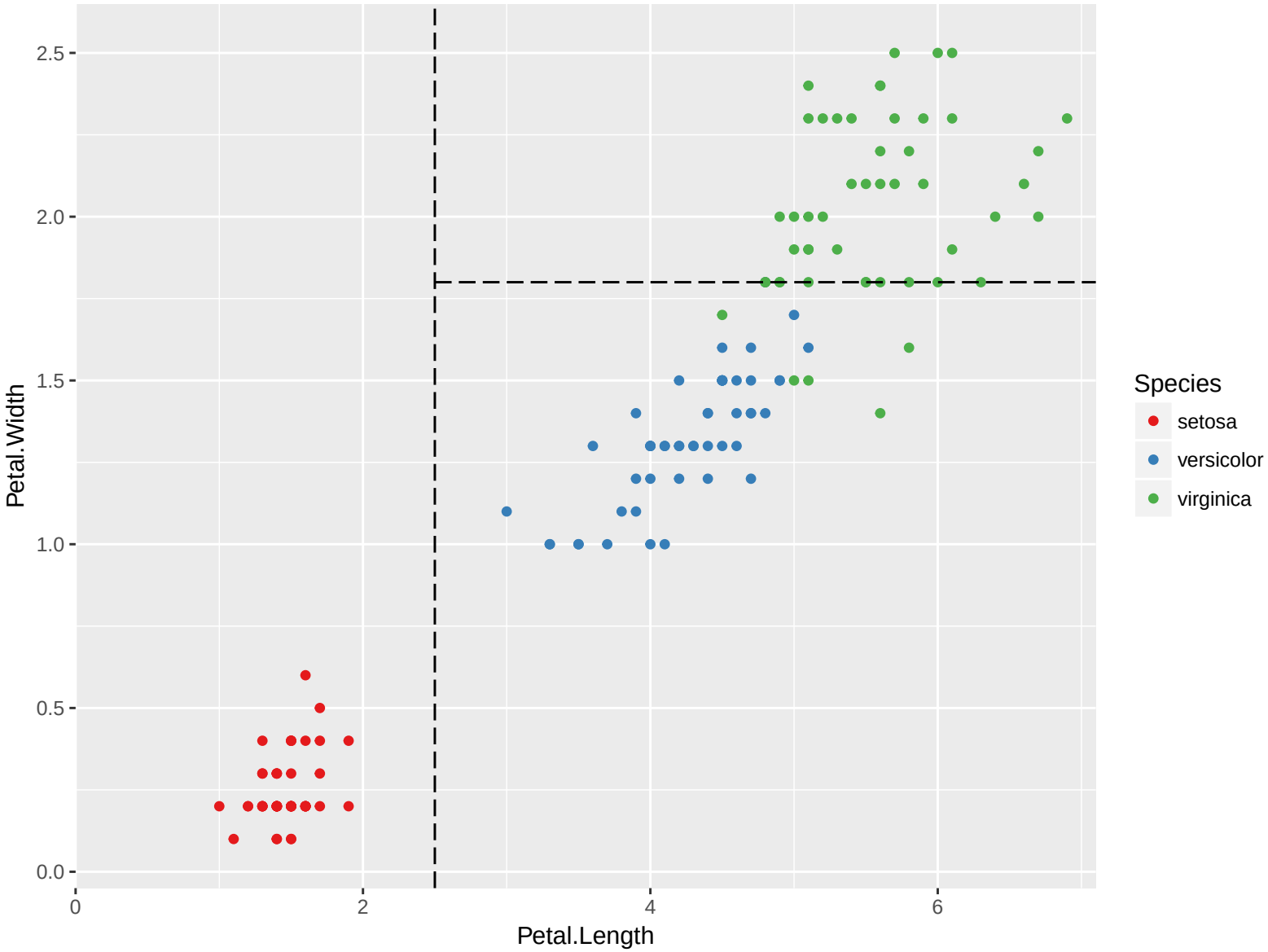
CART in action



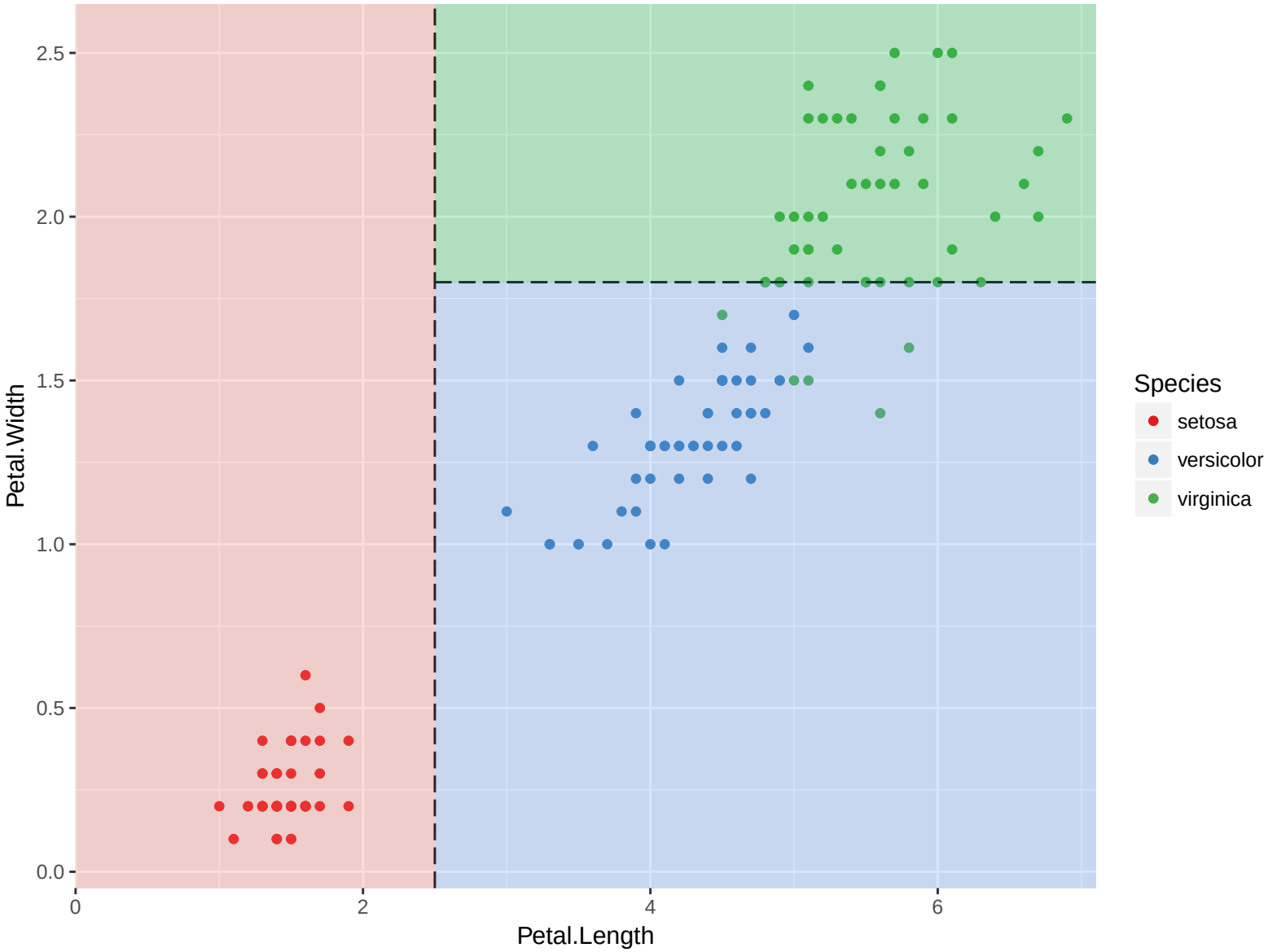
CART in action



CART in action



CART in action



How does CART choose the splits?

- The algorithm begins with a single *root* node which contains all of the data points.
- Each iteration, the algorithm checks all of the possible splits and selects the one that minimizes the overall **Gini index**, which for each node is:

$$1 - \sum_{j=1}^{m} P_j^2,$$

  where m is the total number of different classes, and P_j is the relative frequency of class j in the given node.
- We use the **Gini index** instead of a more natural measure, such as percentage of misclassifications in each node, because CART is a greedy algorithm and the trees turn out better if we use this complicated measure.

CART Model Parameters

- CART has many input parameters, including:
  - `minsplit`: The minimum number of data points that a node must have in order to be considered for a split.
  - `minbucket`: The minimum number of data points in each leaf node.
  - `cp`: The threshold complexity parameter which the algorithm uses to determine whether or not to split at each node. This indirectly controls the depth of the tree.
- If these thresholds are absent, then the CART algorithm will continue splitting until all points are correctly classified.
  - This results in an extremely deep tree which is completely overfit to the training data.
- Select appropriate values for CART model parameters to avoid overfitting or stopping the splitting procedure too early.
  - Use **cross-validation** for this step.

Outline

k-means Clustering

- The goal of k-means is to find *clusters* of data points which are relatively close to one another.
- We define "closeness" using a distance metric in the feature space, in this case the L_2 distance metric.
- If we use the L_1 distance metric, then we obtain k-median clustering.
- k-means is a heuristic to solve the non-convex mixed integer optimization problem:

$$\min \sum_{k=1}^{K} \sum_{i=1}^{n} z_{ik} \|\mathbf{x}_i - \bar{\mathbf{x}}_k\|_2,$$

where $\bar{\mathbf{x}}_k$ is the centroid of the kth cluster, and

$$z_{ik} = \begin{cases} 1, & \text{if } \mathbf{x}_i \text{ is in a cluster with centroid } \bar{\mathbf{x}}_k, \\ 0, & \text{otherwise.} \end{cases}$$

How does the k-means algorithm work?

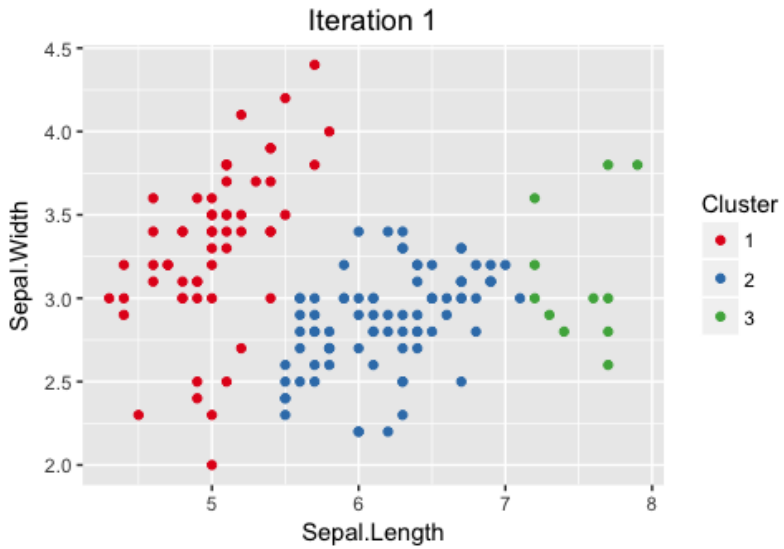
- First, we input the number of clusters K as a model parameter.
- The algorithm starts by randomly selecting K of the points to be centroids.
- Each iteration, we assign each point to its closest cluster centroid, and we recompute each cluster centroid as:

$$\bar{\mathbf{x}}_k = \frac{\sum_{i=1}^{n} z_{ik}\mathbf{x}_i}{\sum_{i=1}^{n} z_{ik}}.$$

- This is equivalent to applying the Newton-Raphson method to the original problem, so this method typically converges very fast to a locally optimal solution.
  - In particular, k-means is much faster than hierarchical clustering.
- In R, we can set the number of random starts (`nstart`) and maximum number of iterations (`iter.max`).

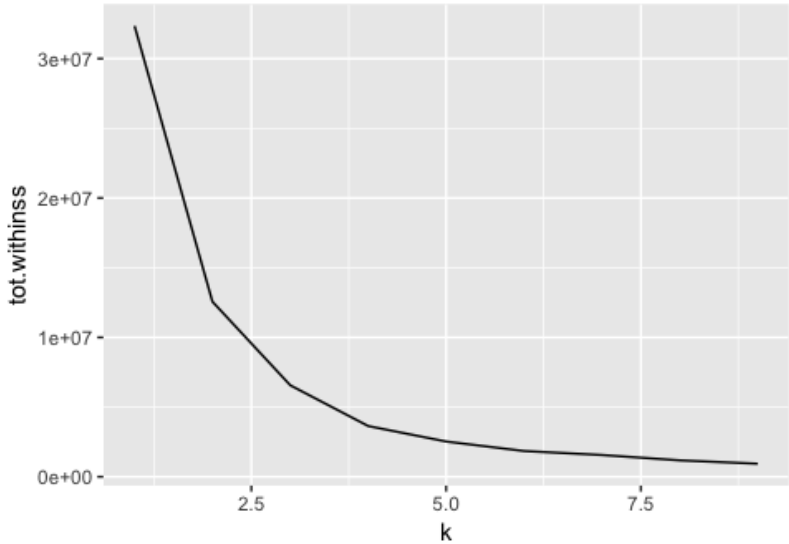
k-means in action

Converges on the `iris` flower data set in 2 iterations.



How do we find the correct number of clusters?

- In Unsupervised Learning, there is no systematic procedure like cross-validation to do parameter selection. (Why is that?)
- For k-means clustering, we can use an "elbow-plot" to get a rough sense of how many clusters to use.
  - We select a value for K that is relatively small, but still has small total sum-of-squared distances.
  - This occurs at the "elbow" in the graph.



Outline

Random Forest

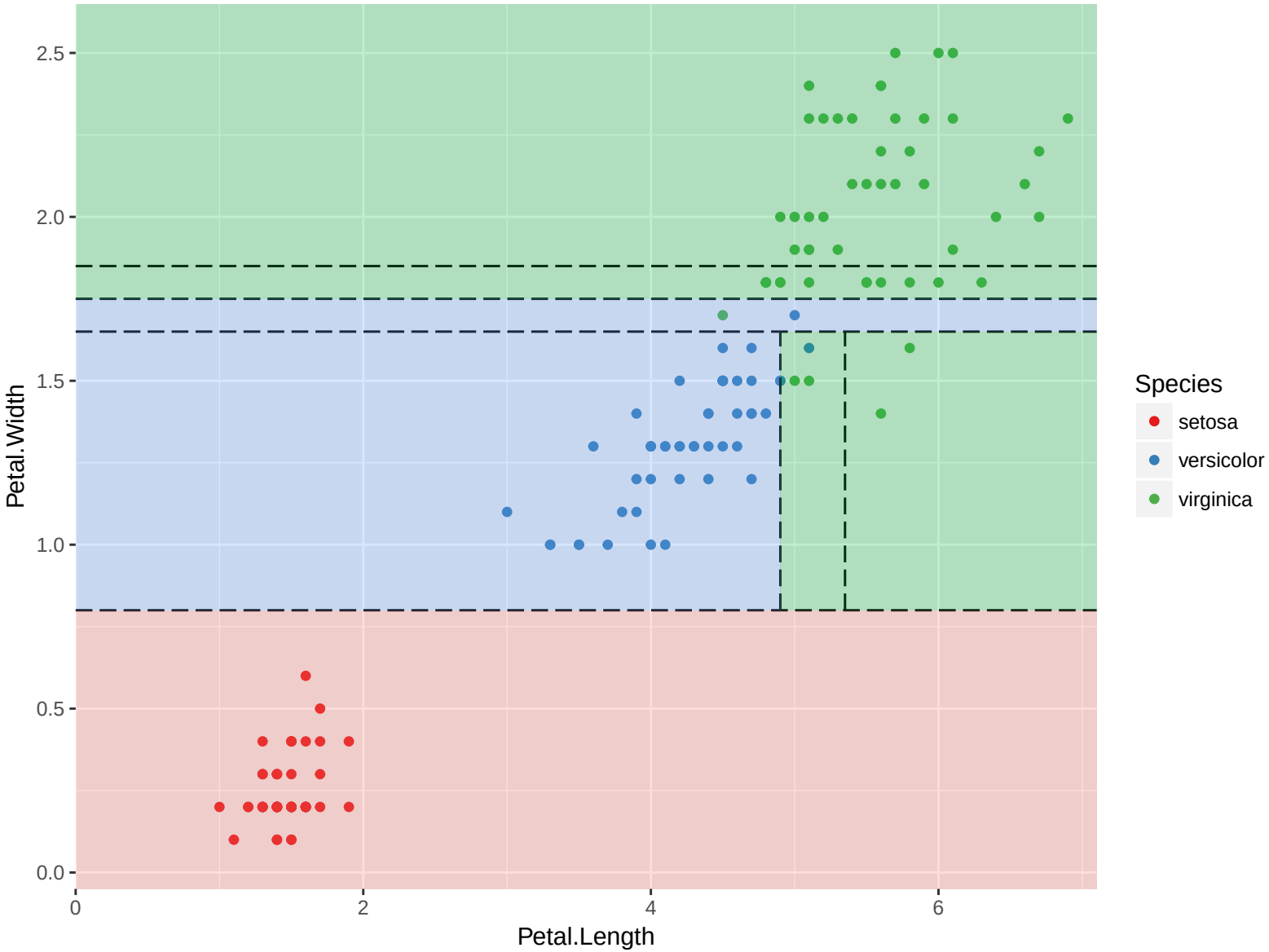
- Derived from CART, this is one of the highest performing methods for classification.
- **Predict function:** A bunch of decision trees averaged together.
- To predict the label of a new data point, we count up the predictions from all of the decision trees and then take the majority vote.
  - This is an example of *ensemble modeling*, which typically increases our predictive power.

Figure: Sequoia National Park

How does Random Forest work?

- To build the random forest, we create a bunch of decision trees using CART.
- In order to force the trees to be different, we restrict each CART tree to make splits using a random subset of the features.
- In addition, we allow the CART trees to continue splitting until the accuracy on the training data is almost 100%.
  - This results in very deep individual trees, which will be mostly unique.
- Because each individual tree is overfit completely, there is no need to specify the `minbucket` or `cp` parameter.
  - The main model parameter is `ntree`, the number of trees in the forest.

Individual Tree from Random Forest on `iris` data set



Support Vector Machines

- For classification problems, SVM optimizes the *decision boundary* in the feature space directly.
  - For example, in CART, the decision boundary was the set of splits that slice up the data.
- The *kernel* function determines the shape of the decision boundary.
  - If we choose a *linear* kernel, then the decision boundary will be a single hyperplane $\mathbf{w}^T\mathbf{x} = b$ in the feature space. (In 2-D, this is just a line)
  - If we choose a *polynomial* or *radial basis function* kernel, then the decision boundary can be curved.
- **Predict function:**

$$f(\mathbf{x}) = \begin{cases} \text{sign}\{\mathbf{w}^T\mathbf{x} - b\}, & \text{for a linear kernel,} \\ \text{sign}\{\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) - b\}, & \text{for a general kernel } K(\mathbf{x}_1, \mathbf{x}_2). \end{cases}$$

- The labels are assumed to be binary $y_i \in \{-1, +1\}$.
  - SVM may also be extended for multi-class problems, although these extensions are largely heuristic.

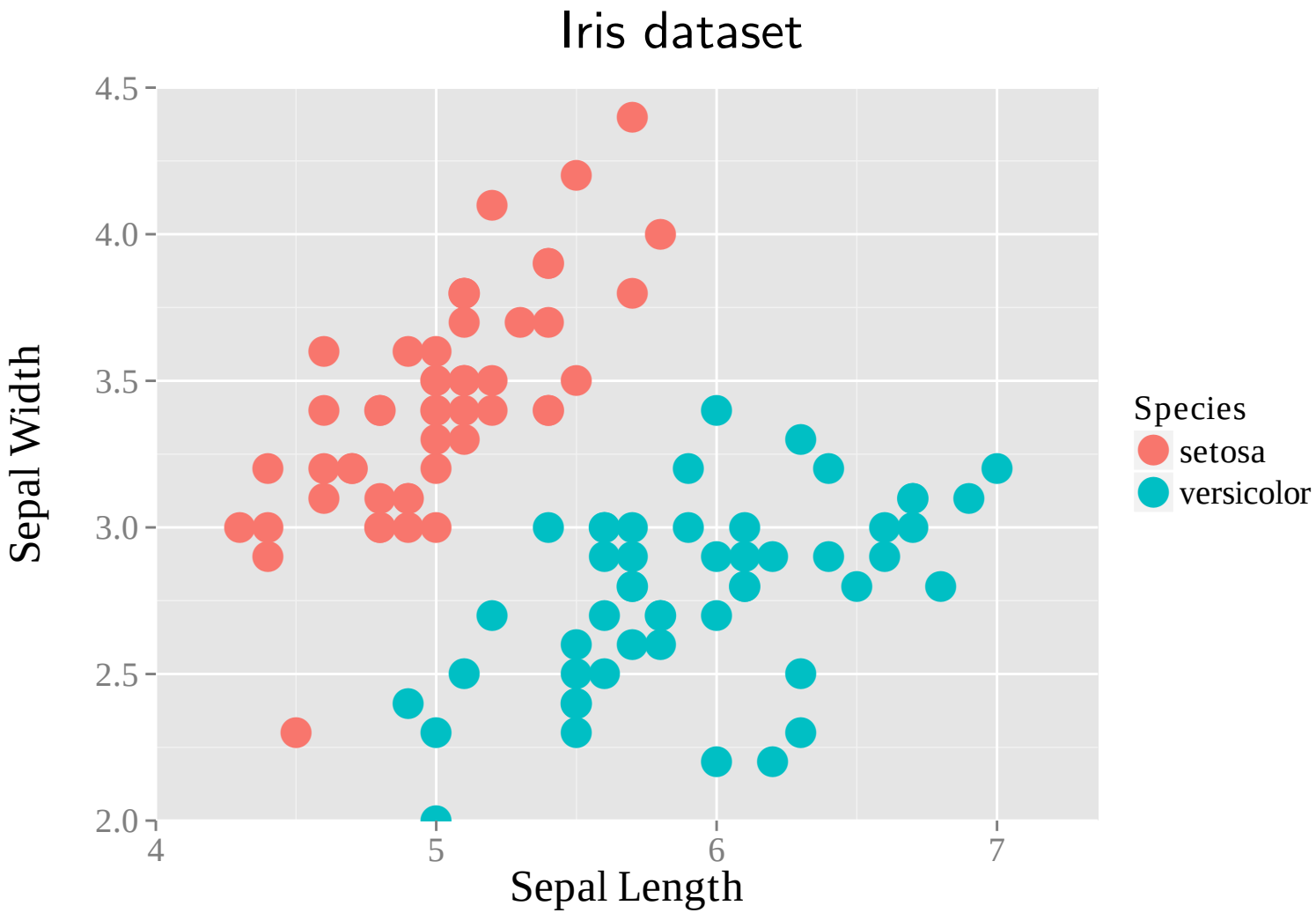
How does SVM work?

- We find the coefficients of the predict function (either $(\mathbf{w}, b)$ for a linear kernel, or (α, b) for a general kernel) by solving the following optimization problem:

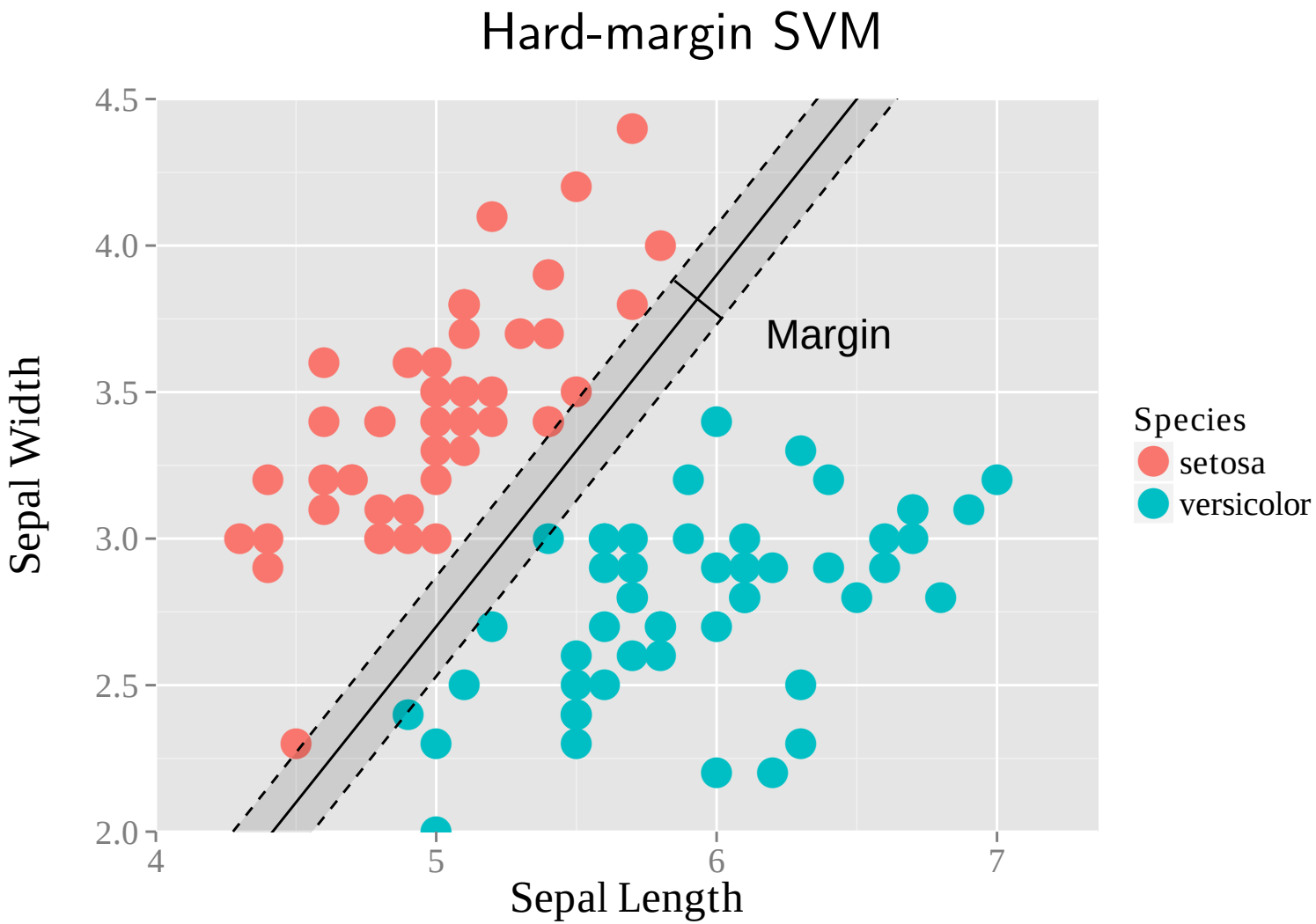
$$\begin{aligned} \max_{\alpha} \quad & C\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \quad i = 1, \ldots, n, \\ & \sum_{i=1}^{n}\alpha_i y_i = 0. \end{aligned}$$

- Because this is a convex, quadratic optimization problem, we have fast algorithms to find the optimal solution.
  - The `R` package `e1071` does this for you.
- The only model parameter that we need to input is C, which controls the model complexity. (also λ if we use the `rbf` kernel)
  - Use **cross-validation** to select these parameters.

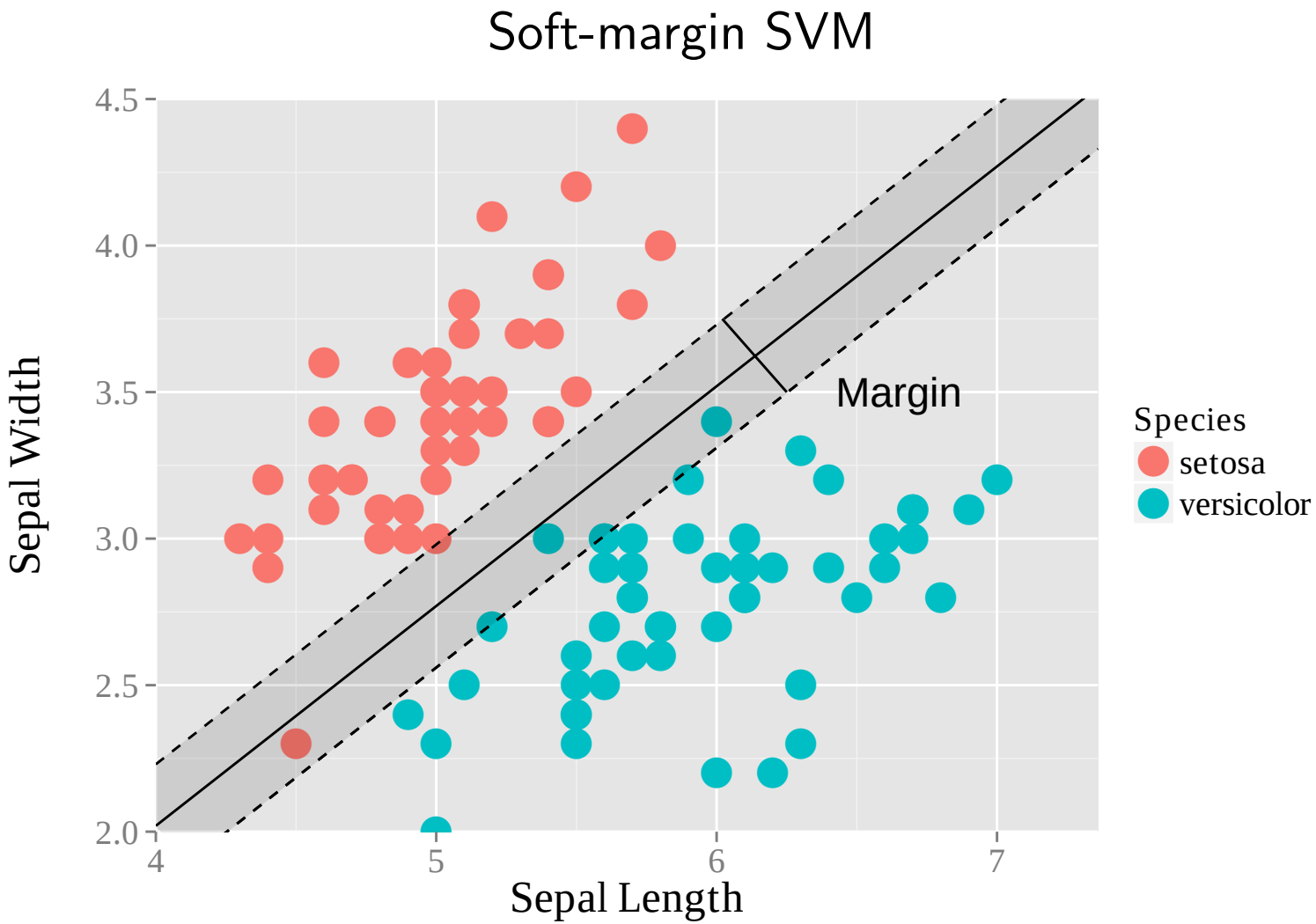
SVM in action



SVM in action

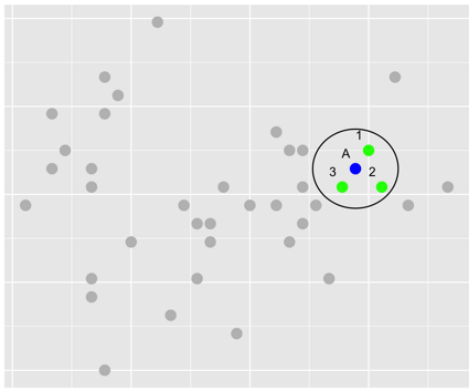


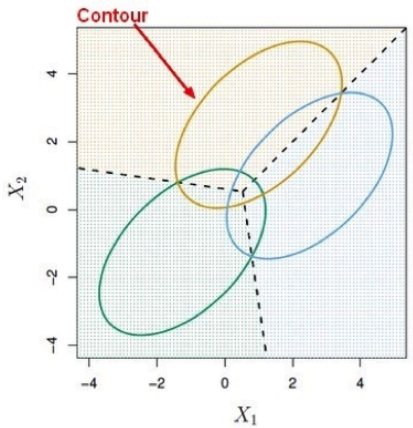
SVM in action



More Supervised Learning Methods

- ***K*-Nearest Neighbors**
  - Classify data points simply using the K closest points
- **Linear Discriminant Analysis**
  - Useful for classification when the groups are well-separated
- **Boosting methods**
  - Retrain the model to improve out-of-sample performance; for example, Additive Logistic Regression
- **Neural Networks**
  - `https://en.wikipedia.org/wiki/Artificial_neural_network`





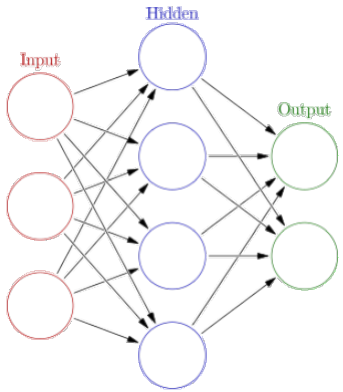
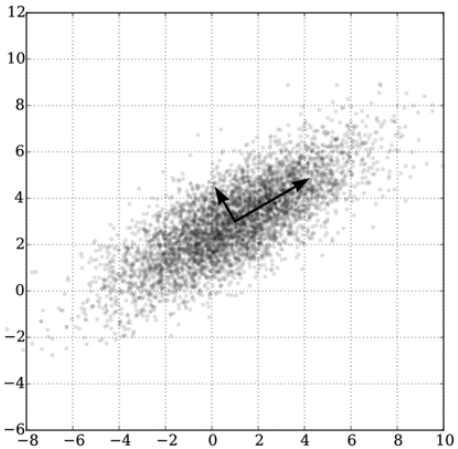


Figure: (From left to right) k-NN, LDA, 1-layer neural network

More Unsupervised Learning Methods

- **Principal Component Analysis**
  - Reduces the dimension of the feature matrix using matrix algebra and SVD
- **Imputation methods**
  - Fills in missing values; for example, the EM algorithm, k-NN impute
- **k-modes Clustering**
  - Clustering for survey data



$$\begin{bmatrix} x_{11} & x_{12} & ? & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & ? & ? & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \\ x_{51} & x_{52} & ? & x_{54} \\ x_{61} & x_{62} & x_{63} & x_{64} \\ x_{71} & x_{72} & ? & ? \\ x_{81} & ? & x_{83} & x_{84} \end{bmatrix} \Rightarrow \begin{bmatrix} x_{11} & x_{12} & w_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & w_{32} & w_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \\ x_{51} & x_{52} & w_{53} & x_{54} \\ x_{61} & x_{62} & x_{63} & x_{64} \\ x_{71} & x_{72} & w_{73} & w_{74} \\ x_{81} & w_{82} & x_{83} & x_{84} \end{bmatrix}$$

Figure: (From left to right) PCA, missing data imputation

Conclusions

- There are **tons** of machine learning methods out there.
- `R` has many useful open-source packages for machine learning.
  - `Python` also has many available in the `scikit-learn` library
- Use **cross-validation** for model selection!
- *The Elements of Statistical Learning* and *The Analytics Edge* are great textbooks on the subject.
- This is an active research area, so new ML algorithms are being developed too.
  - Including some by students in the ORC ☺

The End

- That's all folks!
- Special thanks to Jerry Kung and Allison O'Hair for previous course materials, and Phil Chodrow for extensive course feedback.
- Please fill out feedback forms!
- Any questions? Feel free to email `cpixton@mit.edu` or `cpawlows@mit.edu`