
349:Machine Learning

Fall 2024

Measuring Distance
and k-Nearest Neighbors

Why study distance?

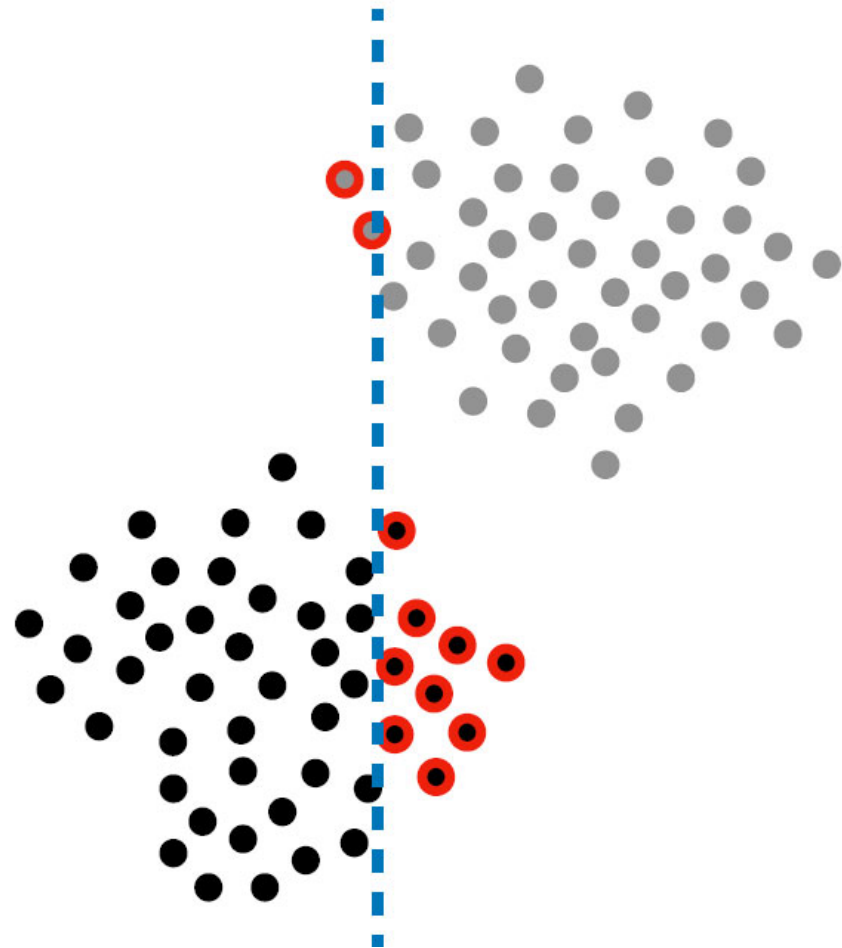
- Concepts of distance can be ambiguous, and intuition can break down
- Clustering requires distance measures
- Local methods require a measure of “locality”
- Search engines require a measure of similarity
- *Virtually* all neural methods require distance measures

Why study distance?

1. Pick data \mathbf{D} , model $\mathbf{M}(\mathbf{w})$ and an objective function $\mathbf{J}(\mathbf{D}, \mathbf{w})$.
2. Initialize model parameters \mathbf{w} somehow.
3. Measure model performance with the objective function $\mathbf{J}(\mathbf{D}, \mathbf{w})$.
4. Modify the parameters \mathbf{w} somehow, hoping to improve $\mathbf{J}(\mathbf{D}, \mathbf{w})$.
5. Repeat steps 3 and 4 until you stop improving or run out of time.

Why study distance?

We move the line $M(w)$ such that it reduces the objective function $J(D, w)$.



What is a “metric”?

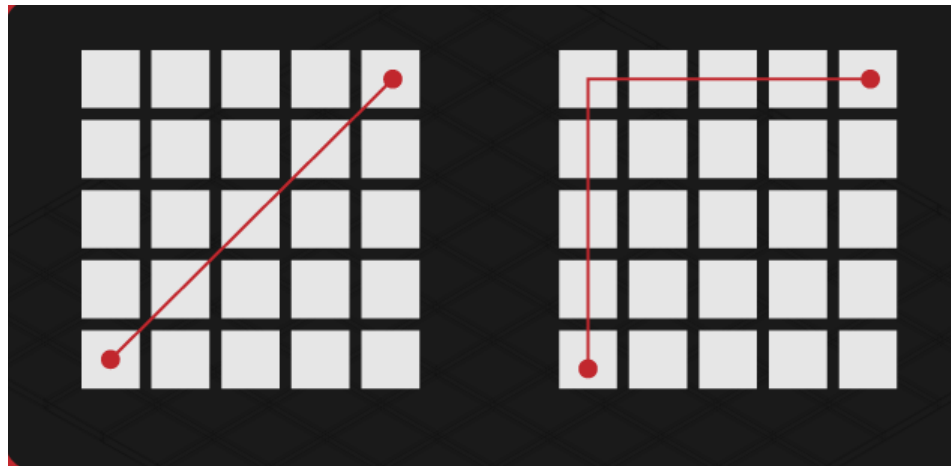
A binary **function** with all four of these qualities:

$$d(x, y) = 0 \quad \text{iff} \quad x = y \quad (\text{reflexivity})$$

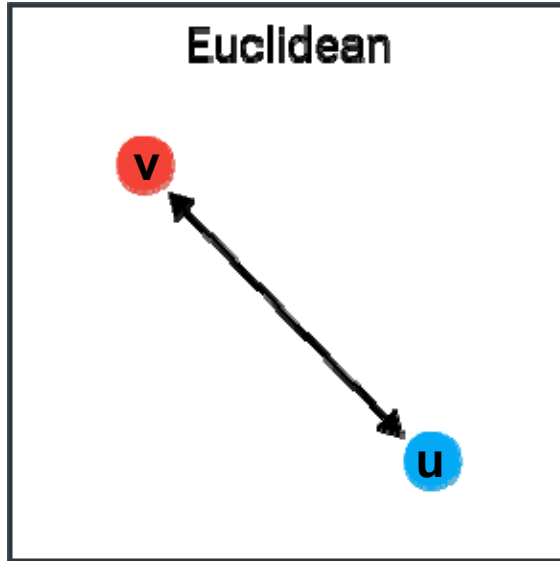
$$d(x, y) \geq 0 \quad (\text{non - negative})$$

$$d(x, y) = d(y, x) \quad (\text{symmetry})$$

$$d(x, y) + d(y, z) \geq d(x, z) \quad (\text{triangle inequality})$$



Euclidean Distance



Source: Maarten Grootendorst

What people intuitively think of as “distance”

Most commonly used distance metric

Straight forward to calculate

Not scale invariant → normalization (in most cases)

“Curse of Dimensionality”

- every point is “equally” distant in high dimensions


Typical use cases include

- K-nearest neighbors, K-means, etc.
- Models with low-dimensional data

$$d(\mathbf{u}, \mathbf{v}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$$

Euclidean Distance -- Generalized Formula

n = the number of dimensions


$$d(\mathbf{u}, \mathbf{v}) = \left[\sum_i^n |u_i - v_i|^2 \right]^{1/2}$$

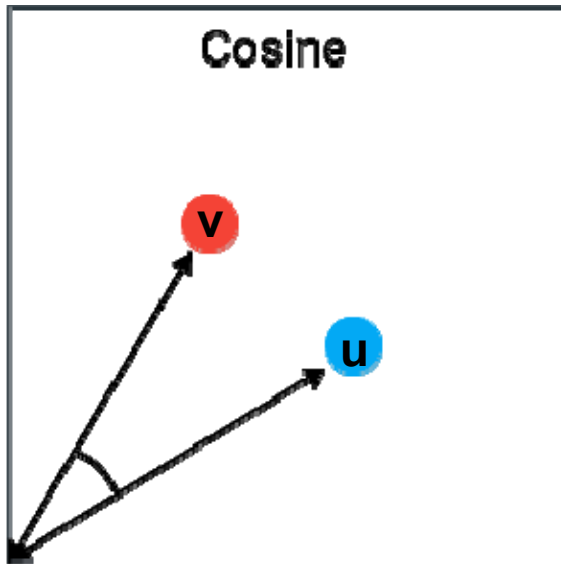
Where...

$$\mathbf{u} = [u_1, u_2, u_3, \dots, u_n]$$

$$\mathbf{v} = [v_1, v_2, v_3, \dots, v_n]$$

$$\mathbf{u}, \mathbf{v} \in \mathcal{R}^n$$

Cosine Similarity



Source: Maarten Grootendorst

Measures similarity using angle between two vectors
Not impacted by dimensionality of feature space
Straight forward to calculate
May be more intuitive in high dimensions
Scale invariant -- vector magnitude is irrelevant
Ranges from -1.0 to 1.0 (e.g., [-1,1])

Typical use cases include

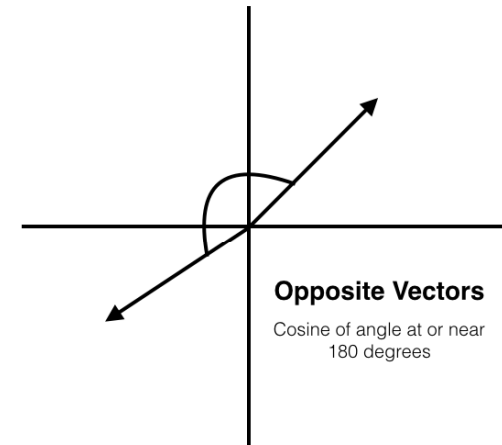
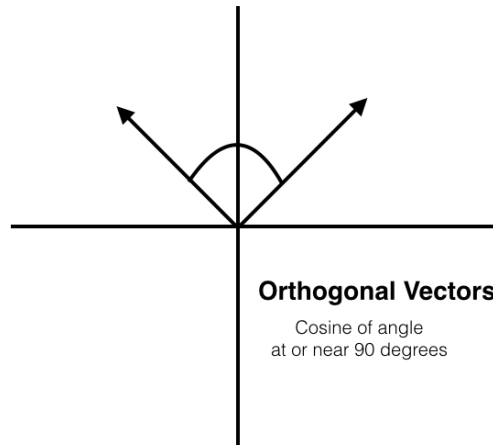
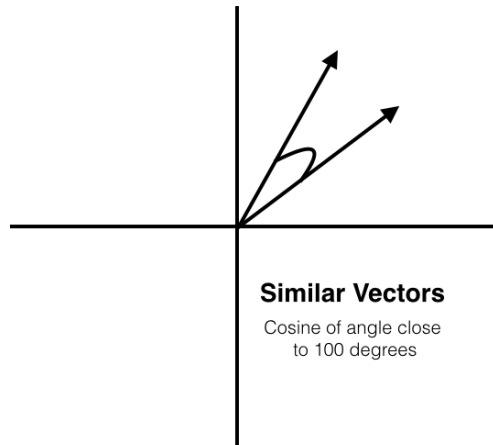
- High-dimensional data
- Natural language processing
- Google user vectors?

$$d(u, v) = \cos(\theta) = \frac{v \cdot u}{\|v\|_2 \|u\|_2}$$

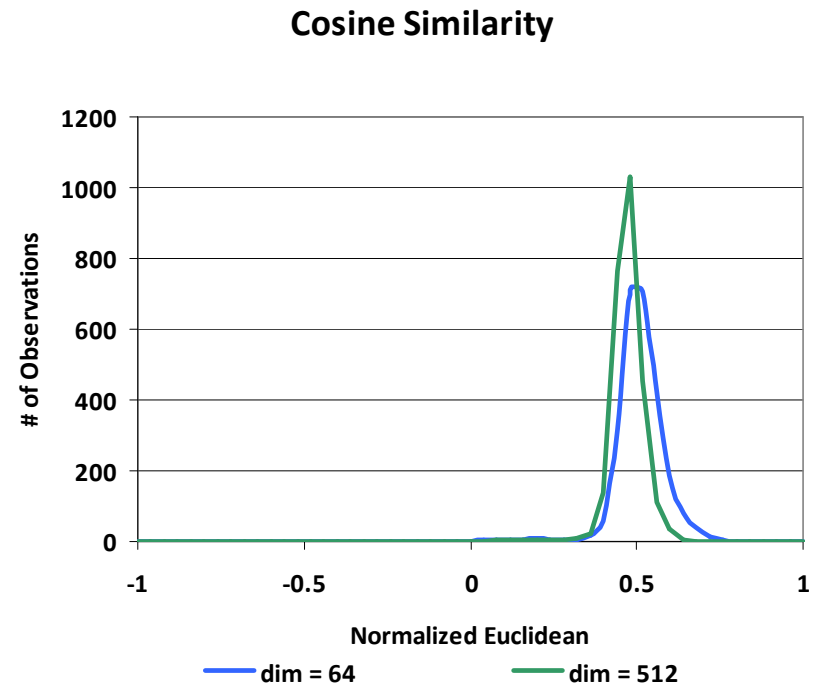
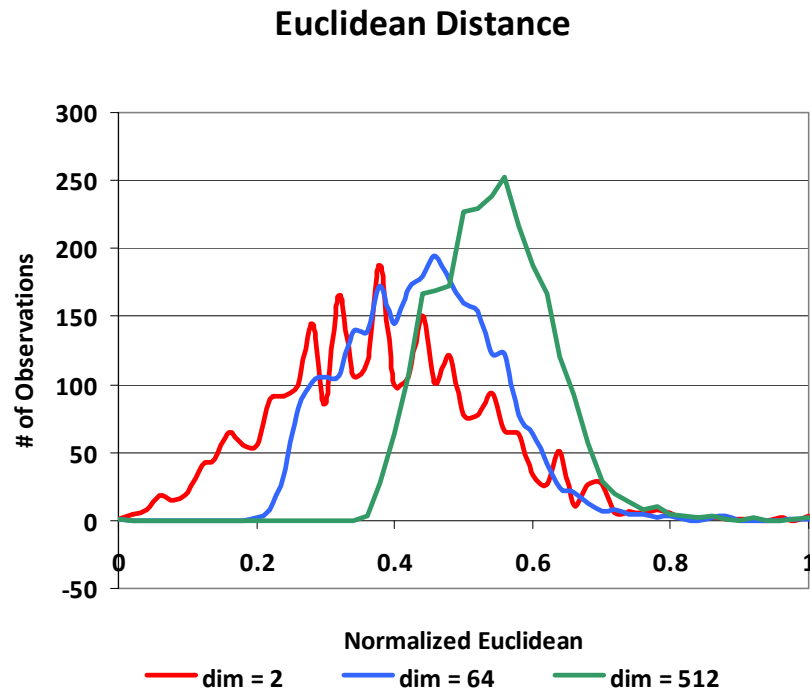
$$\|u\|_2 = d(u, 0) = \sqrt{\sum_{i=1}^n (u_i - 0)^2}$$

Cosine Similarity -- Interpreting Metric

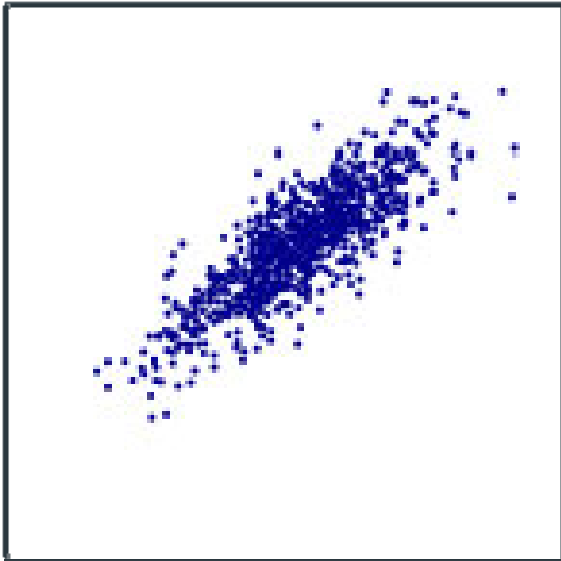
Similarity, unrelatedness and opposites...



Curse of Dimensionality



Pearson Correlation Coefficient



Measures correlation between two variables
Related to, but not identical to cosine similarity
Straight forward to calculate
Harnesses machinery of parametric statistics
Ranges from -1.0 to 1.0 (e.g., [-1,1])

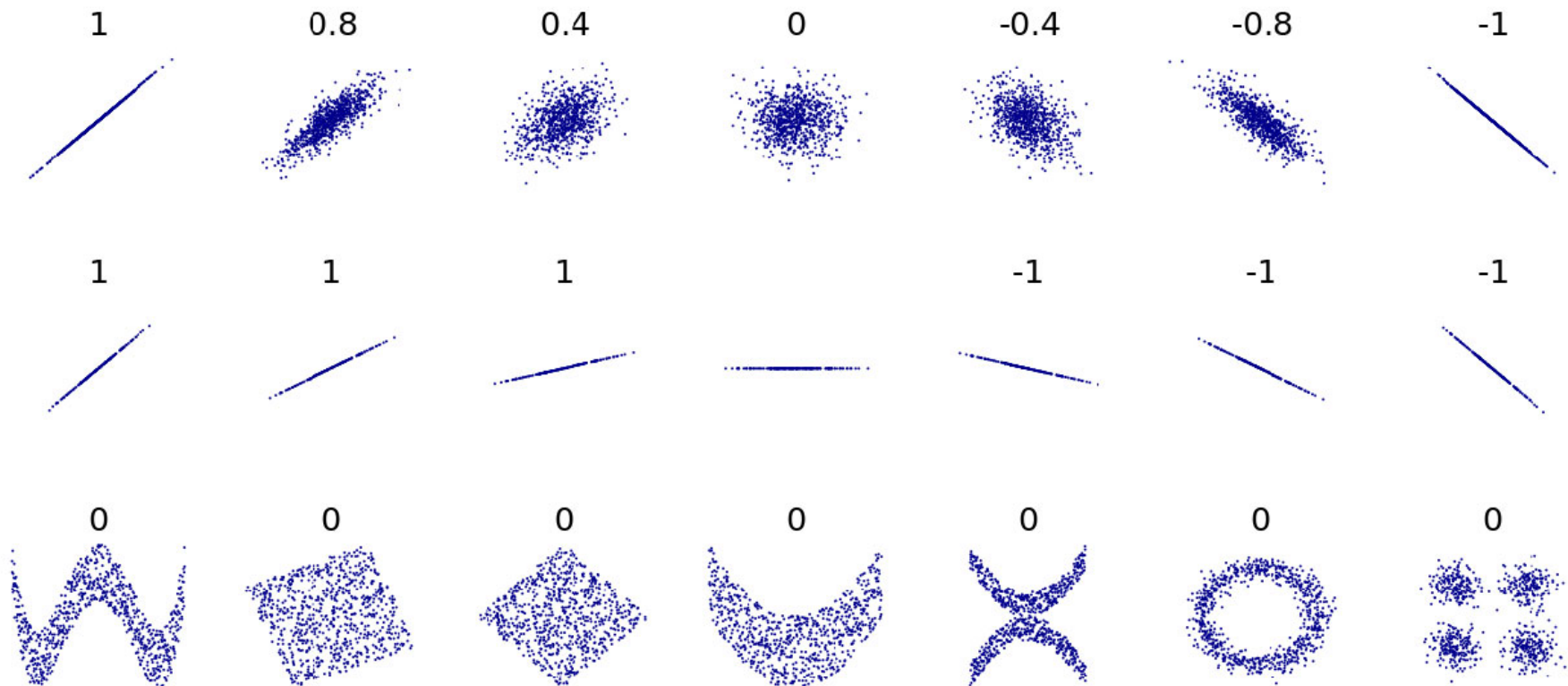
Typical use cases include

- Collaborative filtering (and similar applications)

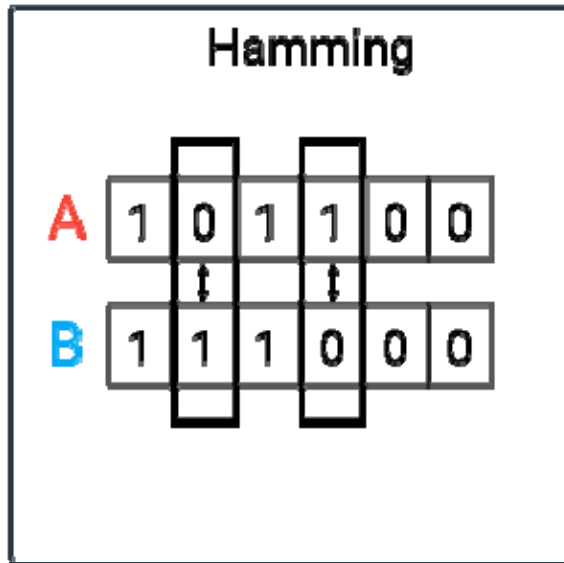
$$r_{\mathbf{xy}} = \frac{\sum_{i=1}^n (x_i - \mu_x) (y_i - \mu_y)}{\sqrt{\sum_{i=1}^n (x_i - \mu_x)^2} \sqrt{\sum_{i=1}^n (y_i - \mu_y)^2}}$$

$$\mu_{\mathbf{x}} = \frac{\sum_{i=1}^n x_i}{n}$$

Pearson Correlation -- Examples



Hamming Distance



Source: Maarten Grootendorst

Features are mapped to binary vectors

Measures number of attributes that are different

Does not take magnitude into account

Does not (explicitly) take similarities into account

Typical use cases include

- Error detection and correction
- Categorical variables

$$d(\vec{x}, \vec{y}) = \sum_{i=1}^n |x_i - y_i|$$

where $\vec{x} = \langle x_1, x_2, \dots, x_n \rangle$,

$\vec{y} = \langle y_1, y_2, \dots, y_n \rangle$

and $\forall i (x_i, y_i \in \{0, 1\})$

Hamming Distance - Example

Blues	Jazz	Rock	Zydeco	Vocals
0	0	1	0	1
0	1	0	0	0
0	0	1	0	0

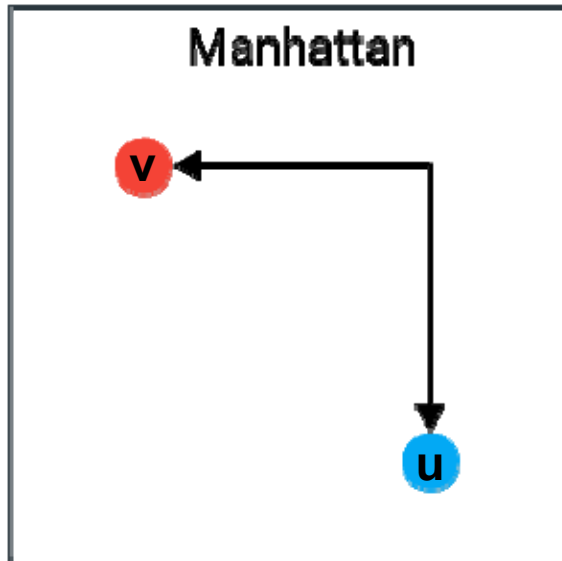
$s1 = \{\text{rock, vocals}\}$

$s2 = \{\text{jazz, no_vocals}\}$

$s3 = \{\text{rock, no_vocals}\}$

Hamming Distance = number of bits different
between binary vectors

Manhattan Distance



Source: Maarten Grootendorst

Sometimes known as “City Block” distance
Less intuitive than Euclidean or Cosine Similarity
Does not suffer from the “Curse of Dimensionality”

- values change by “quanta”

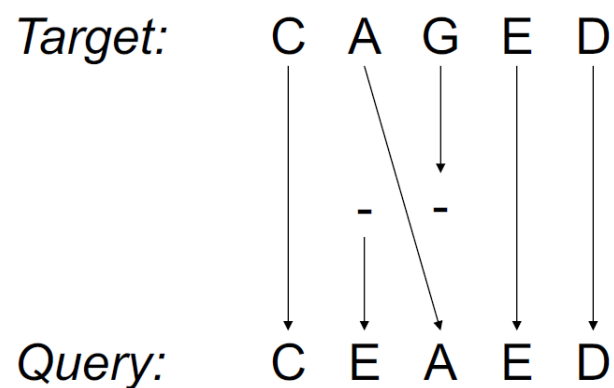
Not continuous → may not be differentiable

Typical use cases include

- Discrete (or binary) values

$$d(u, v) = \sum_{i=1}^n |u_i - v_i|$$

Edit Distance



Counts the number of “edit” operations to transform one string to another → cost of edits

Requires a finite “alphabet” or vocabulary

Operations can include “insert”, “delete” and “substitute”

Can also be applied to non-character attributes

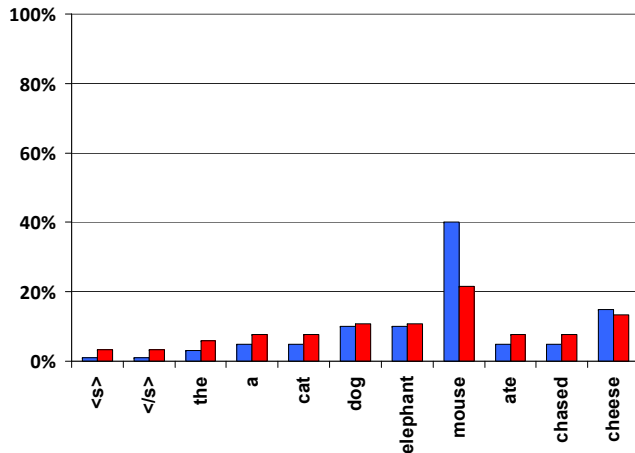
Levenshtein algorithm is common (costly) approach

- many other algorithms exist

Typical use cases include

- Gene sequence matching
- Spell checking
- Music melody matching
- Text generation (at the word level)
- Any sequential categorical data
- Graphical Models

Kullback-Leibler (KL) Divergence



Measures the distance between two distributions
Is not symmetric

Typical use cases include

- Stochastic environments with large number of variables
- Natural language processing
- Generative models

$$D_{KL}(P \parallel Q) = \sum_i \ln \left(\frac{P(i)}{Q(i)} \right) P(i)$$

Norms

Loosely, a Norm is a function that applies a positive value to all vectors (except the 0 vector) in a vector space.

Norms are related to distance metrics applies to $d(u,0)$

$$\begin{aligned}\|u\|_p &= \left(|x_0|^p + |x_1|^p + \cdots + |x_N|^p \right)^{\frac{1}{p}} \\ &= \left(\sum_{n=0}^N |x_n|^p \right)^{\frac{1}{p}}\end{aligned}$$

$\|u\|_0$ = L⁰ norm = Number of Non-Zero Attributes (p = 0)

$\|u\|_1$ = L¹ norm = Manhattan Distance (p=1)

= Hamming Distance (p = 1 and $\forall_i \ u_i, v_i \in \{0,1\}$)

$\|u\|_2$ = L² norm = Euclidean Distance (p = 2)

$\|u\|_\infty$ = L[∞] norm = Maximum Along a Dimension (p = ∞)

Potential Confusion Over Terms

Vector Norm: assigns a strictly positive value to all vectors u in a vector space, except the 0 vector, which has 0 assigned to it.

$$\|u\| \geq 0$$

Normal Vector: A vector is normal to another object if they are perpendicular to each other $\rightarrow \cosim(u,v) = 0$

Normalized Vector: A vector is normalized when it is transformed to have a unit length (e.g, the unit sphere).

$$v = \frac{u}{\|u\|}$$

Nearest Neighbor Classifier

Get some example set of cases with attributes and known outputs (e.g., diagnoses of infectious diseases by an expert)

- In other words, labeled data

When you see a new case, assign its output to be the same as the most similar known case

- Your symptoms most resemble Mr. X \rightarrow Mr. X had the flu \rightarrow You have the flu

General Learning Task

There is a set of possible examples $X = \{\vec{x}_1, \dots, \vec{x}_n\}$

Each example is an k -tuple of attribute values

$$\vec{x}_1 = \langle a_1, \dots, a_k \rangle$$

There is a target function that maps X onto some finite set Y

$$f : X \rightarrow Y$$

The DATA is a set of tuples $\langle \text{example}, \text{target function values} \rangle$

$$D = \{ \langle \vec{x}_1, f(\vec{x}_1) \rangle, \dots, \langle \vec{x}_m, f(\vec{x}_m) \rangle \}$$

Find a **hypothesis** h such that...

$$\forall \vec{x}, h(\vec{x}) \approx f(\vec{x})$$

Single Nearest Neighbor

Given some set of training data...

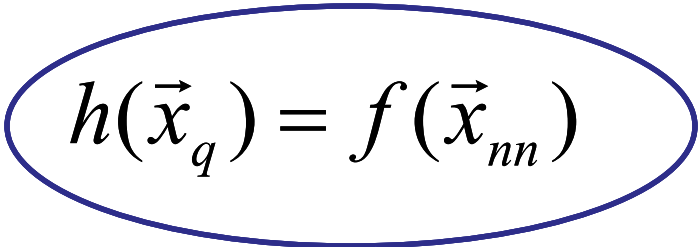
$$D = \{ \langle \vec{x}_1, f(\vec{x}_1) \rangle, \dots, \langle \vec{x}_m, f(\vec{x}_m) \rangle \}$$

...and query point \vec{x}_q , predict $f(\vec{x}_q)$

1. Find the nearest member of the data set to the query

$$\vec{x}_{nn} = \arg \min_{\vec{x} \in D} (d(\vec{x}, \vec{x}_q))$$


2. Assign the nearest neighbor's output to the query

$$h(\vec{x}_q) = f(\vec{x}_{nn})$$


Our hypothesis

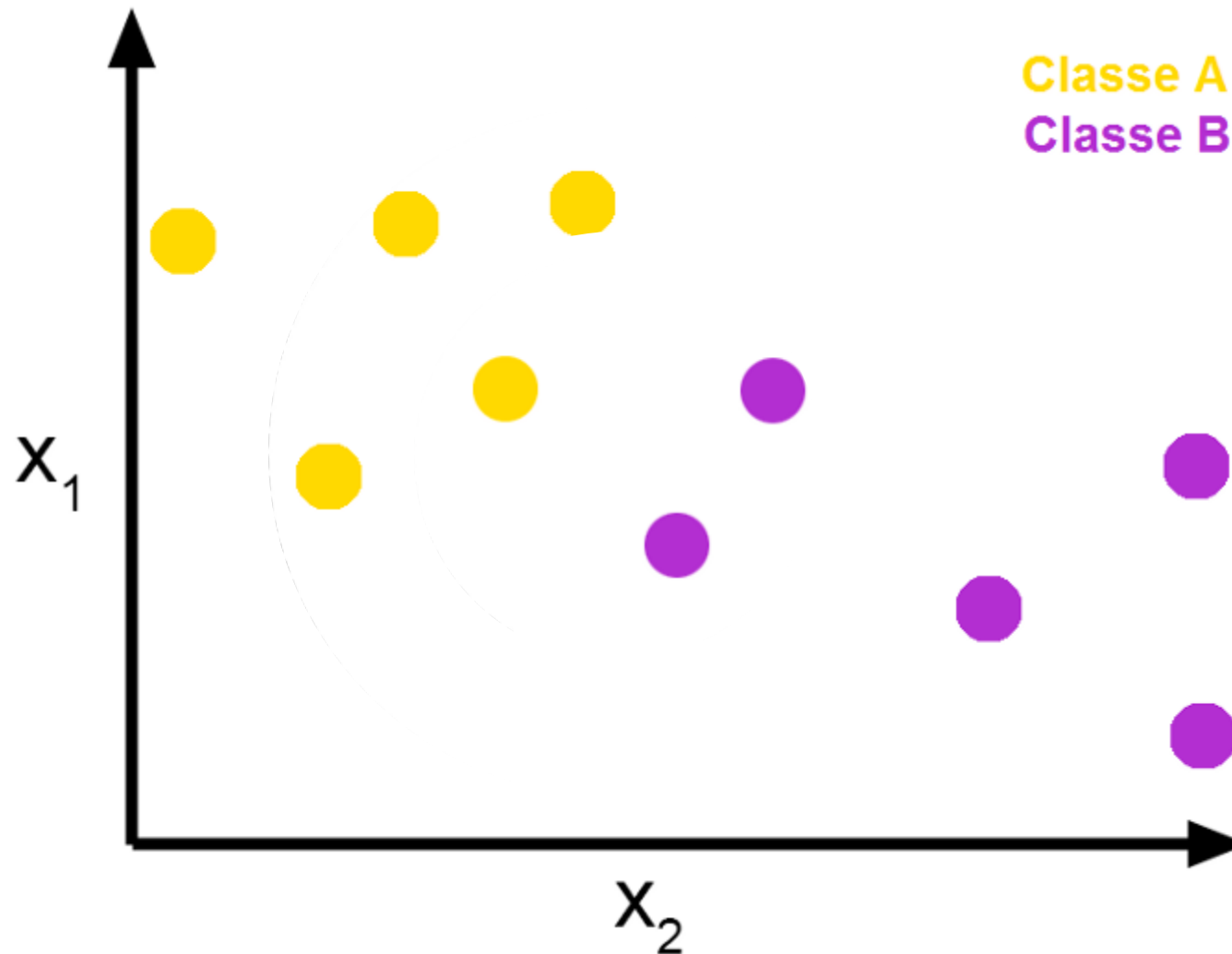
What makes a memory-based learner?

- A distance measure
Nearest neighbor: typically Euclidean
- Number of neighbors to consider
Nearest neighbor: One
- A weighting function (optional)
Nearest neighbor: unused (equal weights)
- How to fit with the neighbors
Nearest neighbor: Same output as nearest neighbor

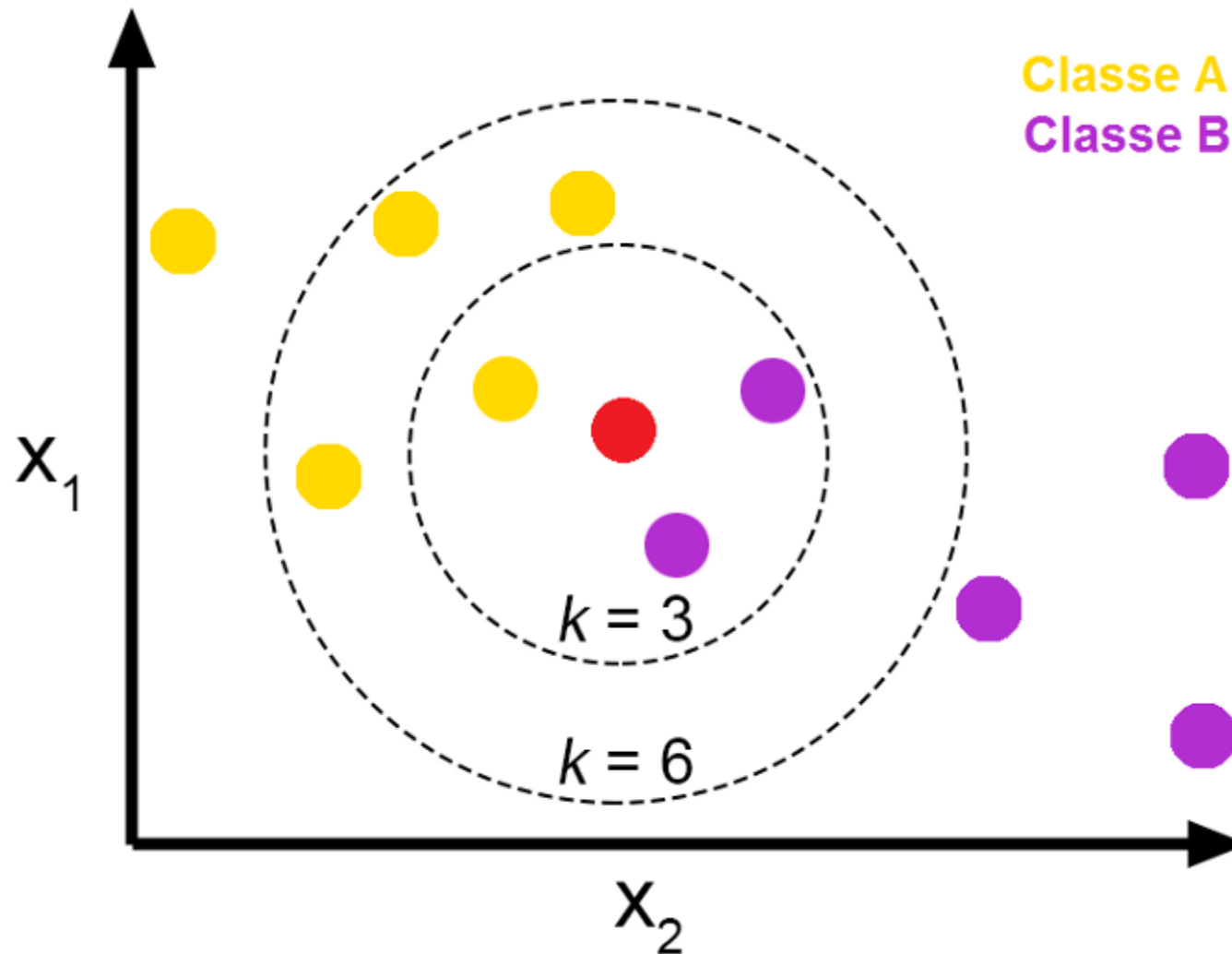
k-Nearest Neighbor

- A distance measure
Euclidean
- Number of neighbors to consider
K (algorithm? hint: sorting)
- A weighting function (optional)
Unused (i.e. equal weights)
- How to fit with the neighbors
Regression: average output among K nearest neighbors.
Classification: most popular output among K nearest neighbors

More Concrete Example

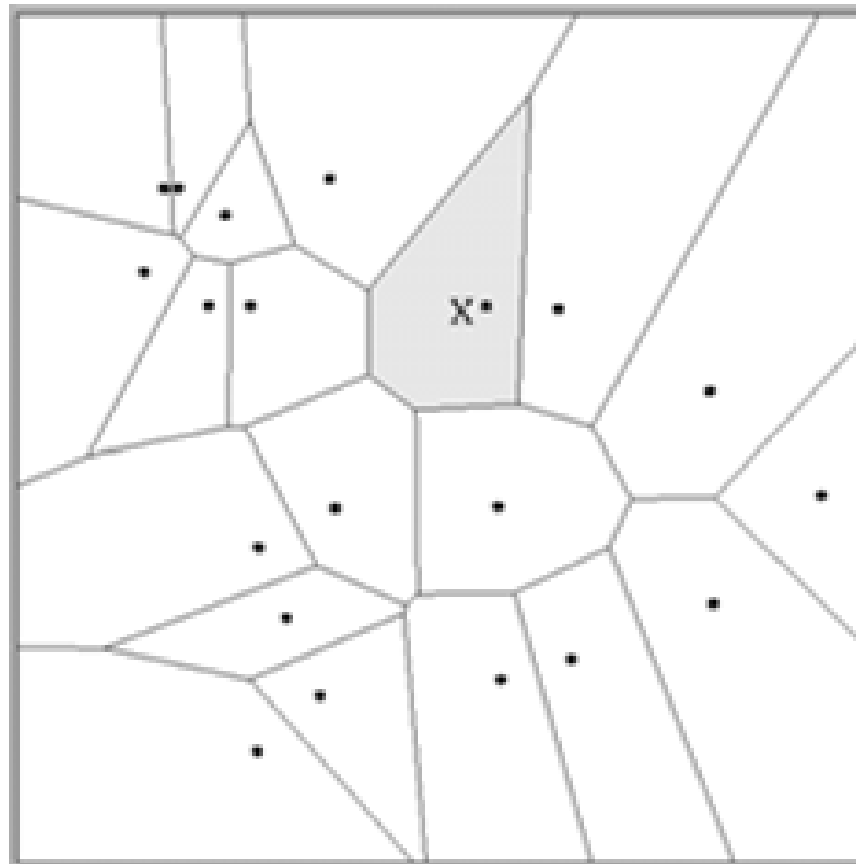


More Concrete Example



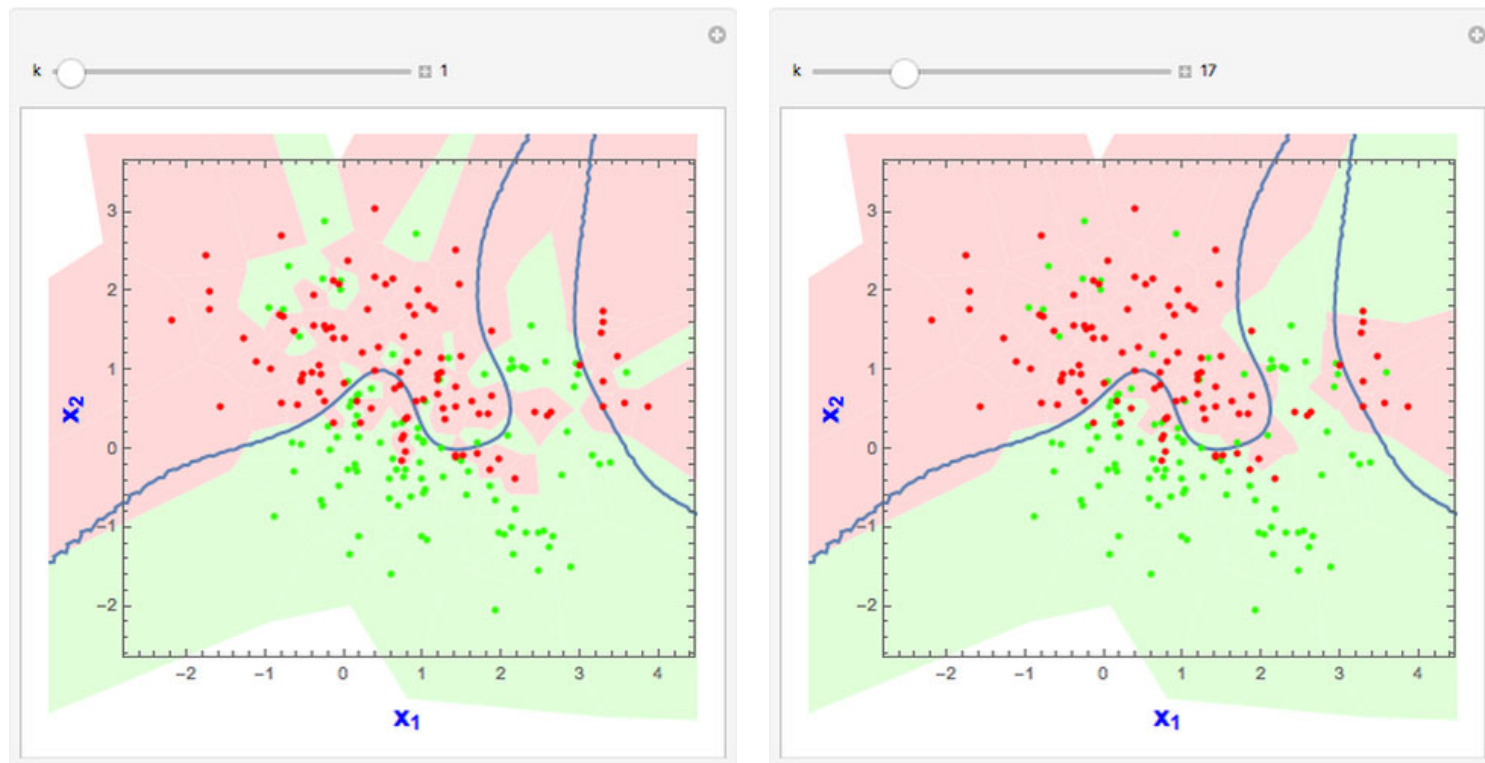
A Two-dimensional Example

- Voronoi diagram



Choosing K

- Making K too small fits the output to the noise in the dataset (overfitting)
- Too large of K can make decision boundaries in classification indistinct (underfitting)
- Choose K empirically using validation set



Weighting Dimensions

3-KNN: Example(1)

Customer	Age	Income	No. credit cards	Class	Distance from John
George	35	35K	3	No	$\text{sqrt} [(35-37)^2+(35-50)^2 +(3-2)^2]=15.16$
Rachel	22	50K	2	Yes	$\text{sqrt} [(22-37)^2+(50-50)^2 +(2-2)^2]=15$
Steve	63	200K	1	No	$\text{sqrt} [(63-37)^2+(200-50)^2 +(1-2)^2]=152.23$
Tom	59	170K	1	No	$\text{sqrt} [(59-37)^2+(170-50)^2 +(1-2)^2]=122$
Anne	25	40K	4	Yes	$\text{sqrt} [(25-37)^2+(40-50)^2 +(4-2)^2]=15.74$
John	37	50K	2	YES	

KNN Classification - Distance

Age	Loan	Default	Distance
25	\$40,000	N	102000
35	\$60,000	N	82000
45	\$80,000	N	62000
20	\$20,000	N	122000
35	\$120,000	N	22000
52	\$18,000	N	124000
23	\$95,000	Y	47000
40	\$62,000	Y	80000
60	\$100,000	Y	42000
48	\$220,000	Y	78000
33	\$150,000	Y	8000
48	\$142,000	?	

Euclidean Distance

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

KNN Classification - Standardized Distance

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	0.61	?	

Standardized Variable

$$X_s = \frac{X - Min}{Max - Min}$$

Properties of k-Nearest Neighbors

Pros:

- Robust to noise
- Very effective when training data is sufficient
- Customized to each query
- Can handle complex decision boundaries and functions by considering the query instance when deciding how to generalize
- Easily adapts when new training data is added

Cons:

- How to weight different dimensions or identify irrelevant dimensions?
- Computationally expensive to label a new query
- Evaluation time grows with the dataset size
- High space requirements (must retain each training example)
- No parameterized model