K-Nearest Neighbors

COSC 410: Applied Machine Learning
Fall 2025
Prof. Forrest Davis

September 2, 2025

Warm-up

1. Talk to the person next to you about the best fall food.

2. Consider Figure 1. What shapes would you assign to the points labeled 1, 2, and 3? Why?

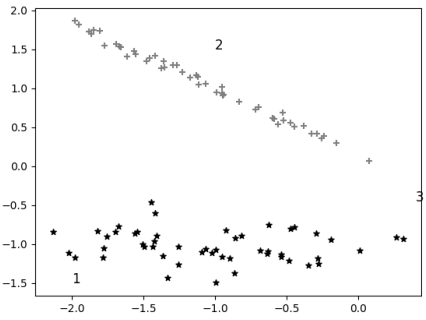

Figure 1: Data with two labels, stars and pluses.

Logistics

- On Monday, September 29 @ 4:30 in the EEP (downstairs in Bernstein), there is a opportunity to meet with Richard Maxwell
  - Richard Maxwell is in Residency at Colgate this semester and is interested in exploring the intersection of robotics and art, and particular theater.
  - If you are interested in robotics, this could be a good way to come to an exciting final project

Learning Objectives

- Describe the basic aims of k-nearest neighbors
- Calculate distance using a variety of metrics
- Understand how k impacts model training and generalization performance
- Articulate limitations of k-nearest neighbors

Summary: We cover k-nearest neighbors, including its aims, some definitions of distance, practical considerations in fitting a model, and some important limitations.

Basic Aims of K-Nearest Neighbors

K-NEAREST NEIGHBORS IS A SIMPLE, yet effective, approach to classification (or regression). It is **non-parametric**, meaning the model does not fit parameters (i.e., there is no 'learning'), and it is an example of instance-based learning. The basic approach is straightforward, when making a prediction about the label (or value) associated with a new data point, you base the prediction on the k nearest points.

Question

Consider Figure 2, with a k of 1 (i.e., you only consider the label of the nearest point), what shape do you predict for the red circle?

With $k > 1$, we have to decide how to weigh the labels of the nearest neighbors. A simple approach to start us off is to determine the label based on the mode label.

Question

Consider Figure 2 again, now with a k of 5. What label do you predict for the red circle?

Can you see how this can extend to regression? Think through it with Figure 3.

Question

What kind of paradigm is applicable to K-Nearest Neighbors and why?

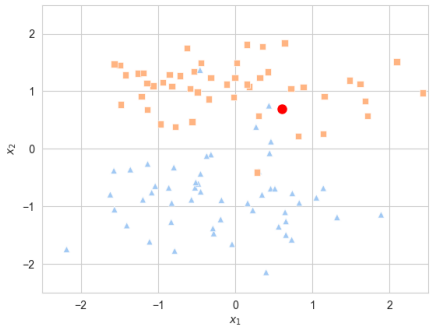

Figure 2: Data with two labels, triangles and squares. One additional data point has been added as a red circle.

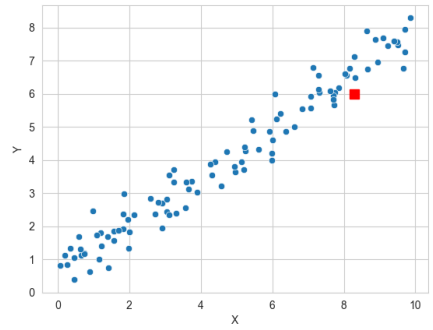

Figure 3: Data with one feature (X) and one label (continuous; Y). One additional data point has been added as a red square.

Defining Closeness

WE HAVE BEEN WORKING WITH AN IMPLICIT understanding of closeness. Let's make this vibe official by considering various distance metrics. Beginning with numeric data, recall vectors. As a help-

ful reference, consider the vectors $\vec{u}$ and $\vec{t}$ drawn from $\mathbb{R}^4$ (meaning they have a dimensionality of 4):

$$\vec{u} = \begin{bmatrix} 1.2 \\ -0.56 \\ 10.3 \\ 7.1 \end{bmatrix} \qquad \vec{t} = \begin{bmatrix} -2 \\ 1 \\ 4 \\ 3.1 \end{bmatrix}$$

Two common distance metrics for data like this (e.g., continuous) is **Euclidean** and **Manhattan** distance, defined below.[1]

[1] Manhattan distance is also called 'city block distance' and 'taxicab distance'. It's possible someone calls it 'uber distance' though it really isn't that great for most problems.

$$\text{euclidean} = \sqrt{\sum_{i=0}^{n-1} (p_i - q_i)^2} \quad : \quad p, q \in \mathbb{R}^n$$

$$\text{manhattan} = \sum_{i=0}^{n-1} |p_i - q_i| \quad : \quad p, q \in \mathbb{R}^n$$

These distance metrics don't work so well for strings (despite being vectors of characters in computer science).[2] Two alternative metrics that work with string data are the **Hamming** and **Levenshtein** distances. The hamming distance is defined for equal length strings and is the total number of positions that differ between the strings. Levenshtein is a measure of the minimum number of single-character edits (that is, insertions, deletions, or substitutions) required to change one string into another. The Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

[2] We can use these metric for numbers as well. For example, in comparing binary numbers we can take the Hamming distance between them.

Question

Calculate the distances between the two data points for each problem using the specified distance metric.

1. Using euclidean distance, determine the distance between $\begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix}$

2. Using Hamming distance, determine the distance between '9 2 1 5 6' and '8 2 3 1 6'

Finding a Good k

NOW THAT WE KNOW WHAT K MEANS, how do we decide what value to use? In other words, how does k affect (1) model fit to training data and (2) model generalizability?[3] Consider the sample data in Figure 4. When we fit a k-nearest neighbors model to this data, we are implicitly creating a **decision boundary**, carving space into one of the two labels, based on the proximate data. As we vary k, the decision boundary changes qualitatively (and quantitatively).

Question

Based on Figure 5, what does a lower k do to the decision boundary? What do you expect to see with a k of 1?

There is a trade-off between accuracy on the training set and generalizability. We will unpack this throughout the semester.

[3] I draw on Chapter 2 of Hastie [2001] for the discussion of decision boundaries and error rates.

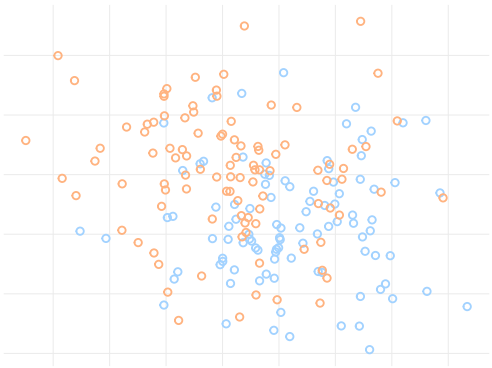

Figure 4: Data with two features and two possible labels (orange and blue).

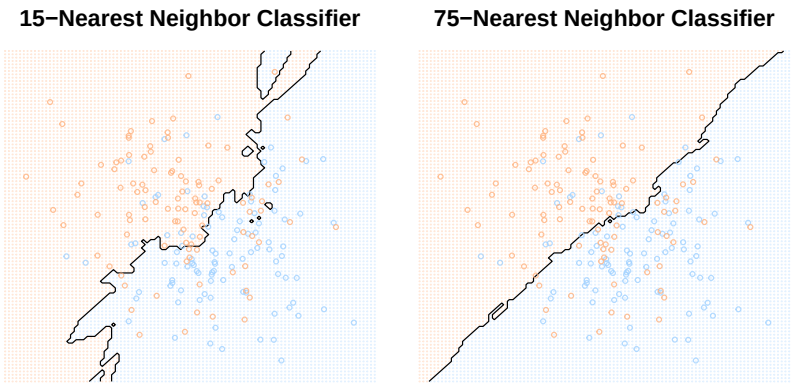

Figure 5: Decision boundary (in black) when fitting two nearest neighbor models to the data in Figure 4.

Question

How does varying k impacts accuracy on the training set and generalizability?

Issues

TO BUILD A DEEPER UNDERSTANDING of how this machine learning algorithm works, let's consider, briefly, some of its limitations. Note this limtations apply to the 'vanilla' version of k-nearest neighbors we have been discussing today. While related to k-nearest neighbors, these issues are important in a variety of settings.

1. **Effect of Scaling Data:** As the amount of data increases, k-nearest neighbors gets (expoentially) slower.
2. **Influence of Distant Neighbors:** In k-nearest, all neighbors are treated equally. You can improve model performance by weighting neighbors based on distance.[4]
3. **Features of Differing Scales:** The algorithm is sensitive to the relative structure of the data. Because of the use of distance, if the features are from different scales, then model performance can be driven by superficial differences between features, leading to poor performance.

[4] A common weighting scheme for k-nearest neighbors is to give each neighbor a weight of $\frac{1}{d}$, where d is the distance between the point and the neighbor.

4. **Curse of Dimensionality:** The overall dimensionality of the input (i.e., the total number of features) can negatively impact performance.[5] Namely, in higher dimensions, the proportion of the range of a specific dimension need to find enough neighbors grows. Additionally, there is an exponential growth in the amount of data you need to fit a representative model. These combine to make k-nearest neighbors unsuitable for high dimensional data (without modification to the data).

   More concretely, suppose we gather data (which uniformly populates the space) from a hypercubical (a volume in high dimensions) neighborhood around a target point. If we aim to capture some fraction r of the data, the expected edge length would be $e_p(r) = r^{\frac{1}{p}}$ (where p is the dimensionality). For one dimension, you might be able to see that to capture 10% of data along the axis (assuming the data is uniformly distributed), we need to sample 10% of the range (as 100% of the data populate the entire range). Notice that in ten dimensions $e_{10}(0.01) = 0.63$ and $e_{10}(0.1) = 0.80$. That is, to capture 1% or 10% of the data in order to gather near neighbors, we have to cover 63% or 80% of the range of each input variable! Moreover, the sampling density is propositional to $N^{\frac{1}{p}}$. If we get a dense sample for a problem with a single feature of 100, then we need a sample size of 100^{10} in ten dimensions to get the same density.

[5] The discussion here draws on Chapter 2 of Hastie [2001].

Practice Problems

Complete these with the people around you.

1. Consider a toy example of k-nearest neighbors. You work at Hamilton Whole Foods and have decided to add a barcode scanner. You got one at discount, so it makes occasionally mistakes in retrieving the binary number (e.g., a sequence of 1s and 0s) from the item. You are piloting it with three example products, given with their true barcodes in Table 1.

   The scanner returns 00100. Using a k of 1, what product would you predict for this scan?

Curry Tofu Marinade (jar)	00110
Kombucha	10101
Grape Leaves	01010

Table 1: Sample products and barcodes.

Before Next Class

- Complete Lab 0, due today
- Work on Codelet 0, due Friday Sep 5
- No new reading, but please complete the pre-class quiz

References

Trevor Hastie. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, New York, 2001. ISBN 978-0-387-95284-0.