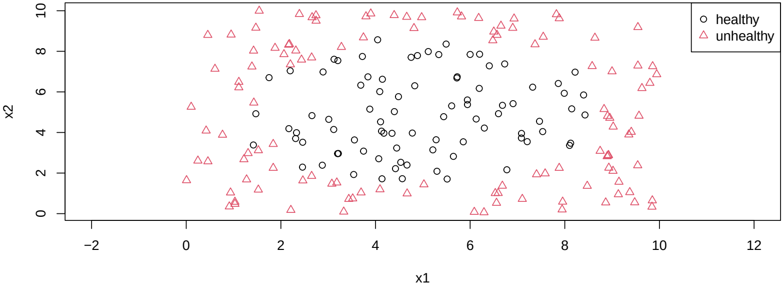
K-Nearest Neighbors and Decision Trees

Ryan Miller

Introduction

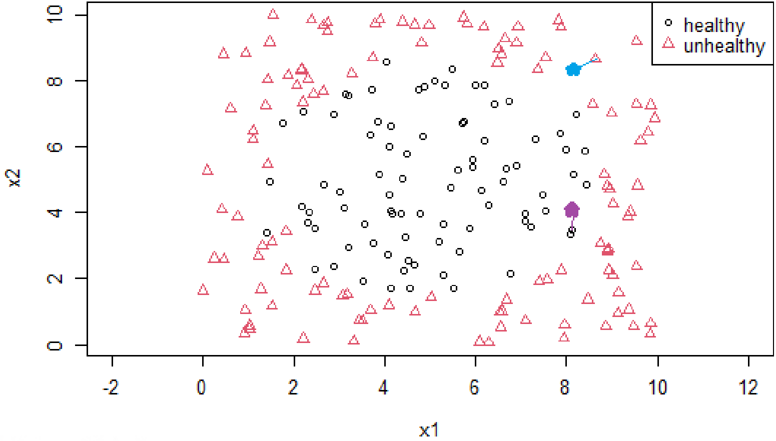
Last time we introduced toy data with a goal of classifying *new data-points* as "healthy" or "unhealthy" using patterns learned from the training data shown below:



Grinnell College
Statistics

K-Nearest Neighbors

A simple rule is to classify each new data-point using its *nearest neighbor*, or the observation closest to its (x_1, x_2) coordinates:



Grinnell College
Statistics

K-Nearest Neighbors

To implement this approach, we a method of identifying neighbors.

$$d_{a,b} = \left(\sum_{j=1}^{m} |x_{a,j} - x_{b,j}|^p\right)^{1/p}$$

- Minkowski distance, $d_{a,b}$, measures the distance between data-points a and b
  - The formula sums *pairwise coordinate differences* across m dimensions (features)
  - The power parameter, p, is chosen by the analyst, with $p = 2$ (euclidean distance) being a popular choice

K-Nearest Neighbors

Once neighbors are identified they must be used to make predictions. There are two common ways to do this:

1. **Uniform weighting** - all neighbors contribute equally, so if 4 of 5 neighbors of the new data-point are "healthy" the predicted probability of "healthy" is 80%
2. **Distance weighting** - neighbors are weighted by the inverse of their distance, allowing closer data-points to contribute more to the prediction (ie: a weighted proportion)

Note: when the outcome is numeric *KNN regression* averages the target variable among neighbors (rather than taking proportions)

Hyperparameters

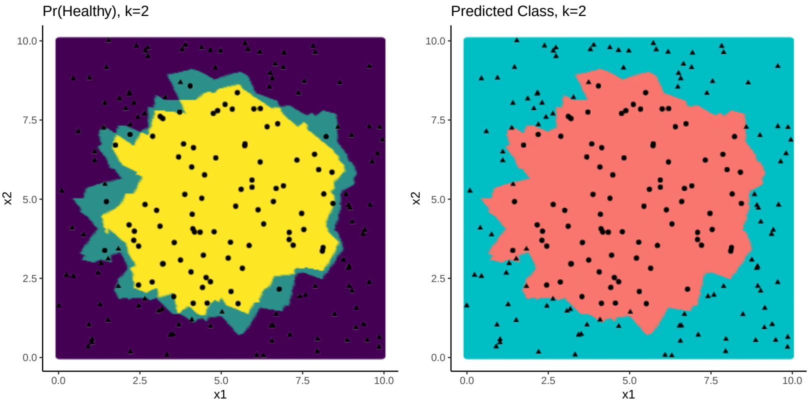
KNN will not achieve state-of-the-art performance in most applications, but it is an interesting algorithm to study because it illustrates two important ideas in machine learning:

1. **hyperparameters** - configurable values that must be set before the algorithm can be used
2. **pre-processing** - steps that must be applied to the data in order for the algorithm to be effective

We will discuss pre-processing in our next lecture. For now we'll focus on hyperparameters, which include:

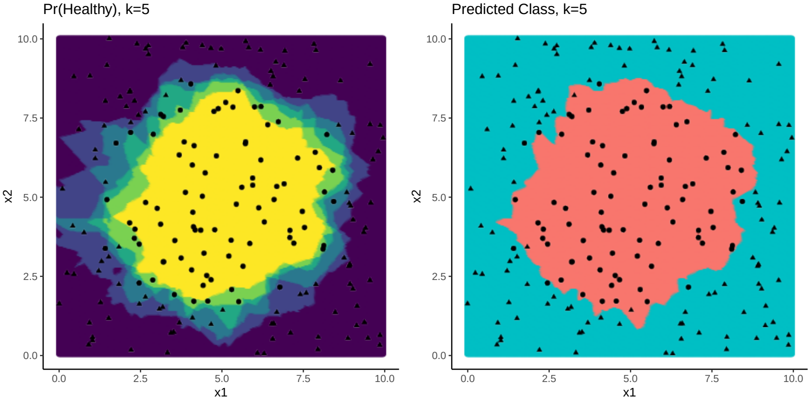
- k or `n_neighbors` - the number of neighbors that contribute to predictions
- p - the power parameter used in Minkowski distance calculations
- er *uniform* or *distance* weighting is used

K-Nearest Neighbors Prediction Surface (k=1)

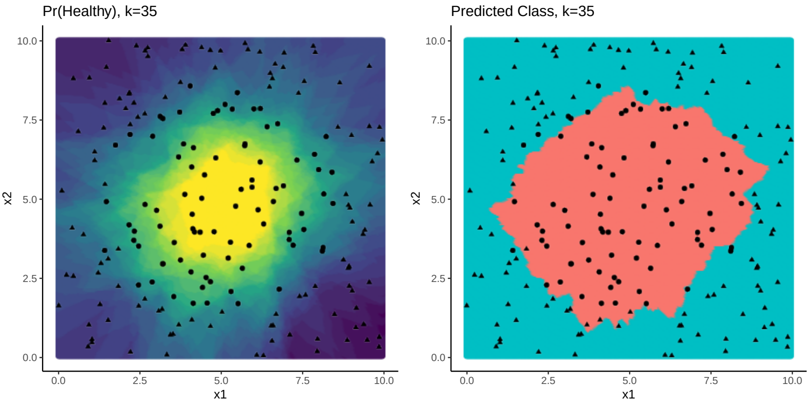


Grinnell College
Statistics

K-Nearest Neighbors Prediction Surface (k=5)



K-Nearest Neighbors Prediction Surface (k=35)



Grinnell College
Statistics

KNN and the Bias-Variance Trade-off

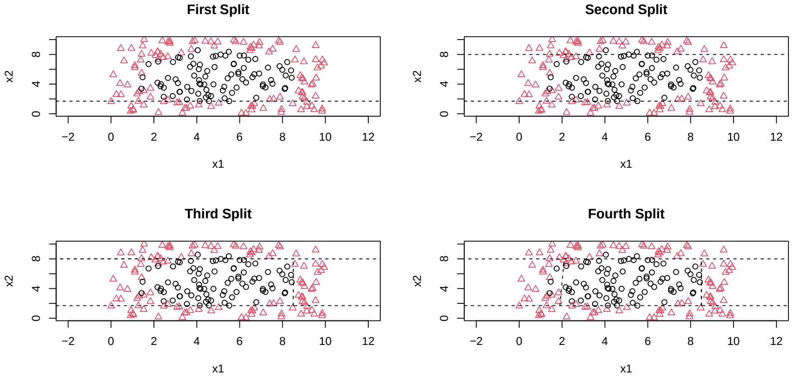
- Smaller values of k lead to more flexible models with *low bias* but *high variance*
- Conversely, larger values of k lead to less flexible models with smoother decision boundaries that are *higher in bias* but *lower in variance*
- Distance weighting generally produces a smoother decision boundary than uniform weighting. How might this impact the bias-variance trade off?

Decision Trees

Decision trees are trained by *recursively partitioning* the p-dimensional feature space (defined by the explanatory variables) until an acceptable level of homogeneity or "purity" is achieved within each partition:

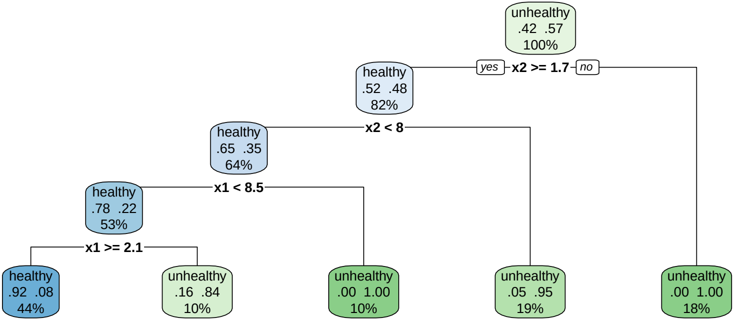
1) Starting with a "parent" node, search for a splitting rule that maximizes the *homogeneity* or *purity* of the "child" nodes
2) Next, considering each node that hasn't yet been split, find another splitting rule that maximizes *purity*
3) Repeat until a stopping criteria has been reached

Decision Trees



Decision Trees

We can express these recursive splits using a tree structure:



Determining the Splits

- The decision tree algorithm considers all possible splits for every feature
  - Only split-points that coincide with observed values are checked, as anything inbetween won't change purity
- Classification trees most often use **Gini impurity**:

$$Gini = \sum_{j=1}^{k} p_j(1-p_j) = 1 - \sum_{j=1}^{k} p_j^2$$

- For binary classification, Gini impurity reduces to $p_1(1-p_1) + p_2(1-p_2)$
  - The split that yields the greatest improvement in Gini impurity is selected
- Regression trees assess purity using *squared error*, or $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Determining the Splits

In our example the initial node's purity was:

$$(0.42 \cdot (1-0.42) + 0.58 \cdot (1-0.58)) = 0.487$$

Determining the Splits

In our example the initial node's purity was:

$$(0.42 \cdot (1 - 0.42) + 0.58 \cdot (1 - 0.58)) = 0.487$$

The first split created child nodes yielding the following purity:

$$0.82 \cdot \big(0.52 \cdot (1-0.52) + 0.48 \cdot (1-0.48)\big) + 0.18 \cdot \big(0 \cdot (1-0) + 1 \cdot (1-1)\big) = 0.409$$

Thus, the *Gini gain* from this split is 0.078

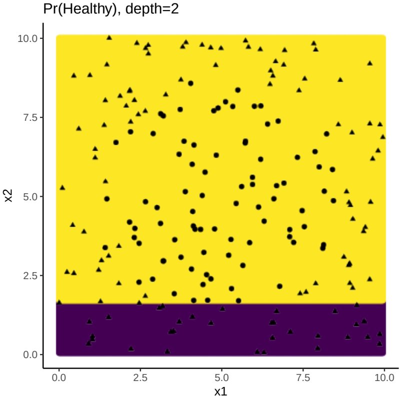
Stopping the Algorithm

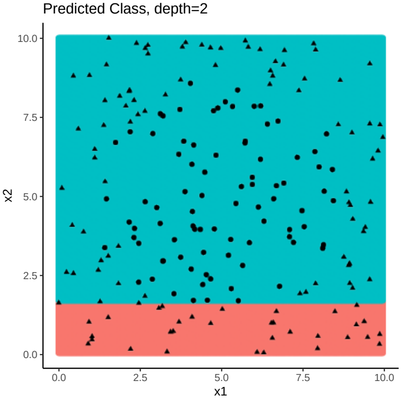
Decision trees can be grown until every terminal node is perfectly pure; however, such trees will be very overfit to the training data. We can exploit the bias-variance trade-off in a fitted tree in the following ways:

1. Restricting the maximum depth of the tree (ie: the number of sequential rules)
2. Allowing only nodes of sufficient size to be eligible for splitting
3. Requiring a certain improvement in purity for a split to occur

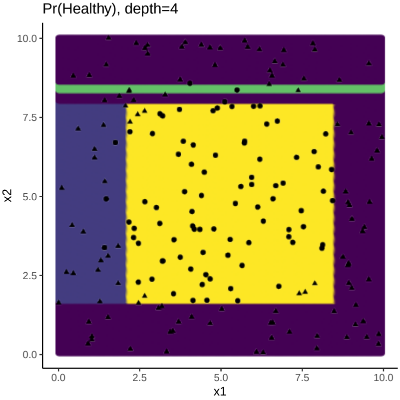
Because all of these are related, it is generally sufficient to focus on maximum depth when tuning hyperparameters.

Decision Tree Prediction Surface (`max_depth = 1`)

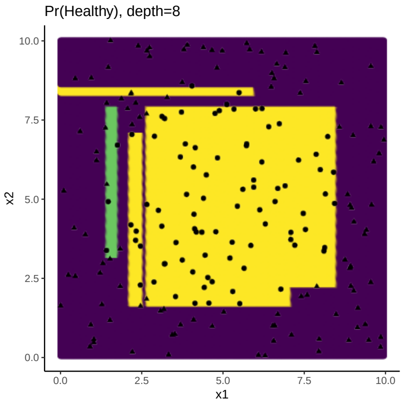




Decision Tree Prediction Surface (max_depth = 4)



Decision Tree Prediction Surface (`max_depth = 8`)



A Few Comments

- KNN produces irregularly shaped decision boundaries that tend to be overly sensitive to the training data for small values of k
- Decision trees produce rectangular decision boundaries and can easily overfit the training data if their maximum depth isn't controlled
- Decision trees are robust to the measurement scale of the predictive features, while KNN is not
  - We will discuss re-scaling (and other data pre-processing steps) next time

What to Know for Thursday's Quiz

- The basic steps of the KNN algorithm, including finding neighbors using distance calculations and generating predicted values using these neighbors
- The basic steps of the Decision Tree algorithm, including the concept of making recursive binary splits to improve purity
- The hyperparameters of the KNN and Decision Tree algorithms, and how each influences the bias-variance trade-off