# Instance Based Learning

Bùi Tiến Lên

2022

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

# Contents

# Notation

| symbol | meaning |
|---|---|
| $a, b, c, N \ldots$ | scalar number |
| $\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y} \ldots$ | column vector |
| $\boldsymbol{X}, \boldsymbol{Y} \ldots$ | matrix |
| $\mathbb{R}$ | set of real numbers |
| $\mathbb{Z}$ | set of integer numbers |
| $\mathbb{N}$ | set of natural numbers |
| $\mathbb{R}^D$ | set of vectors |
| $\mathcal{X}, \mathcal{Y}, \ldots$ | set |
| $\mathcal{A}$ | algorithm |

| operator | meaning |
|---|---|
| $\boldsymbol{w}^\mathsf{T}$ | transpose |
| $\boldsymbol{X}\boldsymbol{Y}$ | matrix multiplication |
| $\boldsymbol{X}^{-1}$ | inverse |

# Parametric vs Non-parametric Models

## Parametric Models

- In the models that we have seen, we select a hypothesis space $\mathcal{H}$ and adjust a *fixed set of parameters* $\boldsymbol{w}$ with the training data $\mathcal{D}$
- We assume that the parameters $\boldsymbol{w}$ summarize the training data $\mathcal{D}$ and we can forget about it

$$y = f(\boldsymbol{x}; \boldsymbol{w}) \tag{1}$$

## Non-parametric Models

- A non parametric model is one that can not be characterized by a fixed set of parameters
- A family of non parametric models is **Instance Based Learning**. The function is based on the training data $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_n\}$

$$y = f(\boldsymbol{x}; \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) \tag{2}$$

# Inductive Bias

## Concept 1

In nonparametric model, we assume that ***similar*** *inputs have* ***similar*** *outputs*.

- This is a reasonable assumption: The world is smooth, and functions, whether they are densities, discriminants, or regression functions, change slowly. Similar instances mean similar things.

# Classification

- k-Nearest Neighbor (k-NN)
- Effects of Hyper-parameters

# When To Consider Nearest Neighbor

- Data points $\boldsymbol{x} \in \mathbb{R}^D$
- Less than $D < 20$ attributes
- Lots of training data $\mathcal{D}$

# Nearest Neighbor

## Learning mode

- Store all training examples $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \mid i = 1, ..., N\}$

## Running mode

- **Nearest neighbor**: Given query instance $\boldsymbol{x}_q$, first locate *the nearest neighbhor* $\boldsymbol{x}^{(1)}$, then estimate

$$h(\boldsymbol{x}_q) = y^{(1)} \tag{3}$$

- $k$-**Nearest neighbor**: Given $\boldsymbol{x}_q$, take vote among its $k$ *nearest neighbors* $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(k)}\}$

$$h(\boldsymbol{x}_q) = \text{majority vote}\{y^{(1)}, y^{(2)}, ..., y^{(k)}\} \tag{4}$$

# Distance

Some common distances in space $\mathbb{R}^D$

- The Minkowski distance of order $p > 0$

$$d(\boldsymbol{x}, \boldsymbol{y}) = L_p(\boldsymbol{x}, \boldsymbol{y}) = \left( \sum_{i=1}^{D} |x_i - y_i|^p \right)^{1/p} \tag{5}$$

- Euclidean distance (popular)

$$d(\boldsymbol{x}, \boldsymbol{y}) = L_2(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{D} (x_i - y_i)^2} \tag{6}$$

# Distance (cont.)

- Manhattan distance

$$d(\boldsymbol{x}, \boldsymbol{y}) = L_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{D} |x_i - y_i| \tag{7}$$

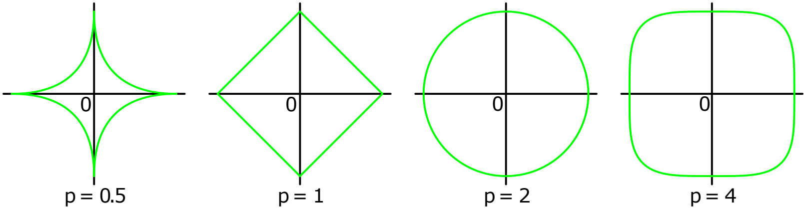
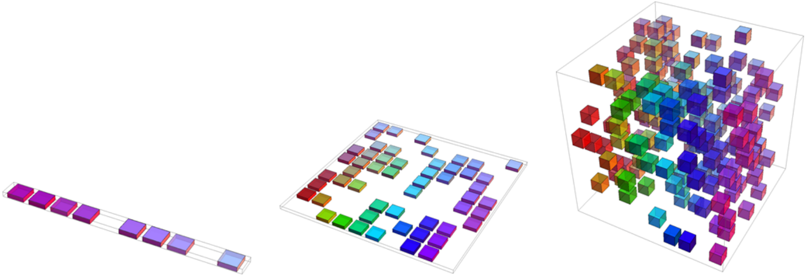



**Figure 1:** Contours of the distance from the origin O for various values of the parameter $p$

# The Curse of dimensionality

- The more dimensions we have, the more examples we need
- The number of examples that we have in a volume of space *decreases exponentially* with the number of dimensions
  - If the number of dimensions is very high, the nearest neighbours can be very far away



# Analysis

## Advantages

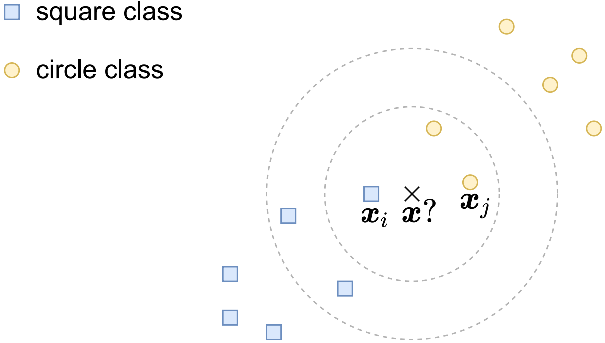
- No training, just store data
- Learn complex target functions
- Don't lose information

## Disadvantages

- Slow at query time
- Easily fooled by irrelevant attributes

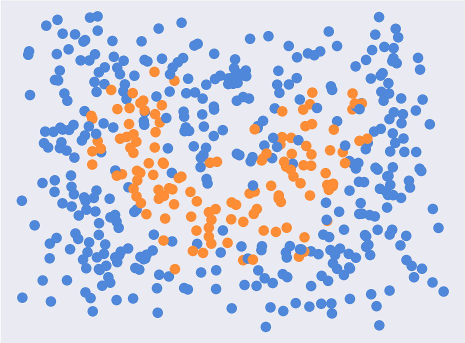
# Parameter k

- if $k = 1$ the *cross point* $\boldsymbol{x}$ should be classified to *square class*
- if $k = 3$ ?
- if $k = 5$ ?



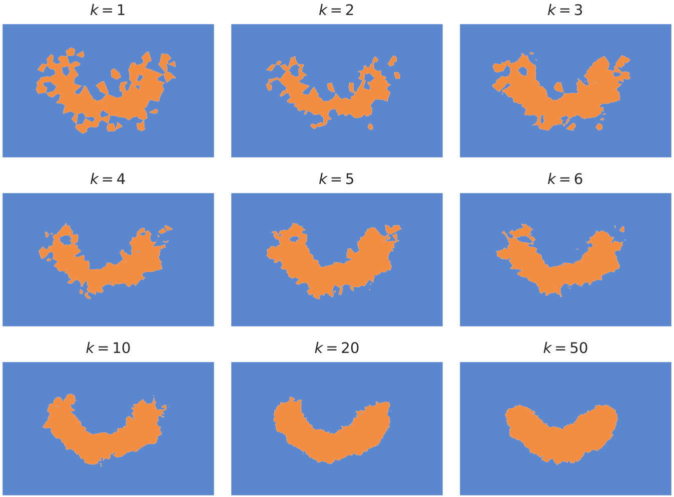
# Parameter k (cont.)

- Data set $\mathcal{D}$ with 500 samples belonging to two classes $\{$blue, orange$\}$



# Parameter k (cont.)

- Decision regions for various values of $k$

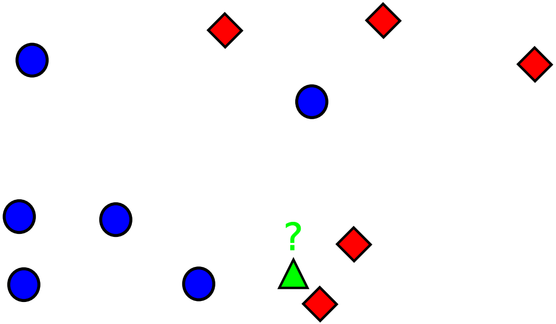


# Metric Learning

- Motivation
- Metric Learning
- Loss Function

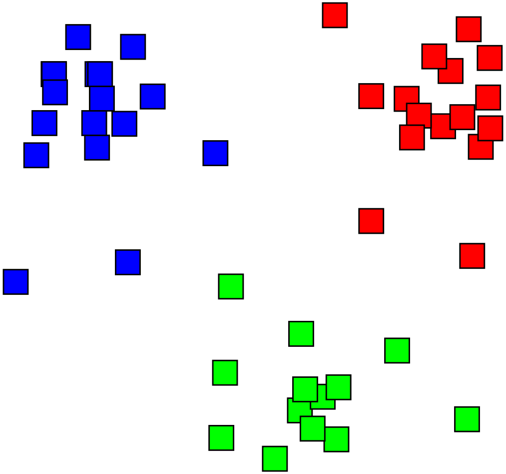
# Motivation

- Nearest neighbor classification



# Motivation (cont.)

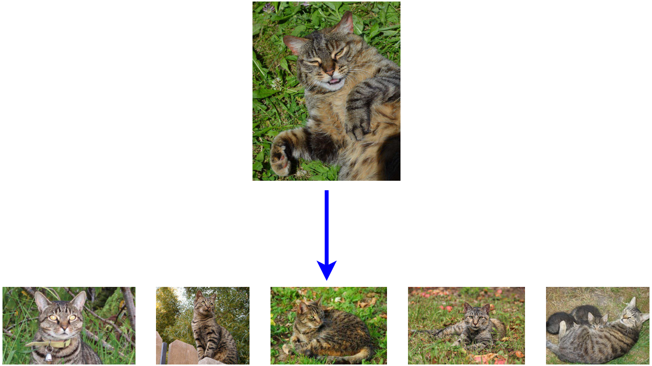
- Clustering



# Motivation (cont.)

- Information retrieval

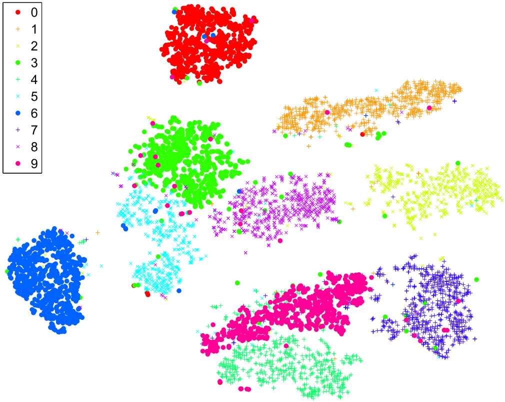
Query image



Most similar images

# Motivation (cont.)

- Data visualization



# Metric Learning

- Given a set of data points $\mathcal{X}$ and their corresponding labels $\mathcal{Y}$
- Select a parametric distance or similarity function

$$d_W(\mathbf{x}, \mathbf{x}') = L\left(f_W(\mathbf{x}), f_W(\mathbf{x}')\right) \tag{8}$$

  - An embedding function (parametric function)

$$f_W(\mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}^n \tag{9}$$

  - A distance function (which is usually fixed beforehand)

$$L(\mathbf{x}, \mathbf{x}') : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R} \tag{10}$$

- The goal is to train the parametric distance, so that the combination $d_W(\mathbf{x}, \mathbf{x}')$ produces small values if the labels $y, y' \in \mathcal{Y}$ of the samples $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ are equal, and larger values if they aren't.

# Metric Learning (cont.)

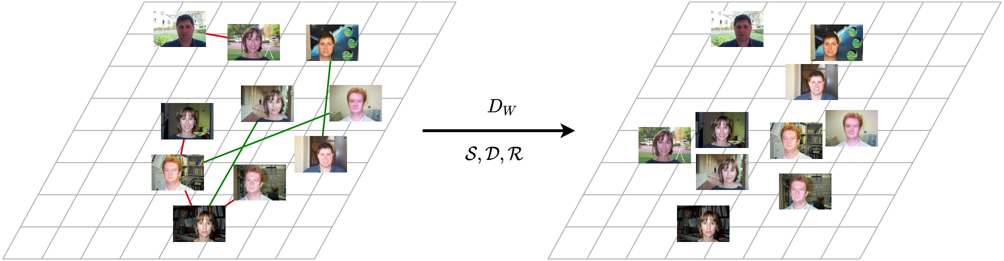
- Collect similarity judgements on data pairs/triplets

$$\begin{aligned} \mathcal{S} &= \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ should be similar}\}, \\ \mathcal{D} &= \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ should be dissimilar}\}. \\ \mathcal{R} &= \{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k) : \boldsymbol{x}_i \text{ should be more similar to } \boldsymbol{x}_j \text{ than to } \boldsymbol{x}_k\}. \end{aligned} \tag{11}$$

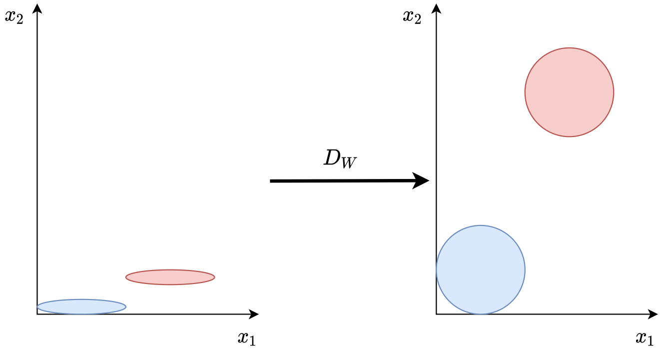
- Estimate parameters s.t. metric best agrees with judgements

$$\hat{W} = \arg\min_{W} \left[ \underbrace{\ell(d_W, \mathcal{S}, \mathcal{D}, \mathcal{R})}_{\text{loss function}} + \underbrace{\lambda R(W)}_{\text{regularization}} \right] \tag{12}$$

# Metric Learning (cont.)



# Metric Learning (cont.)



# Contrastive Approaches

- An embedding function is usually a neural network
- A distance function is $L_2$ distance
- A loss function

# Contrastive Loss

**Contrastive Loss** (Chopra et al. 2005)

- Let $\boldsymbol{x}_1, \boldsymbol{x}_2$ be some samples in the dataset, and $y_1, y_2$ are their corresponding labels. Also, for some condition $A$, let's denote $\mathbb{I}_A$ as the identity function that is equal to 1 if $A$ is true, and 0 otherwise. The loss function is then defined as follows:

$$\ell_{\text{contrast}} = \mathbb{I}_{y_1=y_2} d_W(\boldsymbol{x}_1, \boldsymbol{x}_2) + \mathbb{I}_{y_1 \neq y_2} \max\left(0, \alpha - d_W(\boldsymbol{x}_1, \boldsymbol{x}_2)\right) \tag{13}$$

where $\alpha$ is the margin.

# Triplet Loss

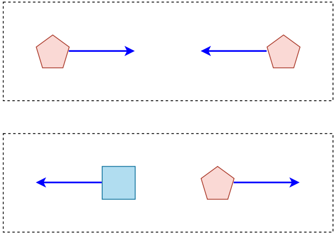
**Triplet Loss** (Schroff et al. 2015)

- Let $\boldsymbol{x}_a, \boldsymbol{x}_p, \boldsymbol{x}_n$ be some samples from the dataset and $\boldsymbol{y}_a, \boldsymbol{y}_p, \boldsymbol{y}_n$ be their corresponding labels, so that $y_a = y_p$ and $y_a \neq y_n$. Usually, $\boldsymbol{x}_a$ is called **anchor** sample, $\boldsymbol{x}_p$ is called **positive** sample because it has the same label as $\boldsymbol{x}_a$, and $\boldsymbol{x}_n$ is called **negative** sample because it has a different label. It is defined as:

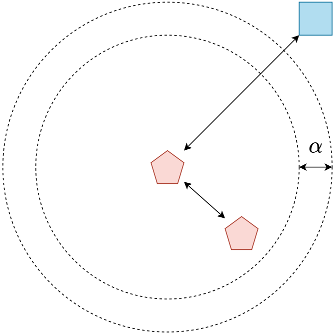
$$\ell_{\text{triplet}} = \max\left(0, d_W(\boldsymbol{x}_a, \boldsymbol{x}_p) - d_W(\boldsymbol{x}_a, \boldsymbol{x}_n) + \alpha\right) \tag{14}$$

where $\alpha$ is the margin.

# Contrastive Loss vs. Triplet Loss



contrastive lost




triplet lost

# Regression

- Kernel Function
- Kernel Regression
- k-NN Regression
- Nadaraya-Watson Model
- Nadaraya-Watson Parametric Model

# Feature Space

Project the data into a **higher dimensional space (feature space)** $\mathcal{F}$

- **Transformation function**

$$\begin{aligned} \phi \quad : \quad \mathbb{R}^D &\rightarrow \quad \mathcal{F} \\ \boldsymbol{x}_i &\rightarrow \phi(\boldsymbol{x}_i) \end{aligned} \tag{15}$$

- Work with $\phi(\boldsymbol{x}_i)$ instead of working with $\boldsymbol{x}_i$.

# The Kernel Function

## Concept 2

A **kernel** is a function $k(\boldsymbol{x}, \boldsymbol{z})$ which represents a dot product in a "hidden" feature space of $\phi$.

$$k(\boldsymbol{x}, \boldsymbol{z}) = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{z}) \tag{16}$$

- **Note that**: we have only dot products $\phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$ to compute; however, this could be very expensive in a high dimensional space.
- **Kernel trick**:

$$\text{instead of } \phi(\boldsymbol{x}) = \phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}, \text{ use } k(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x} \cdot \boldsymbol{z})^2$$

# Common Kernels

- Polynomial:

$$k(\boldsymbol{x}, \boldsymbol{z}) = (u\boldsymbol{x} \cdot \boldsymbol{z} + v)^p \; (u \in \mathbb{R}, v \in \mathbb{R}, p \in \mathbb{N}) \tag{17}$$

- Gaussian:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{\sigma^2}\right) \;, \sigma \in \mathbb{R}^+ \tag{18}$$

**Note**: feature space is infinite-dimensional

# Techniques for Construction of Kernels

In all the following, $k_1, k_2, ..., k_j$ are assumed to be valid kernel functions

1. **Scalar multiplication**: The validity of a kernel is conserved after multiplication by a positive scalar, i.e., for any $\alpha > 0$, the function

$$k(\boldsymbol{x}, \boldsymbol{z}) = \alpha k_1(\boldsymbol{x}, \boldsymbol{z}) \tag{19}$$

2. **Adding a positive constant**: For any positive constant $\alpha > 0$, the function

$$k(\boldsymbol{x}, \boldsymbol{z}) = \alpha + k_1(\boldsymbol{x}, \boldsymbol{z}) \tag{20}$$

# Techniques for Construction of Kernels (cont.)

3. **Linear combination**: A linear combination of kernel functions involving only positive weights, i.e.,

$$k(\boldsymbol{x}, \boldsymbol{z}) = \sum_{j=1}^{m} \alpha_j k_j(\boldsymbol{x}, \boldsymbol{z}), \qquad \text{with } \alpha_j > 0 \tag{21}$$

is a valid kernel function.

4. **Product**: The product of two kernel functions, i.e.,

$$k(\boldsymbol{x}, \boldsymbol{z}) = k_1(\boldsymbol{x}, \boldsymbol{z}) k_2(\boldsymbol{x}, \boldsymbol{z}) \tag{22}$$

is a valid kernel function.

# Techniques for Construction of Kernels (cont.)

5. **Polynomial functions of a kernel output**: Given a polynomial $f : \mathbb{R} \to \mathbb{R}$ with positive coefficients, the function

$$k(\boldsymbol{x}, \boldsymbol{z}) = f(k_1(\boldsymbol{x}, \boldsymbol{z})) \tag{23}$$

is a valid kernel function.

6. **Exponential function of a kernel output**: The function

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp(k_1(\boldsymbol{x}, \boldsymbol{z})) \tag{24}$$

is a valid kernel function.

7. **Product of matrix** and **vectors**:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{x}^\intercal A \boldsymbol{z} \tag{25}$$

where $A$ is a symmetric positive semidefinite matrix.

# Linear Regression Revisted

**Problem**: Given a dataset of input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, find the best linear regresion

- **Primal form**

$$\hat{y} = f(\boldsymbol{x}) = \sum_{i=1}^{D} w_i x_i \tag{26}$$

where

$$\boldsymbol{w} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I}_D)^{-1} \boldsymbol{X}^\intercal \boldsymbol{y} \tag{27}$$

- **Dual Form**

$$\hat{y} = f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i \boldsymbol{x}_i^\intercal \boldsymbol{x} \tag{28}$$

where

$$\boldsymbol{\alpha} = (\boldsymbol{X}\boldsymbol{X}^\intercal + \lambda \boldsymbol{I}_N)^{-1} \boldsymbol{y} \tag{29}$$

# The Kernel Trick

- **Question**: How introduce nonlinearity to

$$\hat{y} = f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i \boldsymbol{x}_i^\top \boldsymbol{x}$$

- **Solution**: Replace the inner product $\boldsymbol{x}_i^\top \boldsymbol{x}$ by $k(\boldsymbol{x}, \boldsymbol{x}_i)$, we have

$$\hat{y} = f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) \tag{30}$$

# Kernel Method

1. Select a kernel function $k(\cdot, \cdot)$
2. Construct a kernel matrix $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ where

$$[\boldsymbol{K}]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{31}$$

3. Compute the coefficients $\boldsymbol{\alpha} \in \mathbb{R}^N$, with

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I}_N)^{-1} \boldsymbol{y} \tag{32}$$

4. Estimate the predicted value for a new sample $\boldsymbol{x}$

$$\hat{y} = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) \tag{33}$$

# Linear Regression vs. Kernel Method

| **Linear regression** | **Kernel method** |
|---|---|
| pick a global model, best fit globally | pick a local model, best fit locally |
| based on the columns (features) | based on the rows (samples) |
| handle linearity | handle nonlinearity |

# k-NN Regression

- **Problem**: Given a dataset of input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, how to learn $f$ to predict the output $\hat{y} = f(\boldsymbol{x})$ for any new input $\boldsymbol{x}$?
- **Solution**: Take the mean of the values of $k$ *nearest neighbors* $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(k)}\}$

$$\hat{y} = \frac{\sum_{i=1}^{k} y^{(i)}}{k} \tag{34}$$

# Nadaraya-Watson Model

- **Problem**: Given a dataset of input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, how to learn $f$ to predict the output $\hat{y} = f(\boldsymbol{x})$ for any new input $\boldsymbol{x}$?
- **Solution**: Consider $(\boldsymbol{x}_i, y_i)$ as a pair of key-value and $x$ as query

| **key** | **value** |
|---|---|
| $\boldsymbol{x}_1$ | $y_1$ |
| $\vdots$ | $\vdots$ |
| $\boldsymbol{x}_N$ | $y_N$ |

$$\hat{y} = \sum_{i=1}^{N} \alpha(\boldsymbol{x}, \boldsymbol{x}_i) y_i, \tag{35}$$

# Nadaraya-Watson Model (cont.)

- We define $\alpha$ using a Gaussian kernel

$$\alpha(\boldsymbol{x}, \boldsymbol{x}_i) = \frac{\exp\left[-\frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}_i\right\|^2\right]}{\sum_{j=1}^{n} \exp\left[-\frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}_j\right\|^2\right]}. \tag{36}$$

and plug it into equation (17)

$$\begin{aligned} \hat{y} &= \sum_{i=1}^{N} \alpha(\boldsymbol{x}, \boldsymbol{x}_i) y_i \\ &= \sum_{i=1}^{N} \frac{\exp\left[-\frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}_i\right\|^2\right]}{\sum_{j=1}^{N} \exp\left[-\frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}_j\right\|^2\right]} y_i \end{aligned} \tag{37}$$

# Nadaraya-Watson Model (cont.)

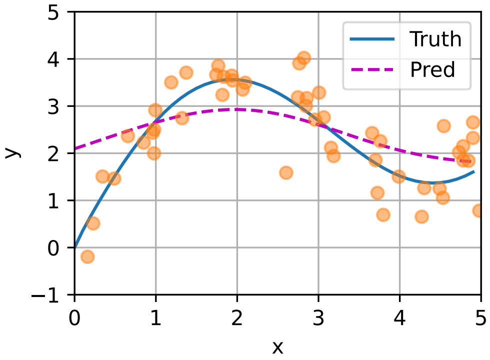
- A key $\boldsymbol{x}_i$ that is closer to the given query $\boldsymbol{x}$ will get more attention via a larger attention weight assigned to the key's corresponding value $y_i$.

# Example 1

- Generate an artificial dataset including 50 training examples and 50 testing examples according to the following nonlinear function with the noise term $\epsilon \sim \mathcal{N}(0, 0.5)$

$$y = 2\sin(x) + x^{0.8} + \epsilon \tag{38}$$

- Find the kernel regression



# Nadaraya-Watson Parametric Model

- Kernel regression enjoys the consistency benefit: given enough data this model converges to the optimal solution.
- Nonetheless, we can easily integrate learnable parameters.
- In the following the distance between the query $\boldsymbol{x}$ and the key $\boldsymbol{x}_i$ is multiplied a learnable parameter $w$:

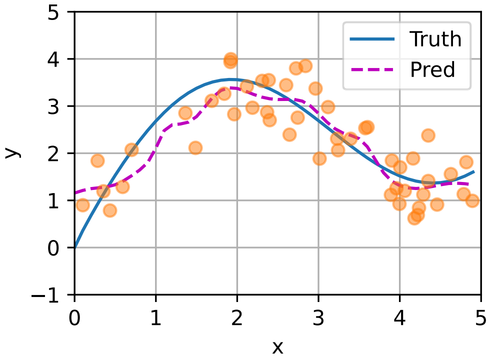
$$\hat{y} = \sum_{i=1}^{N} \frac{\exp\left[-\frac{1}{2}\left(\|\boldsymbol{x} - \boldsymbol{x}_i\| \, w\right)^2\right]}{\sum_{j=1}^{N} \exp\left[-\frac{1}{2}\left(\|\boldsymbol{x} - \boldsymbol{x}_j\| \, w\right)^2\right]} y_i \tag{39}$$

# Example 2

Generate an artificial dataset including 50 training examples and 50 testing examples according to the following nonlinear function with the noise term $\epsilon \sim \mathcal{N}(0, 0.5)$

$$y = 2\sin(x) + x^{0.8} + \epsilon \tag{40}$$

- Find the parametric kernel regression



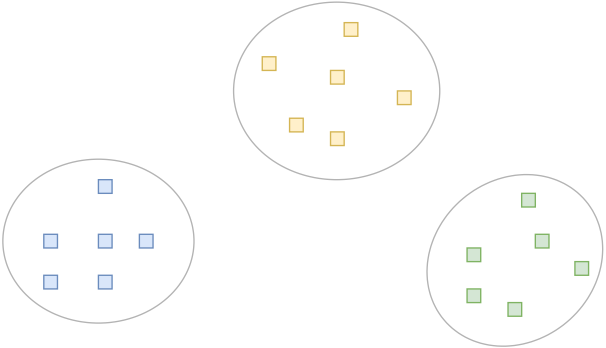
# Clustering

- k-Means
- Hierarchical Clustering
- k-d Tree

# Clustering

## Concept 3

**Cluster analysis** or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense) to each other than to those in other groups (clusters).



# k-Means

## Concept 4

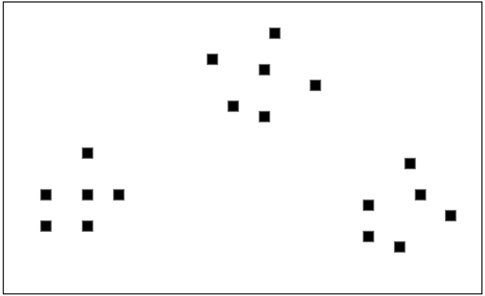
Given a set of observations $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, $k$-means clustering aims to partition the $N$ observations into $k$ $(\leq N)$ sets $\boldsymbol{S} = \{S_1, S_2, ..., S_k\}$ so as to minimize the within-cluster sum of squares

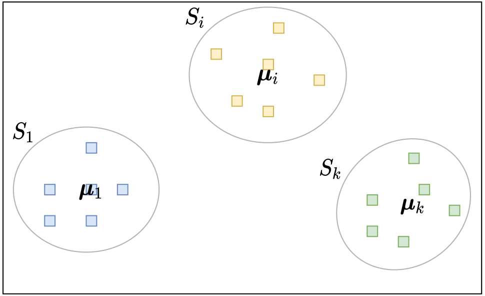
- The objective to find

$$\arg\min_{\boldsymbol{S}} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in S_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2 \tag{41}$$

where $\boldsymbol{\mu}_i$ is the mean of $S_i$

# Illustration





# Naive k-Means Algorithm

1. Initialise a set of $k$ means $\boldsymbol{m}_1^{(0)}, \dots, \boldsymbol{m}_k^{(0)}$
2. For $t = 1, 2, 3, \dots$ do
   - **Assignment step**: Assign each observation to the cluster with the nearest mean: that with the least squared Euclidean distance

$$S_i^{(t)} = \left\{ \boldsymbol{x} \mid L_2(\boldsymbol{x}, \boldsymbol{m}_i^{(t)}) < L_2(\boldsymbol{x}, \boldsymbol{m}_j^{(t)}), \forall j \neq i \right\} \tag{42}$$

   - **Update step**: Recalculate means (centroids) for observations assigned to each cluster.

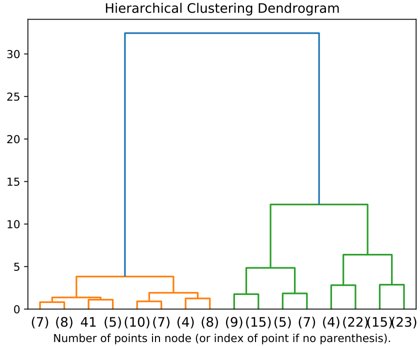
$$\boldsymbol{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\boldsymbol{x} \in S_i^{(t)}} \boldsymbol{x} \tag{43}$$

The algorithm has converged when the assignments no longer change

# Hierarchical Clustering

## Concept 5

**Hierarchical clustering** is a method of cluster analysis which seeks to build a hierarchy of clusters.



# Linkage Function

## Concept 6

A **linkage function** $L$ is used to calculate the distance (similarity/dissimilarity) between arbitrary subsets of the instance space, given a distance metric $d$

- *Single linkage*: defines the distance between two clusters as the smallest pairwise distance between elements from each cluster.

$$L_{single}(A, B) = \min\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in A, \mathbf{y} \in B\} \tag{44}$$

- *Complete linkage*: defines the distance between two clusters as the largest pointwise distance.

$$L_{complete}(A, B) = \max\{d(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in A, \mathbf{y} \in B\} \tag{45}$$

# Agglomerative algorithm

- Given a set of observations $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$

*Initialise* clusters to singleton data points
*Create* a leaf node for every singleton cluster
**Repeat**
  *find* the pair of clusters $X$, $Y$ with lowest linkage
  *merge* $X$, $Y$ into $Z$
  *create* a node for $Z$ (parent node of $X$, $Y$)
**Until** all data points are in one cluster
**Return** the constructed binary tree

# k-d Tree

- The fundamental problem of k-NN is that distance computation is costly and the total cost unavoidably linear in the number of points compared.
- To increase the processing speed, it is possible to partition the data space and reduce this number significantly using k-d tree

## Concept 7

A **k-d tree** (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space

# Algorithm

## Construct k-d tree

- Given and $D$-dimensional dataset $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$
- **Cut** data with a plane at its **median value** along that dimension
- **Recurse** this procedure to create a balanced binary tree k-d tree

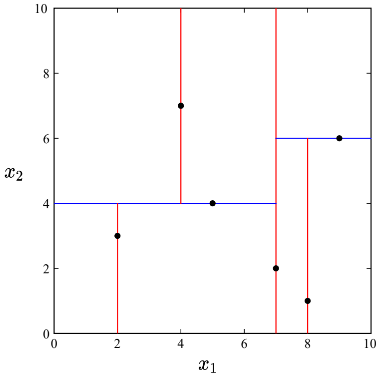
## Nearest neighbor search

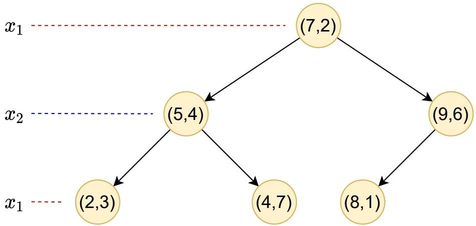
- To locate the NN of an query vector $\boldsymbol{x}$, determine which leaf cell it lies within
- To perform an exhaustive search within this cell.

# Example

Given a dataset $\mathcal{D} = \{(x_1, x_2)\} = \{(2,3),(5,4),(9,6),(4,7),(8,1),(7,2)\}$

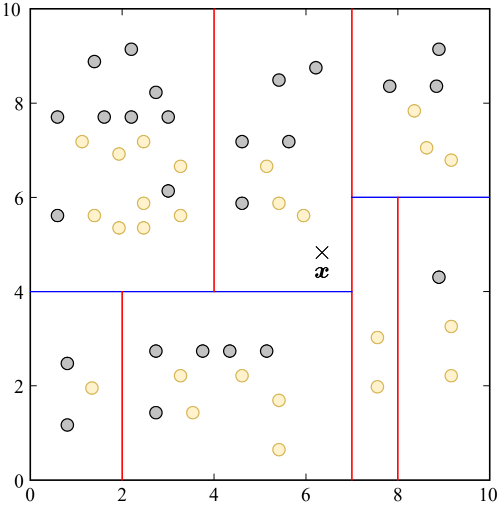
- Construct k-d tree



# Example (cont.)

- Nearest neighbor search



# References

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
*Deep learning*.
MIT press.

Lê, B. and Tô, V. (2014).
*Cở sở trí tuệ nhân tạo*.
Nhà xuất bản Khoa học và Kỹ thuật.

Russell, S. and Norvig, P. (2021).
*Artificial intelligence: a modern approach*.
Pearson Education Limited.