Preliminaries & kNN

Ninghao Liu

University of Georgia

January 10, 2024

Some contents adopted from Stanford CS221, Percy Liang.

Example: Spam Email Detection

Input: x = an email message

From: ninghao.liu@uga.edu
Date: January 05, 2022
Subject: CSCI 4380/6380 announcement

Hello students,

Welcome to CSCI 4380/6380. Here is ...

From: a9k62n@hotmail.com
Date: September 25, 2019
Subject: URGENT

Dear Sir or maDam:

my friend left sum of 10m dollars...

Figure 1: Two email messages.

Output: $y = \{\text{spam, non-spam}\}$
Goal: Build a predictor f.

$$x \longrightarrow \boxed{f} \longrightarrow y$$

Types of Prediction Tasks

- Binary classification (e.g., email $\rightarrow$ spam/not spam):

$$x \longrightarrow \boxed{f} \longrightarrow y \in \{+1, -1\}$$

- Regression (e.g., location, year $\rightarrow$ housing price):

$$x \longrightarrow \boxed{f} \longrightarrow y \in \mathbb{R}$$

(Note: Not a formal dichotomy.)

Types of Prediction Tasks

- Multi-class classification:

 $\longrightarrow$ $\boxed{f}$ $\longrightarrow$ cat

- Ranking:

$\boxed{1}\ \boxed{2}\ \boxed{3}\ \boxed{4}$ $\longrightarrow$ $\boxed{f}$ $\longrightarrow$ 2 3 4 1

- Structured prediction:

la casa blu $\longrightarrow$ $\boxed{f}$ $\longrightarrow$ *the blue house*

Data

Instance/Sample: y is the ground-truth output for x.

$$(x, y)$$

Dataset/Data: Set of instances, which **partially specifies** of the desired behavior of a predictor.

$$\mathcal{D} = [$$
$$(\text{"...10m dollars..."}, +1),$$
$$(\text{"...CSCI 4380..."}, -1),$$
$$]$$

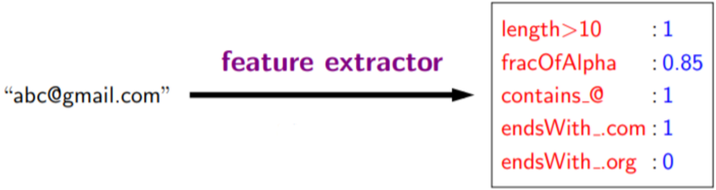
Supervised learning (labels are given) vs **Unsupervised learning** (labels not given).

Feature Extraction

Example task: Predict y, whether a string x is an email address.

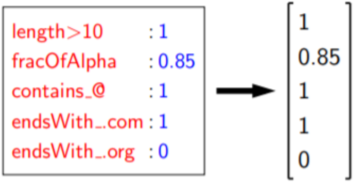
Question: What properties of x might be relevant for predicting y?

Feature extraction: Given input x, produce a set of (feature name, feature value) pairs.



Feature Extraction

In mathematical formulation, a feature vector actually does not need feature names:



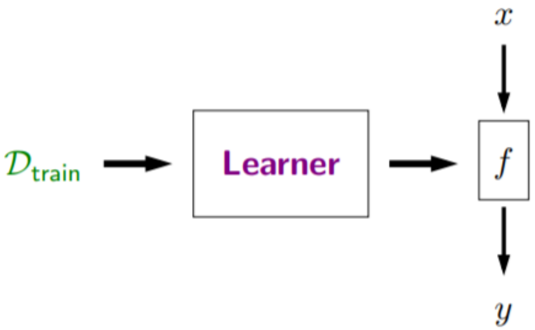
For an input x, its **feature vector** is:

$$\phi(x) = [\phi(x)_1, \phi(x)_2, ..., \phi(x)_D]. \tag{1}$$

We can think of $\phi(x) \in \mathbb{R}^D$ as a point in a D-dimensional space.

- Sometimes we also simply write $\phi(x)$ as $\boldsymbol{x}$ to denote the vector.

The Learning Framework



We want the f to work even for instances that we have not seen in $\mathcal{D}_{train}$.

- Generalization.

k-nearest neighbors algorithm (kNN)

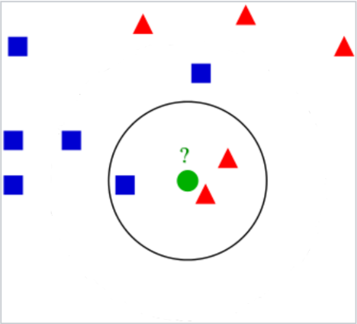
An object is classified by a plurality vote of its neighbors, with the instance being assigned to the class most common among its k nearest neighbors [1].

- If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.

[1] `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`

k-nearest neighbors algorithm (kNN)

An object is classified by a plurality vote of its neighbors, with the instance being assigned to the class most common among its k nearest neighbors [2].

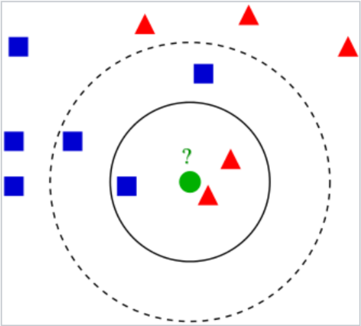


[2] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

k-nearest neighbors algorithm (kNN)

1. Choose the Number of Neighbors (k).
2. Calculate Distance: For each point in the dataset, calculate the distance between that point and the query point for which you're trying to predict a label or value.
3. Find Nearest Neighbors: Identify the k points in the dataset that are closest to the query point.
4. For Classification: Count the number of data points in each category among the k nearest neighbors. Assign the new data point to the category that is most frequent among its k nearest neighbors.

k-nearest neighbors algorithm (kNN)



- How to choose k?
- Why couldn't we simply apply kNN to all the problems?
- How to decide if two instances are near with each other?