K-Nearest Neighbors

Prof Wells

STA 295: Stat Learning

February 20th, 2024

Outline

In today's class, we will. . .

- Implement KNN in R
- Discuss benefits and drawbacks of KNN

Section 1

K-Nearest Neighbors

KNN Review

KNN makes predictions on a test point x_0 by averaging the response value among K "nearby" points in the training set.

KNN Review

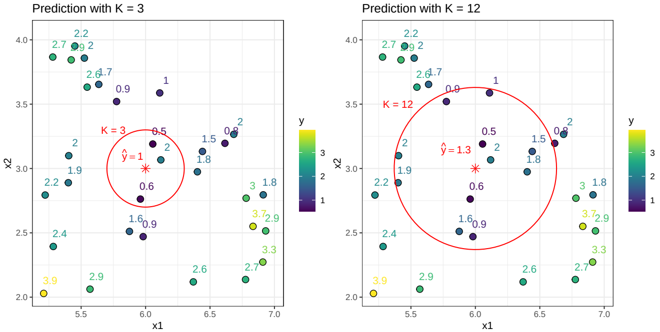
KNN makes predictions on a test point x_0 by averaging the response value among K "nearby" points in the training set.

$$\hat{y}_0 = \frac{1}{K} \sum_{i \in \mathcal{N}_0} y_i$$

KNN Review

KNN makes predictions on a test point x_0 by averaging the response value among K "nearby" points in the training set.

$$\hat{y}_0 = \frac{1}{K} \sum_{i \in \mathcal{N}_0} y_i$$



Effect of K on predictions

Different values of K lead to different estimates $\hat{y}$.

Effect of K on predictions

Different values of K lead to different estimates $\hat{y}$.

- Small K mean that we only look at closest or "most similar" points, which should have response values closest to the true response at the test point.

Effect of K on predictions

Different values of K lead to different estimates $\hat{y}$.

- Small K mean that we only look at closest or "most similar" points, which should have response values closest to the true response at the test point.
  - However, with only a small number of neighbors, predictions are likely to change significantly from training set to training set.

Effect of K on predictions

Different values of K lead to different estimates $\hat{y}$.

- Small K mean that we only look at closest or "most similar" points, which should have response values closest to the true response at the test point.
    - However, with only a small number of neighbors, predictions are likely to change significantly from training set to training set.
    - Small K yields a low **Bias**, high **Variance** model (*flexible model*)

Effect of K on predictions

Different values of K lead to different estimates $\hat{y}$.

- Small K mean that we only look at closest or "most similar" points, which should have response values closest to the true response at the test point.
  - However, with only a small number of neighbors, predictions are likely to change significantly from training set to training set.
  - Small K yields a low **Bias**, high **Variance** model (*flexible model*)
- Large K mean that we look points that are both close and distant; some of these points have responses that might not be close to the true response at the test point.

Effect of K on predictions

Different values of K lead to different estimates $\hat{y}$.

- Small K mean that we only look at closest or "most similar" points, which should have response values closest to the true response at the test point.
  - However, with only a small number of neighbors, predictions are likely to change significantly from training set to training set.
  - Small K yields a low **Bias**, high **Variance** model (*flexible model*)
- Large K mean that we look points that are both close and distant; some of these points have responses that might not be close to the true response at the test point.
  - However, by averaging across a large number of points, predictions will not change much between different training sets.

Effect of K on predictions

Different values of K lead to different estimates $\hat{y}$.

- Small K mean that we only look at closest or "most similar" points, which should have response values closest to the true response at the test point.
  - However, with only a small number of neighbors, predictions are likely to change significantly from training set to training set.
  - Small K yields a low **Bias**, high **Variance** model (*flexible model*)
- Large K mean that we look points that are both close and distant; some of these points have responses that might not be close to the true response at the test point.
  - However, by averaging across a large number of points, predictions will not change much between different training sets.
  - Large K yields a low **Variance**, high **Bias** model (*rigid model*)

Effect of K on predictions

Different values of K lead to different estimates $\hat{y}$.

- Small K mean that we only look at closest or "most similar" points, which should have response values closest to the true response at the test point.
  - However, with only a small number of neighbors, predictions are likely to change significantly from training set to training set.
  - Small K yields a low **Bias**, high **Variance** model (*flexible model*)
- Large K mean that we look points that are both close and distant; some of these points have responses that might not be close to the true response at the test point.
  - However, by averaging across a large number of points, predictions will not change much between different training sets.
  - Large K yields a low **Variance**, high **Bias** model (*rigid model*)
- In Ch. 5 (next week), we discuss methods for choosing optimal K

KNN in R

- In R, we have two options for KNN:
  - Use `knn` from `class` package (textbook's choice)
  - Use the `kknn` function from the `kknn` package (preferable)

KNN in R

- In R, we have two options for KNN:
  - Use `knn` from `class` package (textbook's choice)
  - Use the `kknn` function from the `kknn` package (preferable)
- Both `kknn` and `knn` fit a model **and** makes predictions all in one command.

KNN in R

- In R, we have two options for KNN:
  - Use `knn` from `class` package (textbook's choice)
  - Use the `kknn` function from the `kknn` package (preferable)
- Both `kknn` and `knn` fit a model **and** makes predictions all in one command.
  - Compare to linear models, which first fit a model using `lm` and then make predictions using `predict`

KNN in R

- In R, we have two options for KNN:
    - Use `knn` from `class` package (textbook's choice)
    - Use the `kknn` function from the `kknn` package (preferable)
- Both `kknn` and `knn` fit a model **and** makes predictions all in one command.
    - Compare to linear models, which first fit a model using `lm` and then make predictions using `predict`
- The `knn` function requires a model matrix (use `model.matrix`); while `kknn` instead uses a formula (`y ~.`)

KNN in R

- In R, we have two options for KNN:
  - Use `knn` from `class` package (textbook's choice)
  - Use the `kknn` function from the `kknn` package (preferable)
- Both `kknn` and `knn` fit a model **and** makes predictions all in one command.
  - Compare to linear models, which first fit a model using `lm` and then make predictions using `predict`
- The `knn` function requires a model matrix (use `model.matrix`); while `kknn` instead uses a formula (`y ~.`)
- `kknn` allows us to (optionally) weight observations **by** distance

KNN in R

- In R, we have two options for KNN:
    - Use `knn` from `class` package (textbook's choice)
    - Use the `kknn` function from the `kknn` package (preferable)
- Both `kknn` and `knn` fit a model **and** makes predictions all in one command.
    - Compare to linear models, which first fit a model using `lm` and then make predictions using `predict`
- The `knn` function requires a model matrix (use `model.matrix`); while `kknn` instead uses a formula (`y ~.`)
- `kknn` allows us to (optionally) weight observations **by** distance
- `kknn` also allows us to use different notions of distance (Euclidean, Manhattan, Minkowski, Hamming, and more)

Meet the Neighbors

The `GrinnellHouses` data set contains data on 929 houses sold between 2005 and 2015 in Grinnell, IA.

Meet the Neighbors

The `GrinnellHouses` data set contains data on 929 houses sold between 2005 and 2015 in Grinnell, IA.

- Data was collected by local real estate broker Matt Karjalahti, and tidied by Grinnell economists L. Logan and E. Ohrn

Meet the Neighbors

The `GrinnellHouses` data set contains data on 929 houses sold between 2005 and 2015 in Grinnell, IA.

- Data was collected by local real estate broker Matt Karjalahti, and tidied by Grinnell economists L. Logan and E. Ohrn

```
##                     Address Latitude Longitude Bedrooms Baths SquareFeet
## 1    1510 First Ave #112  41.73880 -92.71378        2     1       1120
## 2         1020 Center St  41.74558 -92.73168        3     1       1224
## 3      918 Chatterton St  41.74404 -92.71308        4     1       1540
## 4 1023 & 1025 Spring St.  41.74503 -92.72896        3     1       1154
## 5            503 2nd Ave  41.74041 -92.73002        3     1       1277
## 6          9090 Clay St  41.81942 -92.77381        3     1       1079
##     LotSize YearBuilt YearSold MonthSold DaySold OrigPrice ListPrice SalePrice
## 1        NA      1993     2005         9      16     17000     10500      7000
## 2 0.1721763      1900     2006         3      20     35000     35000     27000
## 3        NA      1970     2006         3      15     54000     47000     28000
## 4        NA      1900     2006         2       1     65000     49000     30000
## 5 0.2066116      1900     2005         8      19     35000     35000     30750
## 6 0.1993572      1900     2005         5      27     45900     45900     42000
##      Age
## 1 Modern
## 2    Old
## 3    Mid
## 4    Old
## 5    Old
```

KNN in Action!

- When you apply for a home mortgage loan from a bank, you offer the soon-to-by-purchased home as collateral.

KNN in Action!

- When you apply for a home mortgage loan from a bank, you offer the soon-to-by-purchased home as collateral.
- To ensure this asset can cover the loan amount in event that the loan is not repaid, the bank contracts a 3rd party *Assessor* to value the home.

KNN in Action!

- When you apply for a home mortgage loan from a bank, you offer the soon-to-by-purchased home as collateral.
- To ensure this asset can cover the loan amount in event that the loan is not repaid, the bank contracts a 3rd party *Assessor* to value the home.
  - The assessor records features of the home (Year Built, Year Sold, Location, Sq. Footage, Bedrooms, Bathrooms, etc.) and then creates a list of **comps**: houses with comparable features that have been recently sold.

KNN in Action!

- When you apply for a home mortgage loan from a bank, you offer the soon-to-by-purchased home as collateral.
- To ensure this asset can cover the loan amount in event that the loan is not repaid, the bank contracts a 3rd party *Assessor* to value the home.
  - The assessor records features of the home (Year Built, Year Sold, Location, Sq. Footage, Bedrooms, Bathrooms, etc.) and then creates a list of **comps**: houses with comparable features that have been recently sold.
  - The assessor assigns a value to the home based on the (weighted) average of the sale price of these comps.

KNN in Action!

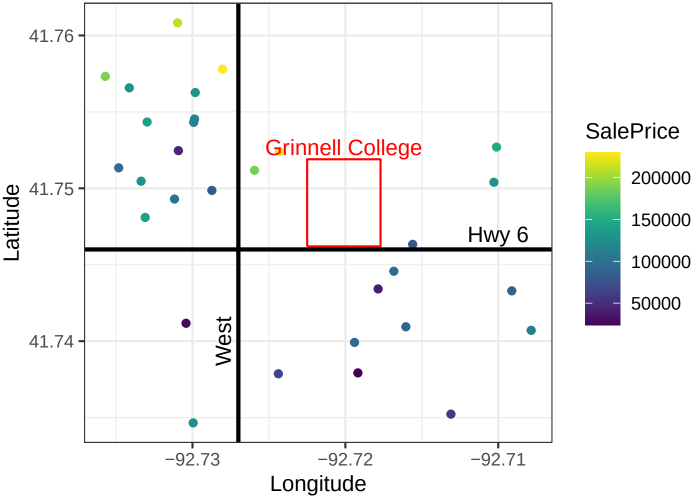
- When you apply for a home mortgage loan from a bank, you offer the soon-to-by-purchased home as collateral.
- To ensure this asset can cover the loan amount in event that the loan is not repaid, the bank contracts a 3rd party *Assessor* to value the home.
  - The assessor records features of the home (Year Built, Year Sold, Location, Sq. Footage, Bedrooms, Bathrooms, etc.) and then creates a list of **comps**: houses with comparable features that have been recently sold.
  - The assessor assigns a value to the home based on the (weighted) average of the sale price of these comps.
  - The assessor uses the fact that the “value” of the home is exactly what people are willing to pay for similar homes in the same market.

KNN in Action!

- When you apply for a home mortgage loan from a bank, you offer the soon-to-by-purchased home as collateral.
- To ensure this asset can cover the loan amount in event that the loan is not repaid, the bank contracts a 3rd party *Assessor* to value the home.
    - The assessor records features of the home (Year Built, Year Sold, Location, Sq. Footage, Bedrooms, Bathrooms, etc.) and then creates a list of **comps**: houses with comparable features that have been recently sold.
    - The assessor assigns a value to the home based on the (weighted) average of the sale price of these comps.
    - The assessor uses the fact that the "value" of the home is exactly what people are willing to pay for similar homes in the same market.
- But this is **exactly** the KNN algorithm.

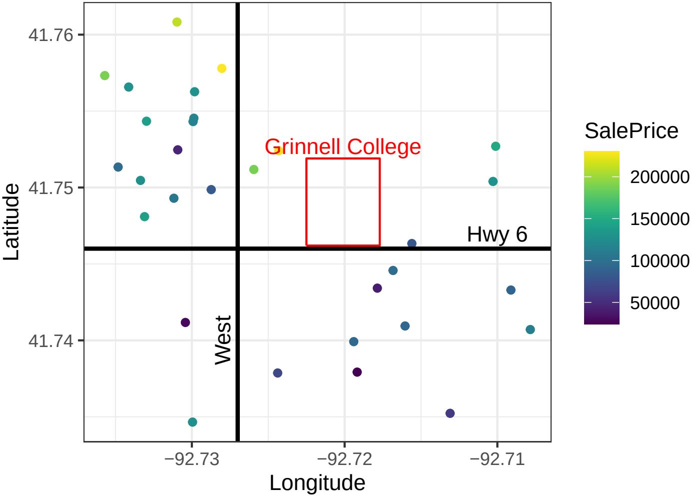
Location, Location, Location

Let's predict house price, based on Latitude (N-S) and Longitude (E-W)



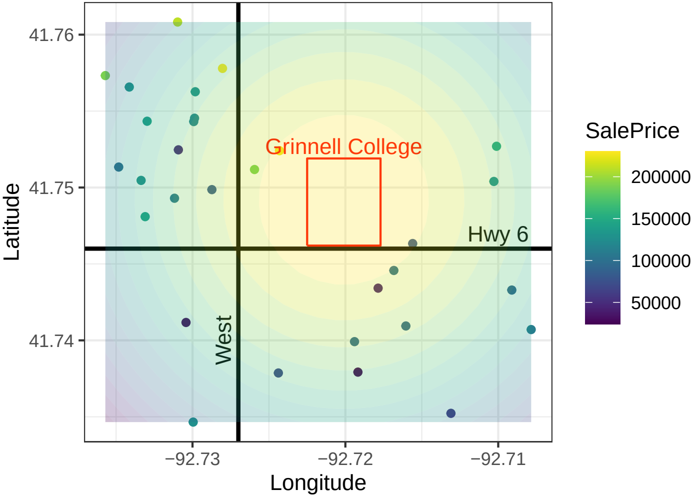
Location, Location, Location

Why is a multilinear model potentially a poor choice?



Location, Location, Location

How well would the linear model do if homes closest to the college have the highest price?



Building the KNN "model"

- We first divide our data set into training and test components:

Building the KNN "model"

- We first divide our data set into training and test components:

```
set.seed(10) #Allows for reproducibile random numbers

n <- nrow(GrinnellHouses) #number of observations
prop <- 0.7 #proportion in the training set
train_size <- round(n*prop) #rounded number in training set

train_indices <- sample(1:n, size = train_size, replace = F)
#creates list of indices to include in training

head(train_indices) #an example of a few indices in training set
```

```
## [1] 491 649 330 368 460 439
```

```
GrinnellHouses_train <- GrinnellHouses[train_indices, ]
#subsets for training set by training indices

GrinnellHouses_test <- GrinnellHouses[-train_indices, ]
#subsets all observations not in training set
```

KNN predictions

- Now let's predict on the test set for variety of values of k

KNN predictions

- Now let's predict on the test set for variety of values of k

```r
library(kknn)

House_fit3 <- kknn(SalePrice ~ Longitude + Latitude,
                   train = GrinnellHouses_train,
                   test = GrinnellHouses_test,
                     k = 3, kernel = "rectangular")

House_fit5 <- kknn(SalePrice ~ Longitude + Latitude,
                   train = GrinnellHouses_train,
                   test = GrinnellHouses_test,
                     k = 5, kernel = "rectangular")

House_fit10 <- kknn(SalePrice ~ Longitude + Latitude,
                    train = GrinnellHouses_train,
                    test = GrinnellHouses_test,
                     k = 10, kernel = "rectangular")
```

KNN predictions

- Now let's predict on the test set for variety of values of k

```
library(kknn)

House_fit3 <- kknn(SalePrice ~ Longitude + Latitude,
                   train = GrinnellHouses_train,
                   test = GrinnellHouses_test,
                     k = 3, kernel = "rectangular")

House_fit5 <- kknn(SalePrice ~ Longitude + Latitude,
                   train = GrinnellHouses_train,
                   test = GrinnellHouses_test,
                     k = 5, kernel = "rectangular")

House_fit10 <- kknn(SalePrice ~ Longitude + Latitude,
                    train = GrinnellHouses_train,
                    test = GrinnellHouses_test,
                     k = 10, kernel = "rectangular")
```

- Setting `kernel = "rectangular"` corresponds to classic KNN (we'll talk about other options later)

Comparing Predictions

Here are some of the predicted and actual values for each of the models

- We access the predictions from the model using `$fitted.values`

Comparing Predictions

Here are some of the predicted and actual values for each of the models

- We access the predictions from the model using `$fitted.values`

	actual_price	knn1	knn3	knn5	knn10	knn30	lin_model
675	114900	77500	99833.33	106700	90600.0	177526.67	132099.2
378	70000	36000	66500.00	63200	71091.9	96097.30	132669.8
686	125000	65500	64666.67	99000	102000.0	91527.30	132674.3
15	58000	124500	131666.67	136900	121356.0	117745.33	132619.4
329	142000	68000	82000.00	85550	79185.0	91638.33	132666.5
23	72000	95000	120000.00	120400	120300.0	118950.00	132653.0
616	189500	285000	198333.33	214000	201200.0	185926.67	133016.0
496	115500	112000	120666.67	117500	135150.0	131875.00	132641.3
869	105000	111000	73000.00	85850	81041.9	97313.97	132666.4
243	268000	125900	143600.00	141660	167055.0	226237.23	132614.5

Comparing Predictions

Here are some of the predicted and actual values for each of the models

- We access the predictions from the model using `$fitted.values`

	actual_price	knn1	knn3	knn5	knn10	knn30	lin_model
675	114900	77500	99833.33	106700	90600.0	177526.67	132099.2
378	70000	36000	66500.00	63200	71091.9	96097.30	132669.8
686	125000	65500	64666.67	99000	102000.0	91527.30	132674.3
15	58000	124500	131666.67	136900	121356.0	117745.33	132619.4
329	142000	68000	82000.00	85550	79185.0	91638.33	132666.5
23	72000	95000	120000.00	120400	120300.0	118950.00	132653.0
616	189500	285000	198333.33	214000	201200.0	185926.67	133016.0
496	115500	112000	120666.67	117500	135150.0	131875.00	132641.3
869	105000	111000	73000.00	85850	81041.9	97313.97	132666.4
243	268000	125900	143600.00	141660	167055.0	226237.23	132614.5

- Which model performed best?

Comparing Predictions

Here are some of the predicted and actual values for each of the models

- We access the predictions from the model using `$fitted.values`

	actual_price	knn1	knn3	knn5	knn10	knn30	lin_model
675	114900	77500	99833.33	106700	90600.0	177526.67	132099.2
378	70000	36000	66500.00	63200	71091.9	96097.30	132669.8
686	125000	65500	64666.67	99000	102000.0	91527.30	132674.3
15	58000	124500	131666.67	136900	121356.0	117745.33	132619.4
329	142000	68000	82000.00	85550	79185.0	91638.33	132666.5
23	72000	95000	120000.00	120400	120300.0	118950.00	132653.0
616	189500	285000	198333.33	214000	201200.0	185926.67	133016.0
496	115500	112000	120666.67	117500	135150.0	131875.00	132641.3
869	105000	111000	73000.00	85850	81041.9	97313.97	132666.4
243	268000	125900	143600.00	141660	167055.0	226237.23	132614.5

- Which model performed best?

metric	knn1	knn3	knn5	knn10	knn30	lin_model
RMSE	62850.24	58005.88	57727.92	57129.2	61405.59	79479

Reflections

- Why do we think the $K = 10$ modeled outperformed the $K = 3$ model? Why did it outperform the $K = 30$ model?

Reflections

- Why do we think the $K = 10$ modeled outperformed the $K = 3$ model? Why did it outperform the $K = 30$ model?
  - $K = 3$ was too flexible (high variance, low bias), estimates were susceptible to individual irregularity in prices
  - i.e. house next door with flooded basement tanked the sale price

Reflections

- Why do we think the $K = 10$ modeled outperformed the $K = 3$ model? Why did it outperform the $K = 30$ model?
  - $K = 3$ was too flexible (high variance, low bias), estimates were susceptible to individual irregularity in prices
  - i.e. house next door with flooded basement tanked the sale price
  - $K = 30$ was too rigid (high bias, low variance), estimates relied too heavily on dissimilar houses
  - i.e. houses across town in entirely different neighborhood were used for *comps*

Reflections

- Why do we think the $K = 10$ modeled outperformed the $K = 3$ model? Why did it outperform the $K = 30$ model?
  - $K = 3$ was too flexible (high variance, low bias), estimates were susceptible to individual irregularity in prices
  - i.e. house next door with flooded basement tanked the sale price
  - $K = 30$ was too rigid (high bias, low variance), estimates relied too heavily on dissimilar houses
  - i.e. houses across town in entirely different neighborhood were used for *comps*
- What are some possible fixes to improve model accuracy?

Reflections

- Why do we think the $K = 10$ modeled outperformed the $K = 3$ model? Why did it outperform the $K = 30$ model?
  - $K = 3$ was too flexible (high variance, low bias), estimates were susceptible to individual irregularity in prices
  - i.e. house next door with flooded basement tanked the sale price
  - $K = 30$ was too rigid (high bias, low variance), estimates relied too heavily on dissimilar houses
  - i.e. houses across town in entirely different neighborhood were used for *comps*
- What are some possible fixes to improve model accuracy?
  - Include more variables: date sold, date built, bedrooms, bathrooms, square footage, etc.

Reflections

- Why do we think the $K = 10$ modeled outperformed the $K = 3$ model? Why did it outperform the $K = 30$ model?
  - $K = 3$ was too flexible (high variance, low bias), estimates were susceptible to individual irregularity in prices
  - i.e. house next door with flooded basement tanked the sale price
  - $K = 30$ was too rigid (high bias, low variance), estimates relied too heavily on dissimilar houses
  - i.e. houses across town in entirely different neighborhood were used for *comps*
- What are some possible fixes to improve model accuracy?
  - Include more variables: date sold, date built, bedrooms, bathrooms, square footage, etc.
- What is one problem with including more variables?

Reflections

- Why do we think the $K = 10$ modeled outperformed the $K = 3$ model? Why did it outperform the $K = 30$ model?
    - $K = 3$ was too flexible (high variance, low bias), estimates were susceptible to individual irregularity in prices
    - i.e. house next door with flooded basement tanked the sale price
    - $K = 30$ was too rigid (high bias, low variance), estimates relied too heavily on dissimilar houses
    - i.e. houses across town in entirely different neighborhood were used for *comps*
- What are some possible fixes to improve model accuracy?
    - Include more variables: date sold, date built, bedrooms, bathrooms, square footage, etc.
- What is one problem with including more variables?
    - There are fewer "similar" houses nearby

Reflections

- Why do we think the $K = 10$ modeled outperformed the $K = 3$ model? Why did it outperform the $K = 30$ model?
  - $K = 3$ was too flexible (high variance, low bias), estimates were susceptible to individual irregularity in prices
  - i.e. house next door with flooded basement tanked the sale price
  - $K = 30$ was too rigid (high bias, low variance), estimates relied too heavily on dissimilar houses
  - i.e. houses across town in entirely different neighborhood were used for *comps*
- What are some possible fixes to improve model accuracy?
  - Include more variables: date sold, date built, bedrooms, bathrooms, square footage, etc.
- What is one problem with including more variables?
  - There are fewer "similar" houses nearby
  - Not clear how to assess "closeness" if predictors are all on different scales (i.e. *Lat* / *Long* are in angular degrees, but *Year Sold* is in years)

Section 2

KNN Considerations

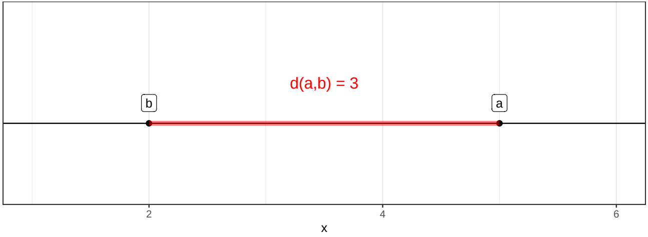
Quantifying Distance: Euclidean

When using Latitude and Longitude as predictors, it made sense to define "closeness" of points based on our every-day notion of distance. This distance is called the **Euclidean Distance**

Quantifying Distance: Euclidean

When using Latitude and Longitude as predictors, it made sense to define "closeness" of points based on our every-day notion of distance. This distance is called the **Euclidean Distance**

- In 1 dimension (numbers on a line), the Euclidean Distance between two numbers a and b is their absolute value $d(a, b) = |a - b|$



Quantifying Distance: Euclidean

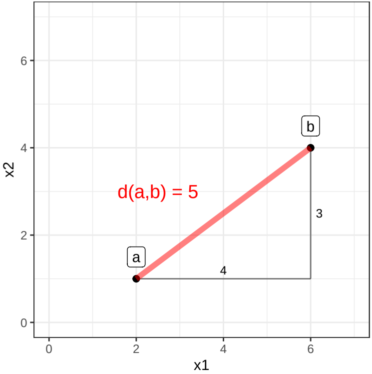
- In 2 dimensions (points on a plane), the Euclidean Distance between two points $\mathbf{a} = (a_1, a_2)$ and $\mathbf{b} = (b_1, b_2)$ is given by the Pythagorean Theorem:

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

Quantifying Distance: Euclidean

- In 2 dimensions (points on a plane), the Euclidean Distance between two points $\mathbf{a} = (a_1, a_2)$ and $\mathbf{b} = (b_1, b_2)$ is given by the Pythagorean Theorem:

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$



Quantifying Distance: Euclidean

- In p-dimensions, the Euclidean Distance between two points $\mathbf{a} = (a_1, a_2, \ldots, a_p)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_p)$ is given by the multidimensioanl Pythagorean Theorem:

Quantifying Distance: Euclidean

- In p-dimensions, the Euclidean Distance between two points $\mathbf{a} = (a_1, a_2, \ldots, a_p)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_p)$ is given by the multidimensioanl Pythagorean Theorem:

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_p - b_p)^2} = \sqrt{\sum_{i=1}^{p} (a_i - b_i)^2}$$

Quantifying Distance: Euclidean

- In p-dimensions, the Euclidean Distance between two points $\mathbf{a} = (a_1, a_2, \ldots, a_p)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_p)$ is given by the multidimensioanl Pythagorean Theorem:

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_p - b_p)^2} = \sqrt{\sum_{i=1}^{p} (a_i - b_i)^2}$$

- For example, suppose $\mathbf{a} = (1, 0, 3)$ and $\mathbf{b} = (-1, 2, 2)$. Then $d(\mathbf{a}, \mathbf{b}) =$

Quantifying Distance: Euclidean

- In p-dimensions, the Euclidean Distance between two points $\mathbf{a} = (a_1, a_2, \ldots, a_p)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_p)$ is given by the multidimensioanl Pythagorean Theorem:

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_p - b_p)^2} = \sqrt{\sum_{i=1}^{p} (a_i - b_i)^2}$$

- For example, suppose $\mathbf{a} = (1, 0, 3)$ and $\mathbf{b} = (-1, 2, 2)$. Then $d(\mathbf{a}, \mathbf{b}) =$

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{(1 - (-1))^2 + (0 - 2)^2 + (3 - 2)^2} = \sqrt{2^2 + 2^2 + 1^2} = 3$$

Quantifying Distance: Manhattan

An alternative notion of distance is called the **Manhattan Distance**

Quantifying Distance: Manhattan

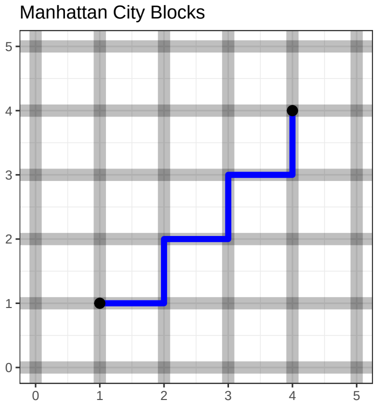
An alternative notion of distance is called the **Manhattan Distance**

- The name stems from how you might measure distance when driving around the city of Manhattan, where you can only travel along city blocks (but not through buildings)

Quantifying Distance: Manhattan

An alternative notion of distance is called the **Manhattan Distance**

- The name stems from how you might measure distance when driving around the city of Manhattan, where you can only travel along city blocks (but not through buildings)



Quantifying Distance: Manhattan

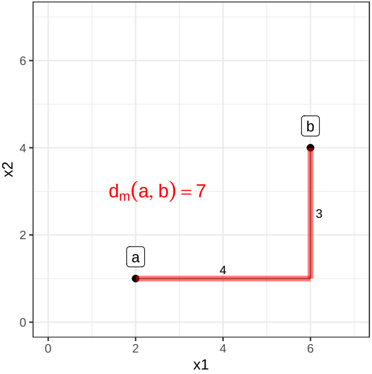
- In 2 dimensions, the Manhattan Distance between two points $\mathbf{a} = (a_1, a_2)$ and $\mathbf{b} = (b_1, b_2)$ is the sum of the absolute distances between their coordinates

$$d_m(\mathbf{a}, \mathbf{b}) = |a_1 - b_1| + |a_2 - b_2|$$

Quantifying Distance: Manhattan

- In 2 dimensions, the Manhattan Distance between two points $\mathbf{a} = (a_1, a_2)$ and $\mathbf{b} = (b_1, b_2)$ is the sum of the absolute distances between their coordinates

$$d_m(\mathbf{a}, \mathbf{b}) = |a_1 - b_1| + |a_2 - b_2|$$



Quantifying Distance: Euclidean

- In p-dimensions, the Manhattan Distance between two points $\mathbf{a} = (a_1, a_2, \ldots, a_p)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_p)$ is given by sum of coordinate-wise absolute values:

Quantifying Distance: Euclidean

- In p-dimensions, the Manhattan Distance between two points $\mathbf{a} = (a_1, a_2, \ldots, a_p)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_p)$ is given by sum of coordinate-wise absolute values:

$$d_m(\mathbf{a}, \mathbf{b}) = |a_1 - b_1| + |a_2 - b_2| + \cdots + |a_p - b_p| = \sum_{i=1}^{p} |a_i - b_i|$$

Quantifying Distance: Euclidean

- In p-dimensions, the Manhattan Distance between two points $\mathbf{a} = (a_1, a_2, \ldots, a_p)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_p)$ is given by sum of coordinate-wise absolute values:

$$d_m(\mathbf{a}, \mathbf{b}) = |a_1 - b_1| + |a_2 - b_2| + \cdots + |a_p - b_p| = \sum_{i=1}^{p} |a_i - b_i|$$

- For example, suppose $\mathbf{a} = (1, 0, 3)$ and $\mathbf{b} = (-1, 2, 2)$. Then $d_m(\mathbf{a}, \mathbf{b}) =$

Quantifying Distance: Euclidean

- In p-dimensions, the Manhattan Distance between two points $\mathbf{a} = (a_1, a_2, \ldots, a_p)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_p)$ is given by sum of coordinate-wise absolute values:

$$d_m(\mathbf{a}, \mathbf{b}) = |a_1 - b_1| + |a_2 - b_2| + \cdots + |a_p - b_p| = \sum_{i=1}^{p} |a_i - b_i|$$

- For example, suppose $\mathbf{a} = (1, 0, 3)$ and $\mathbf{b} = (-1, 2, 2)$. Then $d_m(\mathbf{a}, \mathbf{b}) =$

$$d_m(\mathbf{a}, \mathbf{b}) = |1 - (-1)| + |0 - 2| + |3 - 2| = 2 + 2 + 1 = 5$$

Distances in KNN

When implementing KNN, we choose a distance metric to use to determine "closeness"

Distances in KNN

When implementing KNN, we choose a distance metric to use to determine “closeness”

```
kknn(y~., distance = 2, ...) #Euclidean Distance

kknn(y~., distance = 1, ...) #Manhattan Distance
```

Distances in KNN

When implementing KNN, we choose a distance metric to use to determine "closeness"

```
kknn(y~., distance = 2, ...) #Euclidean Distance

kknn(y~., distance = 1, ...) #Manhattan Distance
```

- We should use **Euclidean distance** if most (or all) predictors measure the same type of thing (i.e. predictors are latitude and longitude)

Distances in KNN

When implementing KNN, we choose a distance metric to use to determine "closeness"

```
kknn(y~., distance = 2, ...) #Euclidean Distance

kknn(y~., distance = 1, ...) #Manhattan Distance
```

- We should use **Euclidean distance** if most (or all) predictors measure the same type of thing (i.e. predictors are latitude and longitude)
- We might use **Manhattan distance** if predictors are incomparable (i.e. number of rooms, year built, whether house has AC); or if predictors are binary.

Distances in KNN

When implementing KNN, we choose a distance metric to use to determine "closeness"

```
kknn(y~., distance = 2, ...) #Euclidean Distance

kknn(y~., distance = 1, ...) #Manhattan Distance
```

- We should use **Euclidean distance** if most (or all) predictors measure the same type of thing (i.e. predictors are latitude and longitude)
- We might use **Manhattan distance** if predictors are incomparable (i.e. number of rooms, year built, whether house has AC); or if predictors are binary.
- The choice of distance function can radically alter predictions.
  - i.e. points that are close in Euclidean distance might not be close in Manhattan distance.

Standardization

Suppose we have two predictors X_1 and X_2, where the standard deviation of X_1 is 10, while the standard deviation of X_2 is 1000.

Standardization

Suppose we have two predictors X_1 and X_2, where the standard deviation of X_1 is 10, while the standard deviation of X_2 is 1000.

- When using KNN, which of X_1 or X_2 will be *more influential* in prediction?

Standardization

Suppose we have two predictors X_1 and X_2, where the standard deviation of X_1 is 10, while the standard deviation of X_2 is 1000.

- When using KNN, which of X_1 or X_2 will be *more influential* in prediction?
- X_2 will be far more influential!
  - The distance between points is the sum of the distances in each predictor

Standardization

Suppose we have two predictors X_1 and X_2, where the standard deviation of X_1 is 10, while the standard deviation of X_2 is 1000.

- When using KNN, which of X_1 or X_2 will be *more influential* in prediction?
- X_2 will be far more influential!
  - The distance between points is the sum of the distances in each predictor
  - The typical distance between the X_2 values is in the 1000s, while the typical distance between the X_1 values is in the 10s.

Standardization

Suppose we have two predictors X_1 and X_2, where the standard deviation of X_1 is 10, while the standard deviation of X_2 is 1000.

- When using KNN, which of X_1 or X_2 will be *more influential* in prediction?
- X_2 will be far more influential!
  - The distance between points is the sum of the distances in each predictor
  - The typical distance between the X_2 values is in the 1000s, while the typical distance between the X_1 values is in the 10s.
  - Points will be nearby if they have similar X_2 values. The X_1 value will be irrelevant (since typical distances in X_1 are so much smaller than typical distances in X_2)

Standardization

Suppose we have two predictors X_1 and X_2, where the standard deviation of X_1 is 10, while the standard deviation of X_2 is 1000.

- When using KNN, which of X_1 or X_2 will be *more influential* in prediction?
- X_2 will be far more influential!
  - The distance between points is the sum of the distances in each predictor
  - The typical distance between the X_2 values is in the 1000s, while the typical distance between the X_1 values is in the 10s.
  - Points will be nearby if they have similar X_2 values. The X_1 value will be irrelevant (since typical distances in X_1 are so much smaller than typical distances in X_2)
- To fix this problem, we standardize all predictors so they have mean 0 and standard deviation 1:

Standardization

Suppose we have two predictors X_1 and X_2, where the standard deviation of X_1 is 10, while the standard deviation of X_2 is 1000.

- When using KNN, which of X_1 or X_2 will be *more influential* in prediction?
- X_2 will be far more influential!
  - The distance between points is the sum of the distances in each predictor
  - The typical distance between the X_2 values is in the 1000s, while the typical distance between the X_1 values is in the 10s.
  - Points will be nearby if they have similar X_2 values. The X_1 value will be irrelevant (since typical distances in X_1 are so much smaller than typical distances in X_2)
- To fix this problem, we standardize all predictors so they have mean 0 and standard deviation 1:

$$X_j^{\text{std}} = \frac{X_j - \bar{X}_j}{\sigma_{X_j}}$$

Standardization

Suppose we have two predictors X_1 and X_2, where the standard deviation of X_1 is 10, while the standard deviation of X_2 is 1000.

- When using KNN, which of X_1 or X_2 will be *more influential* in prediction?
- X_2 will be far more influential!
  - The distance between points is the sum of the distances in each predictor
  - The typical distance between the X_2 values is in the 1000s, while the typical distance between the X_1 values is in the 10s.
  - Points will be nearby if they have similar X_2 values. The X_1 value will be irrelevant (since typical distances in X_1 are so much smaller than typical distances in X_2)
- To fix this problem, we standardize all predictors so they have mean 0 and standard deviation 1:

$$X_j^{\mathrm{std}} = \frac{X_j - \bar{X}_j}{\sigma_{X_j}}$$

- In `kknn`, predictors are automatically standardized before predictions are made.

House Prices Redux

Let's use $K = 3, 5, 10, 30$ to make predictions for sale price using year of sale, house age (Old, Mid, Modern), and location.

- We use Manhattan distance, since predictors are incomparable

```
## Example Code
House_fit3_more <- kknn(SalePrice ~ Longitude + Latitude + Age + YearSold,
                   train = GrinnellHouses_train,
                   test = GrinnellHouses_test,
                     k = 3, kernel = "rectangular", distance = 1)


house_lm_more <- lm(SalePrice ~ Longitude + Latitude + Age + YearSold,
                    data = GrinnellHouses_train)

House_fit_lm_more <- predict(house_lm_more, newdata = GrinnellHouses_test)
```

Results

Some predictions:

Results

Some predictions:

	actual_price	knn1	knn3	knn5	knn10	knn30	lin_model
170	93000	94000	78666.67	79200	85440	103946.7	93144.97
567	294500	75000	73200.00	97520	127160	142296.7	189978.10
901	169500	148000	136666.67	171800	157250	134071.7	131516.80
388	85000	61000	53166.67	87700	89100	89900.0	96851.49

- Which model performed best?

Results

Some predictions:

	actual_price	knn1	knn3	knn5	knn10	knn30	lin_model
170	93000	94000	78666.67	79200	85440	103946.7	93144.97
567	294500	75000	73200.00	97520	127160	142296.7	189978.10
901	169500	148000	136666.67	171800	157250	134071.7	131516.80
388	85000	61000	53166.67	87700	89100	89900.0	96851.49

- Which model performed best?

metric	knn1	knn3	knn5	knn10	knn30	lin_model
RMSE	79113.17	64101.93	64291.1	65431.08	65288.48	67496.36

Results

Some predictions:

	actual_price	knn1	knn3	knn5	knn10	knn30	lin_model
170	93000	94000	78666.67	79200	85440	103946.7	93144.97
567	294500	75000	73200.00	97520	127160	142296.7	189978.10
901	169500	148000	136666.67	171800	157250	134071.7	131516.80
388	85000	61000	53166.67	87700	89100	89900.0	96851.49

- Which model performed best?

metric	knn1	knn3	knn5	knn10	knn30	lin_model
RMSE	79113.17	64101.93	64291.1	65431.08	65288.48	67496.36

- Compare to the models with only longitude and latitude

Results

Some predictions:

	actual_price	knn1	knn3	knn5	knn10	knn30	lin_model
170	93000	94000	78666.67	79200	85440	103946.7	93144.97
567	294500	75000	73200.00	97520	127160	142296.7	189978.10
901	169500	148000	136666.67	171800	157250	134071.7	131516.80
388	85000	61000	53166.67	87700	89100	89900.0	96851.49

- Which model performed best?

metric	knn1	knn3	knn5	knn10	knn30	lin_model
RMSE	79113.17	64101.93	64291.1	65431.08	65288.48	67496.36

- Compare to the models with only longitude and latitude

metric	knn1	knn3	knn5	knn10	knn30	lin_model
RMSE	62850.24	58005.88	57727.92	57129.2	61405.59	79479

Results

Some predictions:

	actual_price	knn1	knn3	knn5	knn10	knn30	lin_model
170	93000	94000	78666.67	79200	85440	103946.7	93144.97
567	294500	75000	73200.00	97520	127160	142296.7	189978.10
901	169500	148000	136666.67	171800	157250	134071.7	131516.80
388	85000	61000	53166.67	87700	89100	89900.0	96851.49

- Which model performed best?

metric	knn1	knn3	knn5	knn10	knn30	lin_model
RMSE	79113.17	64101.93	64291.1	65431.08	65288.48	67496.36

- Compare to the models with only longitude and latitude

metric	knn1	knn3	knn5	knn10	knn30	lin_model
RMSE	62850.24	58005.88	57727.92	57129.2	61405.59	79479

The Curse of Dimensionality

- So what happened? Why did every model (except linear) got worse with more predictors?

The Curse of Dimensionality

- So what happened? Why did every model (except linear) got worse with more predictors?
- Some of the included variables may not have been useful in predicting Sale Price

The Curse of Dimensionality

- So what happened? Why did every model (except linear) got worse with more predictors?
- Some of the included variables may not have been useful in predicting Sale Price
- When we find distances between a test point and the training set, with a large number of predictors, the test point will tend to be far away from most if not all of the training set.

The Curse of Dimensionality

- So what happened? Why did every model (except linear) got worse with more predictors?
- Some of the included variables may not have been useful in predicting Sale Price
- When we find distances between a test point and the training set, with a large number of predictors, the test point will tend to be far away from most if not all of the training set.
  - I can find a house which is geographically close to mine, but. . .
  - It might be hard to find a house which is *simultaneously* geographically close, and was built in the same year, and was sold recently, and which has the same number of bedrooms, and. . .

The Curse of Dimensionality

- So what happened? Why did every model (except linear) got worse with more predictors?
- Some of the included variables may not have been useful in predicting Sale Price
- When we find distances between a test point and the training set, with a large number of predictors, the test point will tend to be far away from most if not all of the training set.
    - I can find a house which is geographically close to mine, but...
    - It might be hard to find a house which is *simultaneously* geographically close, and was built in the same year, and was sold recently, and which has the same number of bedrooms, and...
    - The response values for the "neighbors" might not be representative of the response value for the test point.

The Curse of Dimensionality

- So what happened? Why did every model (except linear) got worse with more predictors?
- Some of the included variables may not have been useful in predicting Sale Price
- When we find distances between a test point and the training set, with a large number of predictors, the test point will tend to be far away from most if not all of the training set.
  - I can find a house which is geographically close to mine, but...
  - It might be hard to find a house which is *simultaneously* geographically close, and was built in the same year, and was sold recently, and which has the same number of bedrooms, and...
  - The response values for the "neighbors" might not be representative of the response value for the test point.
- This is known as the **curse of dimensionality**

The Curse of Dimensionality

- So what happened? Why did every model (except linear) got worse with more predictors?
- Some of the included variables may not have been useful in predicting Sale Price
- When we find distances between a test point and the training set, with a large number of predictors, the test point will tend to be far away from most if not all of the training set.
    - I can find a house which is geographically close to mine, but. . .
    - It might be hard to find a house which is *simultaneously* geographically close, and was built in the same year, and was sold recently, and which has the same number of bedrooms, and. . .
    - The response values for the “neighbors” might not be representative of the response value for the test point.
- This is known as the **curse of dimensionality**
    - In general, non-parametric methods are more susceptible to this than parametric methods

The Curse of Dimensionality

- So what happened? Why did every model (except linear) got worse with more predictors?
- Some of the included variables may not have been useful in predicting Sale Price
- When we find distances between a test point and the training set, with a large number of predictors, the test point will tend to be far away from most if not all of the training set.
  - I can find a house which is geographically close to mine, but. . .
  - It might be hard to find a house which is *simultaneously* geographically close, and was built in the same year, and was sold recently, and which has the same number of bedrooms, and. . .
  - The response values for the "neighbors" might not be representative of the response value for the test point.
- This is known as the **curse of dimensionality**
  - In general, non-parametric methods are more susceptible to this than parametric methods
  - Non-parametric methods tend to perform *worse* that parametric methods when there are small number of observations per predictor in the model