# COMP4131: Data Modelling and Analysis

## Lecture 8: Naive Bayes and K Nearest Neighbors

Daokun Zhang

University of Nottingham Ningbo China

*daokun.zhang@nottingham.edu.cn*

April 7, 2025

# Overview

# Naive Bayes Classifiers

# Naive Bayes Classification

Suppose we have a set of training samples $\mathcal{T}_r = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, where each sample is described by a $m$-dimensional feature vector $\mathbf{x} = (x_1, x_2, \cdots, x_m)$, and a categorical label $y \in \mathcal{Y} = \{1, 2, \cdots, C\}$, with $C$ being the class number.

For a new test sample with feature vector $\mathbf{x}^*$, we can predict its class label as

$$y^* = \arg\max_{y \in \mathcal{Y}} \mathbb{P}(y|\mathbf{x}^*),$$

where

$$\mathbb{P}(y|\mathbf{x}^*) = \frac{\mathbb{P}(\mathbf{x}^*, y)}{\mathbb{P}(\mathbf{x}^*)} = \frac{\mathbb{P}(\mathbf{x}^*|y)\mathbb{P}(y)}{\mathbb{P}(\mathbf{x}^*)}.$$

As $\mathbb{P}(\mathbf{x}^*)$ is a constant with regard to $y$, then we have

$$y^* = \arg\max_{y \in \mathcal{Y}} \mathbb{P}(\mathbf{x}^*, y) = \arg\max_{y \in \mathcal{Y}} \mathbb{P}(\mathbf{x}^*|y)\mathbb{P}(y).$$

# Naive Bayes Classification

The task becomes to estimate the conditional probability $\mathbb{P}(\mathbf{x}^*|y)$ and prior probability $\mathbb{P}(y)$.

The prior probability can be estimated by counting the number of training samples having the targeting class labels

$$\mathbb{P}(y = c) = \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c\}|}{n},$$

where $|\cdot|$ is the size of a set and $\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c\}$ is the set of samples with class label $y = c$.

To calculate the conditional probability $\mathbb{P}(\mathbf{x}^*|y)$, we can leverage the conditional independence assumption

$$\mathbb{P}(\mathbf{x}^*|y) = \mathbb{P}(x_1^*|y) \cdot \mathbb{P}(x_2^*|y) \cdots \mathbb{P}(x_m^*|y),$$

where $\mathbb{P}(x_j^*|y)$ with $1 \leq j \leq m$ is the occurrence probability of feature value $x_j^*$ conditioned on the class label $y$.

# Naive Bayes Classification

For different data types, $\mathbb{P}(x_j^*|y)$ can be estimated in different ways.

For categorical data, where $x_j^*$ takes one of the pre-defined values $\{f_1, f_2 \cdots, f_{n_j}\}$, we assume $x_j^*|y = c$ follows a Categorical distribution

$$x_j^*|y = c \sim \mathbf{Cat}(n_j, \mathbf{p}),$$

where $\mathbf{p} = (p_1, p_2, \cdots, p_{n_j})$ with $p_k = \mathbb{P}(x_j^* = f_k|y = c)$ and $\sum_{k=1}^{n_j} p_k = 1$. That is to say

$$\mathbb{P}(x_j^* = f_k|y = c) = \prod_{k=1}^{n_j} p_k^{\delta(x_j^*, f_k)},$$

where $\delta(x_j^*, f_k)$ is a Kronecker delta function, whose value is equal to 1 if $x_j^* = f_k$ and 0 otherwise.

# Naive Bayes Classification

The parameter $p_k$ $(1 \leq k \leq n_j)$ can be estimated by the maximum likelihood estimation (MLE) method, for which the log likelihood is formulated as

$$\begin{aligned}
\log L(\mathbf{p}) &= \log \prod_{(\mathbf{x},y)\in\mathcal{T}_r : y=c} \mathbb{P}(x_j | y = c) \\
&= \log \prod_{(\mathbf{x},y)\in\mathcal{T}_r : y=c} \left\{ \prod_{s=1}^{n_j} p_s^{\delta(x_j, f_s)} \right\} \\
&= \sum_{(\mathbf{x},y)\in\mathcal{T}_r : y=c} \left\{ \sum_{s=1}^{n_j} \delta(x_j, f_s) \log p_s \right\}
\end{aligned}$$

subject to the constraint

$$\sum_{s=1}^{n_j} p_s = 1.$$

The Lagrangian function with the constraint than has the following form

$$
\begin{aligned}
\mathcal{L}(\mathbf{p}) &= \log L(\mathbf{p}) + \lambda \left( \sum_{s=1}^{n_j} p_s - 1 \right) \\
&= \sum_{(\mathbf{x},y)\in\mathcal{T}_r : y=c} \left\{ \sum_{s=1}^{n_j} \delta(x_j, f_s) \log p_s \right\} + \lambda \left( \sum_{s=1}^{n_j} p_s - 1 \right).
\end{aligned}
$$

The derivative of $\mathcal{L}(\mathbf{p})$ w.r.t $p_k$ is

$$
\frac{\partial \mathcal{L}(\mathbf{p})}{\partial p_k} = \sum_{(\mathbf{x},y)\in\mathcal{T}_r : y=c} \frac{\delta(x_j, f_k)}{p_k} + \lambda.
$$

Set the derivative to zero, we have

$$
p_k = - \sum_{(\mathbf{x},y)\in\mathcal{T}_r : y=c} \frac{\delta(x_j, f_k)}{\lambda}.
$$

# Naive Bayes Classification

Using the property $\sum_{s=1}^{n_j} p_s = 1$,

$$\lambda = -\sum_{s=1}^{n_j} \sum_{(\mathbf{x},y)\in\mathcal{T}_r : y=c} \delta(x_j, f_s).$$

Then

$$p_k = \frac{\sum_{(\mathbf{x},y)\in\mathcal{T}_r : y=c} \delta(x_j, f_k)}{\sum_{s=1}^{n_j} \sum_{(\mathbf{x},y)\in\mathcal{T}_r : y=c} \delta(x_j, f_s)}$$

$$= \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c, x_j = f_k\}|}{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c\}|}.$$

Then we can get the estimation for the probability $\mathbb{P}(x_j = f_k | y = c)$ as

$$\mathbb{P}(x_j = f_k | y = c) = \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c, x_j = f_k\}|}{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c\}|},$$

and the prior probability $\mathbb{P}(y = c)$ as

$$\mathbb{P}(y = c) = \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c\}|}{n}.$$

$|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c, x_j = f_k\}|$ is the number of training samples with label $y$ taking value $c$ and attribute $x_j$ taking value $f_k$, which can also be represented as **Count**$(y = c, x_j = f_k)$.

$|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c\}|$ is the number of training samples with label $y$ taking value $c$, which can also be represented as **Count**$(y = c)$.

# Naive Bayes Classification

To avoid the case that $\mathbb{P}(x_j = f_k | y = c) = 0$ with the number of supported training samples equal 0, we adopt the Laplace smoothing to adjust the probability estimation

$$\mathbb{P}(x_j = f_k | y = c) = \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c, x_j = f_k\}| + \alpha}{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c\}| + n_j \alpha}.$$

where the "pseudocount" $\alpha > 0$ is the smoothing parameter.

# An Alternative Way to Parameter Estimation

It looks like the probability $\mathbb{P}(x_j = f_k | y = c)$ can be estimated in a more straightforward way

1. $|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c, x_j = f_k\}|$ is the occurrence times of the event $\{y = c, x_j = f_k\}$ in training data
2. $|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c\}|$ is the occurrence times of the event $\{y = c\}$ in training data
3. According to the classical definition of probability, we can directly calculate the (unsmoothed) probability as

$$\mathbb{P}(x_j = f_k | y = c) = \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c, x_j = f_k\}|}{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c\}|}.$$

So, why shall we make an assumption on the distribution of attribute values and use MLE to estimate the distribution parameters?

# Extensions

According to the difference in data type, the Categorical Naive Bayes can be extended into the following algorithm variants by modeling the distribution of attribute values

- **Gaussian Naive Bayes**
  - $x_j$ takes continuous values
  - assume $x_j|y$ follows a Gaussian distribution
- **Bernoulli Naive Bayes**
  - $x_j$ takes binary values $\{0, 1\}$, indicating the occurrence status of a word in a text sample
  - assume $x_j|y$ follows a Bernoulli distribution
- **Multinomial Naive Bayes**
  - $x_j$ takes non-negative integer values $\{0, 1, 2, 3, \cdots\}$, indicating the occurrence times of a word in a text sample
  - assume $x_1, x_2, \cdots, x_m|y$ jointly follow a Multinomial distribution

# Categorical Naive Bayes: Example

Suppose we are give a set of training samples, where each sample includes the weather condition record of a day as attributes, and the decision to play tennis or not as a binary class label. Please use Categorical Naive Bayes to predict whether we shall play tennis or not for a new day.

*PlayTennis*: training examples

| Day | Outlook | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

# Categorical Naive Bayes: Example

From the collected training data, we can first summarize the categorical value set for each attribute/label as

- **Attribute**: Outlook $\in$ {Sunny, Rain, Overcast}
- **Attribute**: Temperature $\in$ {Hot, Cool, Mild}
- **Attribute**: Humidity $\in$ {High, Normal}
- **Attribute**: Wind $\in$ {Strong, Weak}
- **Label**: PlayTennis $\in$ {Yes, No}

# Categorical Naive Bayes: Example

We can count the occurrence times of the interested events as

- $\textbf{Count}(\text{PlayTennis} = \text{Yes}) = ?$
- $\textbf{Count}(\text{PlayTennis} = \text{No}) = ?$
- $\textbf{Count}(\text{Outlook} = \text{Sunny}, \text{PlayTennis} = \text{Yes}) = ?$
- $\textbf{Count}(\text{Outlook} = \text{Rain}, \text{PlayTennis} = \text{Yes}) = ?$
- $\textbf{Count}(\text{Outlook} = \text{Overcast}, \text{PlayTennis} = \text{Yes}) = ?$
- $\textbf{Count}(\text{Outlook} = \text{Sunny}, \text{PlayTennis} = \text{No}) = ?$
- $\textbf{Count}(\text{Outlook} = \text{Rain}, \text{PlayTennis} = \text{No}) = ?$
- $\textbf{Count}(\text{Outlook} = \text{Overcast}, \text{PlayTennis} = \text{No}) = ?$
- $\textbf{Count}(\text{Temperature} = \text{Hot}, \text{PlayTennis} = \text{Yes}) = ?$
- $\textbf{Count}(\text{Temperature} = \text{Cool}, \text{PlayTennis} = \text{Yes}) = ?$
- $\textbf{Count}(\text{Temperature} = \text{Mild}, \text{PlayTennis} = \text{Yes}) = ?$

# Categorical Naive Bayes: Example

- $\textbf{Count}(\text{Temperature} = \text{Hot}, \text{PlayTennis} = \text{No}) = ?$
- $\textbf{Count}(\text{Temperature} = \text{Cool}, \text{PlayTennis} = \text{No}) = ?$
- $\textbf{Count}(\text{Temperature} = \text{Mild}, \text{PlayTennis} = \text{No}) = ?$
- $\textbf{Count}(\text{Humidity} = \text{High}, \text{PlayTennis} = \text{Yes}) = ?$
- $\textbf{Count}(\text{Humidity} = \text{Normal}, \text{PlayTennis} = \text{Yes}) = ?$
- $\textbf{Count}(\text{Humidity} = \text{High}, \text{PlayTennis} = \text{No}) = ?$
- $\textbf{Count}(\text{Humidity} = \text{Normal}, \text{PlayTennis} = \text{No}) = ?$
- $\textbf{Count}(\text{Wind} = \text{Strong}, \text{PlayTennis} = \text{Yes}) = ?$
- $\textbf{Count}(\text{Wind} = \text{Weak}, \text{PlayTennis} = \text{Yes}) = ?$
- $\textbf{Count}(\text{Wind} = \text{Strong}, \text{PlayTennis} = \text{No}) = ?$
- $\textbf{Count}(\text{Wind} = \text{Weak}, \text{PlayTennis} = \text{No}) = ?$

# Categorical Naive Bayes: Example

We can calculate the interested probability value as (without the use of Laplace smoothing)

- $\mathbb{P}(\text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Sunny} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Rain} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Overcast} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Sunny} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Rain} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Outlook} = \text{Overcast} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Temperature} = \text{Hot} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Temperature} = \text{Cool} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Temperature} = \text{Mild} \mid \text{PlayTennis} = \text{Yes}) = ?$

# Categorical Naive Bayes: Example

- $\mathbb{P}(\text{Temperature} = \text{Hot} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Temperature} = \text{Cool} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Temperature} = \text{Mild} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Humidity} = \text{High} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Humidity} = \text{Normal} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Humidity} = \text{High} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Humidity} = \text{Normal} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Wind} = \text{Strong} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Wind} = \text{Weak} \mid \text{PlayTennis} = \text{Yes}) = ?$
- $\mathbb{P}(\text{Wind} = \text{Strong} \mid \text{PlayTennis} = \text{No}) = ?$
- $\mathbb{P}(\text{Wind} = \text{Weak} \mid \text{PlayTennis} = \text{No}) = ?$

# Categorical Naive Bayes: Example

Make prediction for a new day

$\mathbf{x}^* = (\text{Outlook} = \text{Sunny}, \text{Temperature} = \text{Cool}, \text{Humidity} = \text{High}, \text{Wind} = \text{Strong})$

We can calculate the joint probability $\mathbb{P}(\mathbf{x}^*, \text{PlayTennis})$ as

$$
\begin{aligned}
\mathbb{P}(\mathbf{x}^*, \text{PlayTennis} = \text{Yes}) =& \mathbb{P}(\text{Outlook} = \text{Sunny} \mid \text{PlayTennis} = \text{Yes}) \cdot \\
& \mathbb{P}(\text{Temperature} = \text{Cool} \mid \text{PlayTennis} = \text{Yes}) \cdot \\
& \mathbb{P}(\text{Humidity} = \text{High} \mid \text{PlayTennis} = \text{Yes}) \cdot \\
& \mathbb{P}(\text{Wind} = \text{Strong} \mid \text{PlayTennis} = \text{Yes}) \cdot \\
& \mathbb{P}(\text{PlayTennis} = \text{Yes}) = \color{red}{?} \\
\mathbb{P}(\mathbf{x}^*, \text{PlayTennis} = \text{No}) =& \mathbb{P}(\text{Outlook} = \text{Sunny} \mid \text{PlayTennis} = \text{No}) \cdot \\
& \mathbb{P}(\text{Temperature} = \text{Cool} \mid \text{PlayTennis} = \text{No}) \cdot \\
& \mathbb{P}(\text{Humidity} = \text{High} \mid \text{PlayTennis} = \text{No}) \cdot \\
& \mathbb{P}(\text{Wind} = \text{Strong} \mid \text{PlayTennis} = \text{No}) \cdot \\
& \mathbb{P}(\text{PlayTennis} = \text{No}) = \color{red}{?}
\end{aligned}
$$

By comparing $\mathbb{P}(\mathbf{x}^*, \text{PlayTennis} = \text{Yes})$ and $\mathbb{P}(\mathbf{x}^*, \text{PlayTennis} = \text{No})$, the label of $\mathbf{x}^*$ is predicted as "$\text{PlayTennis} = \color{red}{?}$".

# Pros of Naive Bayes

- Simple and easy to implement
- Fast and scalable
- Works well with high-dimensional data
- Handles missing data well
- Robust to irrelevant features
- Effective for text classification
- Suitable for multi-class problems
- Provides probabilistic outputs
- Low memory consumption
- Good baseline classifier
- Suitable for binary and categorical data

# Cons of Naive Bayes

- Assumes independence between features
- Can be overly simplistic
- Relatively poor performance with continuous data
- Sensitive to zero frequency problem
- Limited expressiveness for complex problems
- Not ideal for large feature spaces with sparse data
- Requires large amounts of data for accurate probabilities
- Struggles with highly imbalanced data

# K Nearest Neighbors Classifier

# K Nearest Neighbors Classifier

K Nearest Neighbors (KNN) classifier is a non-parametric/lazy classifier, which does not require to train a parametric model from the training data.

KNN classifier classifies unlabeled examples by assigning them the class of the most similar labeled examples.

Formally, given a set of training samples $\mathcal{T}_r = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, where each sample is described by a $m$-dimensional continuous feature vector $\mathbf{x}$ and a class label $y$. For a new test example $\mathbf{x}^*$, KNN determines its label probability as

$$\mathbb{P}(y^* = c|\mathbf{x}^*) = \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c, d(\mathbf{x}^*, \mathbf{x}) \text{ ranks smallest-}K\}|}{K},$$

where $d(\mathbf{x}^*, \mathbf{x})$ is the distance between $\mathbf{x}^*$ and $\mathbf{x}$.

# Distance Metric

For the two samples $\mathbf{x}^*$ and $\mathbf{x}$, there are various choices to measure their distance

- Euclidean distance

$$d(\mathbf{x}^*, \mathbf{x}) = \|\mathbf{x}^* - \mathbf{x}\|_2 = \sqrt{\sum_{j=1}^{m} (x_j^* - x_j)^2}$$

- Manhattan distance

$$d(\mathbf{x}^*, \mathbf{x}) = \|\mathbf{x}^* - \mathbf{x}\|_1 = \sum_{j=1}^{m} |x_j^* - x_j|$$

- Cosine distance

$$d(\mathbf{x}^*, \mathbf{x}) = 1 - \cos(\mathbf{x}^*, \mathbf{x}) = 1 - \frac{\sum_{j=1}^{m} x_j^* x_j}{\sqrt{\sum_{j=1}^{m} {x_j^*}^2}\sqrt{\sum_{j=1}^{m} x_j^2}}$$

# Radius Neighbors Classifier

KNN is only suitable to the case where neighboring samples are uniformly distributed, so that we can use the majority voting by assigning equal weights to the neighboring label references.

For the non-uniform case, the KNN variants, Radius Neighbors and Weighted KNN, would be better choices.

Radius Neighbors Classifier classifies unlabeled test samples by referring to the class of their neighboring samples within a radius in the training set.

The label probability $\mathbb{P}(y^* = c|\mathbf{x}^*)$ for the test sample is estimated as

$$\mathbb{P}(y^* = c|\mathbf{x}^*) = \frac{|\{(\mathbf{x}, y) \in \mathcal{T}_r : y = c, d(\mathbf{x}^*, \mathbf{x}) \leq r\}|}{|\{(\mathbf{x}, y) \in \mathcal{T}_r : d(\mathbf{x}^*, \mathbf{x}) \leq r\}|},$$

where $r$ is the pre-specified radius parameter.

# Weighted KNN

By defining the K Nearest Neighbors of $\mathbf{x}^*$ in the training set as

$$\mathcal{N}(\mathbf{x}^*) = \{(\mathbf{x}, y) \in \mathcal{T}_r : d(\mathbf{x}^*, \mathbf{x}) \text{ ranks smallest-}K\},$$

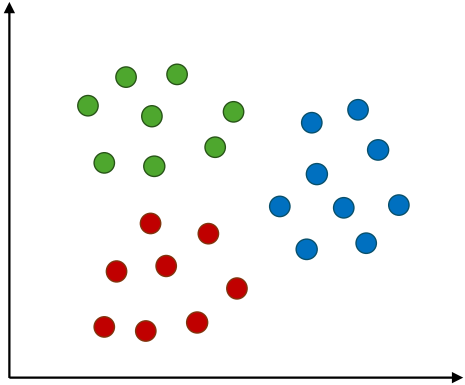
Weighted KNN puts more weights on the neighboring samples closer to the test example $\mathbf{x}^*$ for estimating the probability $\mathbb{P}(y^* = c|\mathbf{x}^*)$

$$\mathbb{P}(y^* = c|\mathbf{x}^*) = \frac{\sum_{(\mathbf{x},y)\in\mathcal{N}(\mathbf{x}^*)} w_{\mathbf{x}} \cdot \delta(y, c)}{\sum_{(\mathbf{x},y)\in\mathcal{N}(\mathbf{x}^*)} w_{\mathbf{x}}},$$

where $\delta(y, c)$ is the Kronecker delta function which is equal to 1 if $y = c$ and 0 otherwise, and $w_{\mathbf{x}}$ is the weight assigned to the neighboring sample $x$, which can be the inverse of the distance between $\mathbf{x}$ and $\mathbf{x}^*$

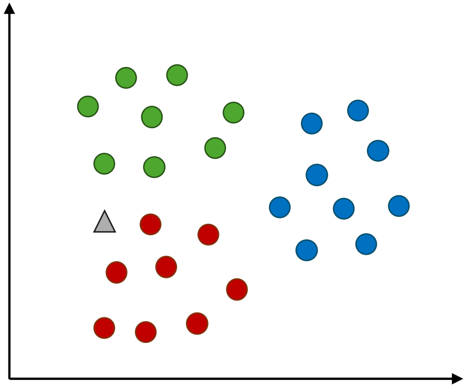
$$w_{\mathbf{x}} = \frac{1}{d(\mathbf{x}^*, \mathbf{x})}.$$

# KNN: Example



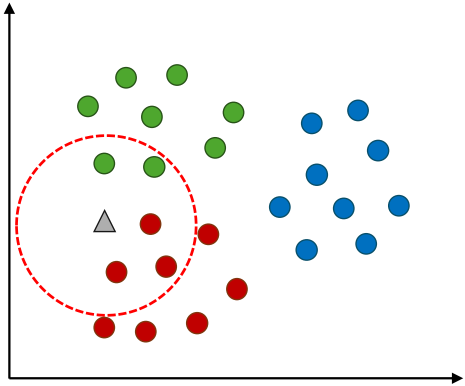
We are given a set of labeled data points in three classes {**Red**,**Green**,**Blue**} as training samples.

# KNN: Example



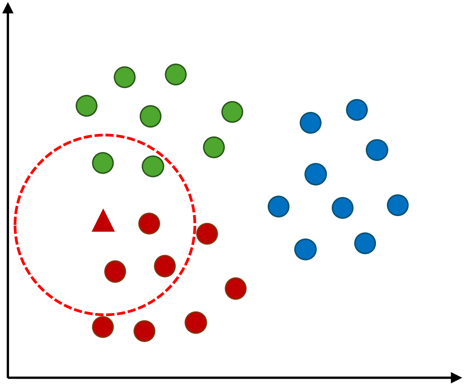
Predict the label of the unlabeled data point represented by the grey triangle.

# KNN: Example



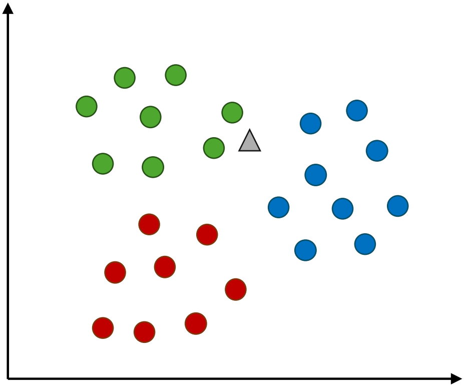
We collect its 5 nearest neighbors with three **Red** data points and two **Green** data points.

# KNN: Example



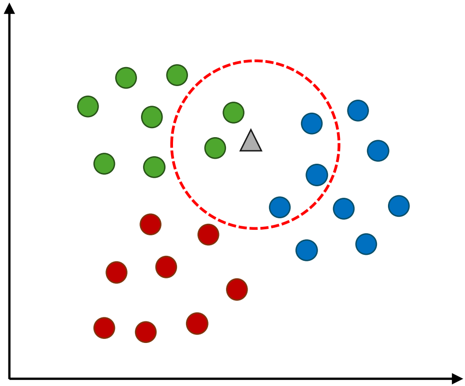
Predict the class of the test data point as **Red**.

# KNN: Example



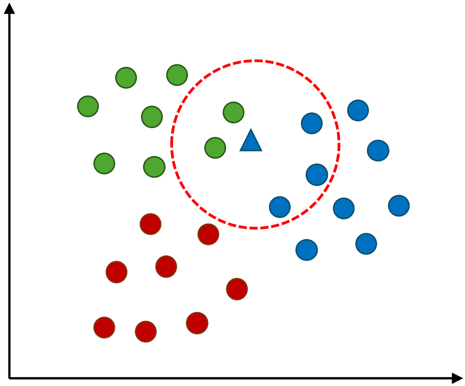
KNN cannot work well for the data points whose K nearest neighbors are not uniformly distributed.

# KNN: Example



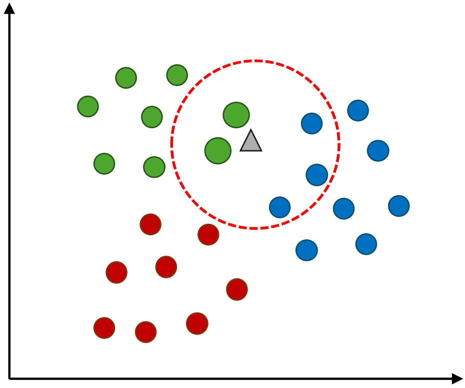
We collect the test data point's 5 nearest neighbors
with three **Blue** data points and two **Green** data points.

# KNN: Example



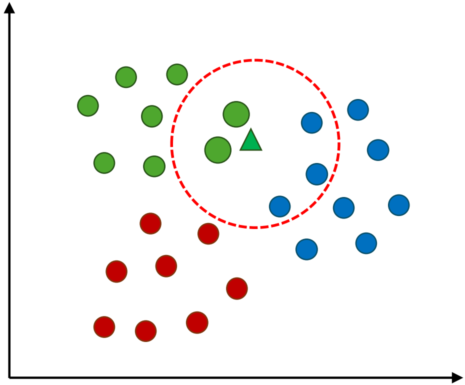
In this case, KNN would unfairly predict the class of the test data point as **Blue**.

# Weighted KNN: Example



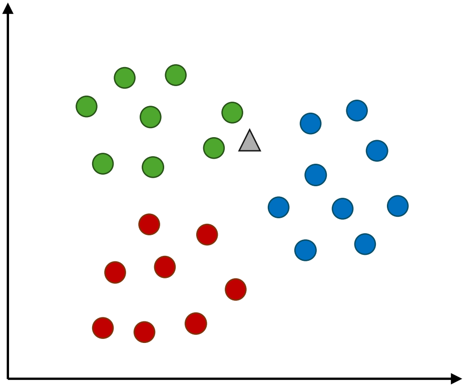
Weighted KNN increases the weights of the closer **Green** neighbors.

# Weighted KNN: Example



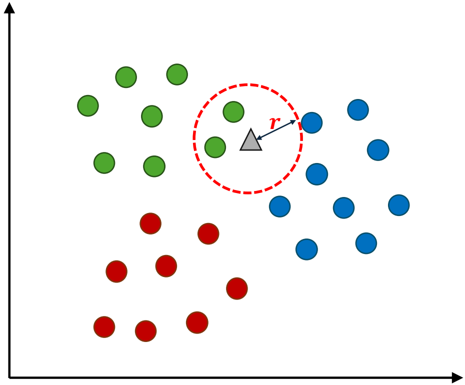
The class of the test data point is then predicted as **Green** fairly.

# Radius Neighbors: Example



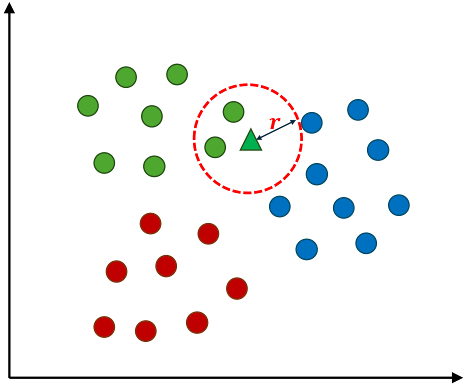
Radius Neighbors Classifier collects neighbors within a fixed radius.

# Radius Neighbors: Example




With radius $r$, two **Green** neighbors are selected.

# Radius Neighbors: Example




The class of the test data point is then predicted as **Green** fairly.

# KNN: Example

"PlayTennis" prediction with normalized daily weather condition attributes: training samples

| Day | Temperature | Humidity | Wind | PlayTennis |
|---|---|---|---|---|
| D1 | 0.8 | 0.9 | 0.2 | No |
| D2 | 0.7 | 0.8 | 0.8 | No |
| D3 | 0.6 | 0.7 | 0.3 | Yes |
| D4 | 0.5 | 0.9 | 0.2 | Yes |
| D5 | 0.2 | 0.5 | 0.2 | Yes |
| D6 | 0.3 | 0.4 | 0.7 | No |
| D7 | 0.5 | 0.6 | 0.5 | Yes |
| D8 | 0.4 | 0.7 | 0.2 | No |
| D9 | 0.1 | 0.6 | 0.2 | Yes |
| D10 | 0.5 | 0.4 | 0.1 | Yes |
| D11 | 0.6 | 0.4 | 0.8 | Yes |
| D12 | 0.4 | 0.9 | 0.6 | Yes |
| D13 | 0.8 | 0.6 | 0.1 | Yes |
| D14 | 0.5 | 0.8 | 0.9 | No |

# KNN: Example

Using KNN to predict whether to play tennis for a new day $D^* = (\text{Temperature}{=}0.2, \text{Humidity} = 0.8, \text{Wind} = 0.8)$, by setting K to 5 and leveraging Euclidean distance as the distance metric.

First, calculate the distance between the new day $D^*$ and all the training days as

- $d(D^*, D1) = ?$
- $d(D^*, D2) = ?$
- $d(D^*, D3) = ?$
- $d(D^*, D4) = ?$
- $d(D^*, D5) = ?$
- $d(D^*, D6) = ?$
- $d(D^*, D7) = ?$
- $d(D^*, D8) = ?$
- $d(D^*, D9) = ?$
- $d(D^*, D10) = ?$
- $d(D^*, D11) = ?$
- $d(D^*, D12) = ?$
- $d(D^*, D13) = ?$
- $d(D^*, D14) = ?$

# KNN: Example

- The 5 nearest neighbors of $\mathrm{D}^*$ is ?
- The number of neighboring days with class "PlayTennis = Yes" is ?
- The number of neighboring days with class "PlayTennis = No" is ?
- So, the class of $\mathrm{D}^*$ is predicted as "PlayTennis = ?".

# Pros and Cons of KNN

Pros:

- Simple implementation
- No training phase
- Adaptable to new data
- Effective for small datasets

Cons:

- High memory requirements
- Computationally expensive for large datasets
- Sensitive to irrelevant features
- Struggles with imbalanced classes

# The End