kNN (or k-Nearest Neighbors)

Intuition: Classify a color (RGB vector)

[255,0,0], [255,51,0], [255,102,0], [204,0,0], [190,0,20]

[0,0,153], [0,51,204], [51,51,255], [0,102,255], [204,0,198]

[225,51,120]

Strategy 1

Suppose we want to classify $[225,51,120]$

*

1. Calculate the *distance* between $[225,51,120]$ and every other vector
2. Find the vector closest to $[225,51,120]$, call it v
3. Classify $[225,51,120]$ the same classification given to v

Applying Strategy 1

If we use the Euclidean distance, the nearest neighbor to $[225,51,120]$ turns out to be $[204,0,198]$ which is BLUE!

So now our red asterisk is considered BLUE (bad classification)

What happened?

The nearest neighbor $[225,51,120]$ (the purple asterisk) is a *noisy* data point. It was classified as BLUE (for the sake of approximation). It is an *outlier*: far (Euclidean distance) from the other data points in BLUE. Our algorithm is sensitive to *noise* (or *mis-labeled* data).

Strategy 2

Our approach of just looking for the nearest neighbor did not perform well. A wiser strategy could be to consult more than one neighbor.

Strategy 2:

Find the nearest **three**, say, neighbors. If two of them are BLUE, then our point can be considered BLUE. Otherwise, it is RED.

This strategy is called 3-NN. It can be generalized to k-NN (k is a positive integer): if the majority of neighbors are BLUE, then classify the new point as BLUE.

Check the code files for implementation

Overfitting

Overfitting occurs when a model learns the training data too well, including the noise and outliers, leading to poor performance on new, unseen data. This is exactly what happened in Strategy 1.

Treatments:

In the context of kNN, we make k larger

We will see different models in the future and different treatments

Bias-variance trade-off

The choice of **k** directly relates to this fundamental concept. A small **k** leads to low bias (the model can fit complex patterns) but high variance (highly fluctuating predictions -- very sensitive to noise). A large **k** leads to high bias (it smooths out local patterns – but predictions may become far off from true values) but low variance.

To understand this better in the context of a classification algorithm, let us talk about the *decision boundary*

Decision Boundary

It is a surface (or curve if the data are 2D, or hypersurface if the data are of dimension >3) which separates the space into zones (in our case we have 2 zones: BLUE and RED).

If we want to classify a point, we find in which zone it falls

For an illustration, check the 3D plots in “Decision Boundaries.ipynb”

Remark

Non-parametric methods: kNN is a **non-parametric** (“lazy”) method, meaning it makes no assumptions about the underlying data distribution. It doesn’t learn **k** during a training phase. It simply stores the training data and performs computations at prediction time. This is a crucial distinction from parametric methods like linear regression or Gaussian Naive Bayes which we will be seeing soon.

Computational Complexity and Data Structures Considerations

The naive implementation of kNN is computationally expensive. For each new data point, you must calculate the distance to every single point in the training set. This is a major drawback for large datasets. The complexity is roughly $O(N \cdot D)$, where N is the number of training samples and D is the number of features.

k-d Trees: This is the most common data structure for accelerating kNN search. A k-d tree partitions the data space into smaller regions, allowing the algorithm to quickly prune away large portions of the data that are too far away to contain the nearest neighbors.

Ball Trees: Similar to k-d trees, ball trees are a type of space-partitioning data structure that can be more effective for high-dimensional data or when the data is not axis-aligned. They work by partitioning points into a series of nested hyperspheres.

Approximate Nearest Neighbor (ANN) search: Practical applications use approximation algorithms like **Locality-Sensitive Hashing (LSH)**

Curse of dimensionality

As the number of features (dimensions) increases, the concept of "nearest neighbor" becomes less meaningful. In high-dimensional spaces, all data points tend to be roughly equidistant from each other, making distance-based methods like kNN ineffective. *Why do you think?*

On the next pages you will find an example that shows the *Distance Concentration Phenomenon*. That is, in high dimensions, random points become nearly equidistant from fixed reference points

Curse of dimensionality (a toy example)

Consider the following two points in $\mathbb{R}^k$ for some fixed $k \geq 2$

$$u = (1,0, \dots, 0)$$
$$v = (0, 0, \dots, 1)$$

Consider the point $t = (1, x_2, \dots, x_k)$, where $x_2, \dots, x_k$ are randomly sampled from the interval $[0, 0.25]$.

The point t is picked to be closer to u than it is to v, and the lower the dimension the clearer that is.

The Euclidean distance between t and $u = \sqrt{\sum_{i=2}^{k} {x_i}^2} \leq \sqrt{\sum_{i=2}^{k-2} {x_i}^2 + 0.25^2}$

The Euclidean distance between t and $v = \sqrt{1 + \sum_{i=2}^{k-2} {x_i}^2 + (1 - x_k)^2} \geq \sqrt{\sum_{i=2}^{k-2} {x_i}^2 + 1 + 0.75^2}$

Toy example continued

For $k = 2$:

The Euclidean distance between t and $u = \sqrt{\sum_{i=2}^{2} {x_i}^2} = x_2 \leq 0.25$

The Euclidean distance between t and $v = \sqrt{1 + (1 - x_2)^2} \geq \sqrt{1 + 0.75^2} = 1.25$

The ratio $\frac{\text{The distance between } t \text{ and } v}{\text{The distance between } t \text{ and } u} \geq 5$
showing a clear distinction of which point is nearest to t

Check the code ‘curse1.ipynb’. There we try dimensions up to 10^5. You will notice that for small k, the ratio is informative (clearly above 1). However, for large k, the ratio gets close to 1 (unclear which point is closer)

A comment

Our example can be seen as the Central Limit Theorem (CLT) in action. *The sum of many small random effects converges to a predictable mean with shrinking relative variance*

Curse of dimensionality videos

Watch this https://youtu.be/dZrGXYty3qc?t=538

And this https://youtu.be/9Tf-_mJhOkU

Numerical Analysis Considerations

Distance metrics: The choice of distance metric is critical. While *Euclidean distance* is common, one should mention others like *Manhattan distance* for non-linear relationships or *cosine similarity* for high-dimensional text data

A video to watch

Watch this https://youtu.be/dZrGXYty3qc?t=127 (starting at 2:03)