# 📚 Module 20: Distance Calculations - Complete Notes

## 🎯 What You'll Learn

Master **distance metrics** — measuring how far apart two points are. Essential for similarity calculations, clustering, and nearest-neighbor algorithms.

---

## 📖 Concept Explained

### Types of Distance

```
Point A: (1, 1)    Point B: (4, 5)

MANHATTAN (City blocks):
    |4−1| + |5−1| = 3 + 4 = 7
    (Can only move horizontally/vertically)

EUCLIDEAN (Straight line):
    √((4−1)² + (5−1)²) = √(9 + 16) = √25 = 5
    (As the crow flies)
```

### Visual

```
    B(4,5)
    |\
    | \  Euclidean = 5
    |  \
    |   \
    ├───A(1,1)

    Manhattan = 3 + 4 = 7 (go right 3, up 4)
```

---

## 💻 Programming Connection

### Code Examples

```python
# Example 1: Manhattan Distance

def manhattan_distance(p1, p2):
    """Sum of absolute differences (L1)"""
    return abs(p1[0] - p2[0]) + abs(p1[1] - p2[1])

print(manhattan_distance((1, 1), (4, 5)))  # 7
```

```python
# Example 2: Euclidean Distance

import math
```

```python
def euclidean_distance(p1, p2):
    """Straight line distance (L2)"""
    return math.sqrt((p2[0] - p1[0])**2 + (p2[1] - p1[1])**2)

print(euclidean_distance((0, 0), (3, 4)))  # 5.0 (3-4-5 triangle!)
```

```python
# Example 3: Find Nearest

def find_nearest(target, points):
    """Find point closest to target"""
    return min(points, key=lambda p: euclidean_distance(target, p))

points = [(0, 0), (5, 5), (2, 1)]
print(find_nearest((3, 3), points))  # (2, 1)
```

```python
# Example 4: N-dimensional Distance (for embeddings!)

def distance_nd(p1, p2):
    """Euclidean distance for any dimension"""
    return math.sqrt(sum((a - b)**2 for a, b in zip(p1, p2)))

# 3D points
a = (1, 2, 3)
b = (4, 5, 6)
print(distance_nd(a, b))  # 5.196
```

---

## 🧪 SDET/Testing Application

```python
# SDET Scenario: Find Similar Embeddings

def find_similar(query_embedding, embeddings, top_k=3):
    """Find most similar embeddings by distance"""
    distances = [(i, distance_nd(query_embedding, emb))
                 for i, emb in enumerate(embeddings)]
    distances.sort(key=lambda x: x[1])
    return distances[:top_k]

# Smaller distance = more similar
```

---

## 🔑 Key Takeaways

✅ **Manhattan** — Grid movement (|dx| + |dy|)
✅ **Euclidean** — Straight line (√(dx² + dy²))
✅ **Smaller distance = More similar**

---

💾 *Save as:* `Module_20_Distance_Calculations.md`