$k$ Nearest Neighbors
k Nearest Neighbors

To classify an observation:

- Look at the labels of some number, say $k$, of neighboring observations.
- The observation is then classified based on its nearest neighbors’ labels.

Super simple idea!

Instance-based learning

- … as opposed to model-based (no pre-processing)
Example

- Let’s try to classify the unknown green point by looking at $k = 3$ and $k = 5$ nearest neighbors.
- For $k = 3$, we see 2 triangles and 1 square; so we classify the point as a triangle.
- For $k = 5$, we see 2 triangles and 3 squares; so we classify the point as a square.
- Typically, we use an odd value of $k$ to avoid ties.
Design Decisions

- Choose $k$
- Define “neighbor”
  - Define a measure of distance/closeness
  - Choose a threshold
- Decide how to classify based on neighbors’ labels
  - By a majority vote, or
  - By a weighted majority vote (weighted by distance), or …
Classifying *iris*

We’re going to demonstrate the use of $k$-NN on the *iris* data set (the flower, not the part of your eye)

<table>
<thead>
<tr>
<th></th>
<th>Sepal.Length</th>
<th>Sepal.Width</th>
<th>Petal.Length</th>
<th>Petal.Width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>2</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>3</td>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>4</td>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>5</td>
<td>5.0</td>
<td>3.6</td>
<td>1.2</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>6</td>
<td>5.4</td>
<td>3.9</td>
<td>1.7</td>
<td>0.4</td>
<td>setosa</td>
</tr>
<tr>
<td>7</td>
<td>4.6</td>
<td>3.4</td>
<td>1.4</td>
<td>0.3</td>
<td>setosa</td>
</tr>
<tr>
<td>8</td>
<td>5.0</td>
<td>3.4</td>
<td>1.5</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>9</td>
<td>4.4</td>
<td>2.9</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>10</td>
<td>4.9</td>
<td>3.1</td>
<td>1.5</td>
<td>0.1</td>
<td>setosa</td>
</tr>
<tr>
<td>11</td>
<td>5.4</td>
<td>3.7</td>
<td>1.5</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>12</td>
<td>4.8</td>
<td>3.4</td>
<td>1.6</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>13</td>
<td>4.8</td>
<td>3.0</td>
<td>1.4</td>
<td>0.1</td>
<td>setosa</td>
</tr>
<tr>
<td>14</td>
<td>4.3</td>
<td>3.0</td>
<td>1.1</td>
<td>0.1</td>
<td>setosa</td>
</tr>
<tr>
<td>15</td>
<td>5.8</td>
<td>4.0</td>
<td>1.2</td>
<td>0.2</td>
<td>setosa</td>
</tr>
</tbody>
</table>
iris

- 3 species (i.e., classes) of iris
  - Iris setosa
  - Iris versicolor
  - Iris virginica
- 50 observations per species
- 4 variables per observation
  - Sepal length
  - Sepal width
  - Petal length
  - Petal width
Visualizing the data
knn in R

- R provides a `knn` function in the `class` library

```r
> library(class) # Use classification package

> knn(train, test, cl, k)
```

- `train`: training data for the k-NN classifier
- `test`: testing data to classify
- `cl`: class labels
- `k`: the number of neighbors to use
A new observation

- New orange point

```r
new_point <- data.frame(
  Sepal.Length = 7.2,
  Sepal.Width  = 3.2,
  Petal.Length = 6.4,
  Petal.Width  = 2.4)
```

- Let’s run $k$-NN in R with $k = 3$
knn in R (cont’d)

> knn(data.frame(iris$Petal.Length, iris$Sepal.Length),
       data.frame(new_point$Petal.Length, new_point$Sepal.Length),
       iris$Species,
       k = 3)

[1] virginica
Levels: setosa versicolor virginica
How does it work?

- Since $k = 3$, R finds the new point’s closest 3 neighbors
- These are all virginica, so it’s easy to classify the new point
- The new observation also gets classified as virginica
Another observation

- This one’s a bit more ambiguous

```r
new_point <- data.frame(
  Sepal.Length = 6.4,
  Sepal.Width  = 2.8,
  Petal.Length = 4.9,
  Petal.Width  = 1.3)
```

- What should we expect?
Another observation

[1] $k = 3$
[1] versicolor
Another observation

[1] $k = 3$
[1] versicolor
[1] $k = 5$
[1] virginica
Another observation

[1] $k = 3$
[1] versicolor
[1] $k = 5$
[1] virginica
[1] $k = 11$
[1] versicolor
Decision Regions

- Decision regions are regions where observations are classified one way or another.
- For *iris*, there are three species, and three (approximate) decision regions.

![Decision Regions Diagram](image)
Decision Boundaries

- (Linear) decision boundaries are lines that separate decision regions.
- On the boundaries, classifiers may give ambiguous results.
- Changing parameters, like $k$ in $k$-NN, changes both the decision regions and the corresponding boundaries.
- Let’s vary $k$ and see what happens.
Decision Boundaries

- Let’s take a closer look at decision boundaries for $k$-NN
- To do so, let’s use a new data set
- Here are some random data, classified as either 0 or 1
- These two classes overlap quite a bit, compared to the iris data
$k = 1$
$k = 3$
$k = 7$
$k = 15$
$k = 25$
$k = 51$
\( k = 101 \)
Small $k$

- $k = 1$
- Models like this are overfit
- Reading too much into training data, extrapolating based on things that aren’t really there
Large $k$

- $k = 101$
- Rather than being overfit, this model is underfit
- The decision boundary doesn’t capture enough information encoded in the training data
Just right

\( k = 15 \)
3D and Beyond

- For visualization reasons, our examples were limited to only two features
- But $k$-NN is not limited to only two features
- Can also use 3, 4, ...., $n$ dimensions
- knn will use all available numeric dimensions
Working in 3D

knn(data.frame(iris$Petal.Length,
                iris$Sepal.Length,
                iris$Sepal.Width),
    data.frame(new_point$Petal.Length,
                new_point$Sepal.Length,
                new_point$Sepal.Width),
    iris$Species,
    k = 11)
k-NN caveats

- *k*-NN can be very slow, especially for very large data sets
  - *k*-NN is not a learning algorithm in the traditional sense, because it doesn’t actually do any learning: i.e., it doesn’t preprocess the data
  - Instead, when it is given a new observation, it calculates the distance between that observation and every existing observation in the data set
- *k*-NN works better with quantitative data than categorical data
  - Data must be quantitative to calculate distances
  - So qualitative data must be converted
- Without clusters in the training data, *k*-NN cannot work well
Model Selection

Find a model that appropriately balances complexity and generalization capabilities: i.e., that optimizes the bias-variance tradeoff.

- **High bias, low variance**
  - A high value of $k$ indicates a high degree of bias, but contains the variance

- **Low bias, high variance**
  - With low values of $k$, the very jagged decision boundaries are a sign of high variance