

KNN Distances

Table of contents

Import notebook funcs	1
---------------------------------	---

Import notebook funcs

```
from notebookfuncs import *

import torch
import matplotlib.pyplot as plt
import seaborn as sns

def generate_n_dimensional_tensor(n, num_points=1000):
    """Generates an n-dimensional tensor of random numbers.

    Args:
        n: The desired number of dimensions.
        num_points: The number of data points to generate.

    Returns:
        An n-dimensional PyTorch tensor.
    """

    return torch.rand(num_points, n)

def euclidean_distance(p1, p2):
    """Calculates the Euclidean distance between two tensors.

    Args:
        p1: The first tensor.
        p2: The second tensor.

    Returns:
        The Euclidean distance between the two tensors.
    """
```

```

    return torch.norm(p1 - p2, dim=-1)

def plot_distance_histograms(distances_list, dimensions):
    """Plots histograms of the Euclidean distances on a single plot.

    Args:
        distances_list: A list of lists, each containing distances for a specific
        ↪ dimension.
        dimensions: A list of dimensions.
    """

    plt.figure(figsize=(12, 6))
    for i, (distances, n) in enumerate(zip(distances_list, dimensions)):
        sns.kdeplot(distances, fill=True, label=f"n = {n}")

    plt.xlabel("Euclidean Distance")
    plt.ylabel("Density")
    plt.title("Kernel Density Estimates of Euclidean Distances")
    plt.legend()
    plt.show()

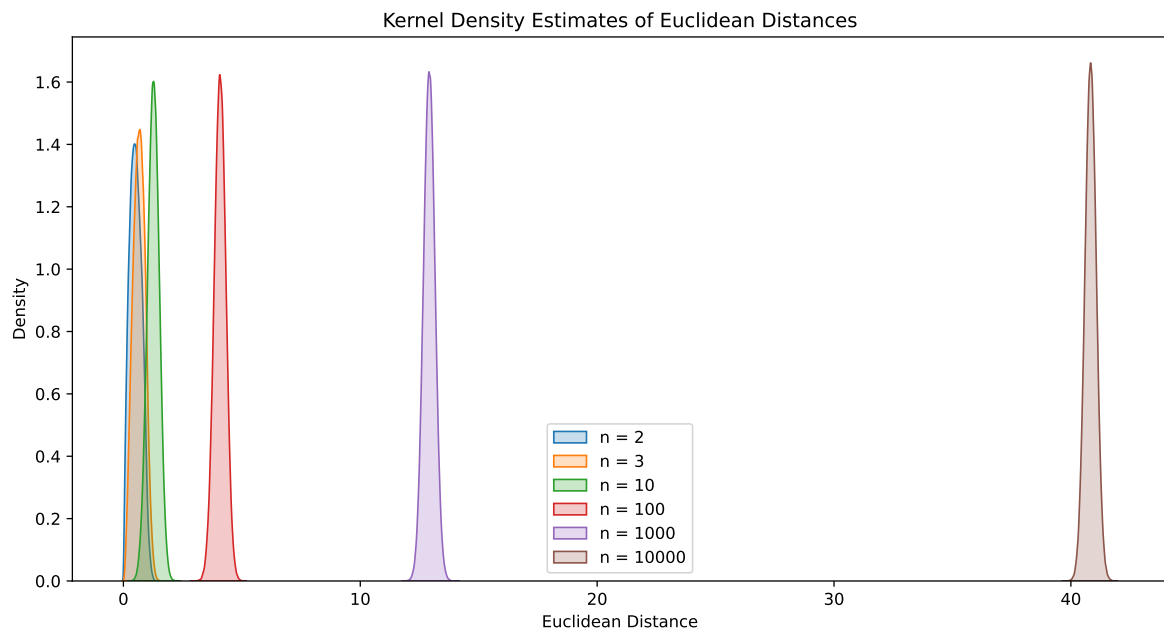
# Define the desired dimensions
dimensions = [2, 3, 10, 100, 1000, 10000]

# Generate and print the arrays
distances_list = []
for n in dimensions:
    tensor = generate_n_dimensional_tensor(n)
    num_points = tensor.shape[0]

    distances = []
    for i in range(num_points):
        for j in range(i + 1, num_points):
            distance = euclidean_distance(tensor[i], tensor[j]).item()
            distances.append(distance)
        distances_list.append(distances)

plot_distance_histograms(distances_list, dimensions)

```



```
printlatex("$\\text{The Curse of Dimensionality } \\implies \\text{ As } n \\to \\infty, \\text{ distances between points increase.}$")
```

The Curse of Dimensionality \implies As $n \rightarrow \infty$, distances between points increase.

Reference: 1. https://www.cs.cornell.edu/courses/cs4780/2022fa/slides/KNN_annotated.pdf

```
allDone();
```

```
<IPython.lib.display.Audio object>
```