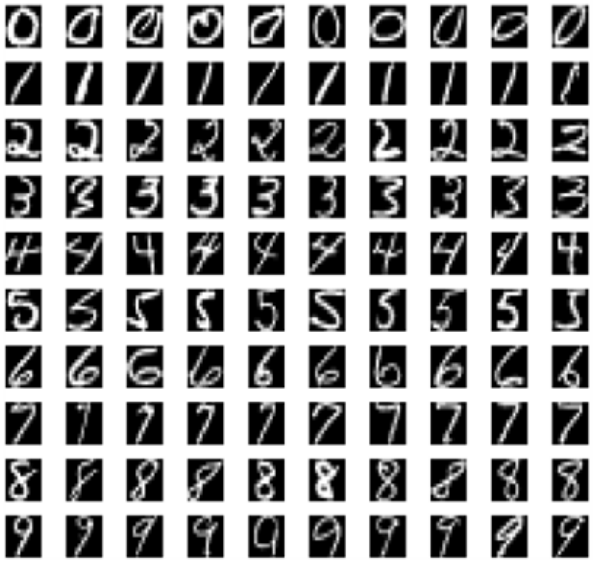
Machine Learning

Contents

2.1. *k*-Nearest Neighbors Algorithm

- Handwritten Digit Example



- Do As Your Neighbor Does

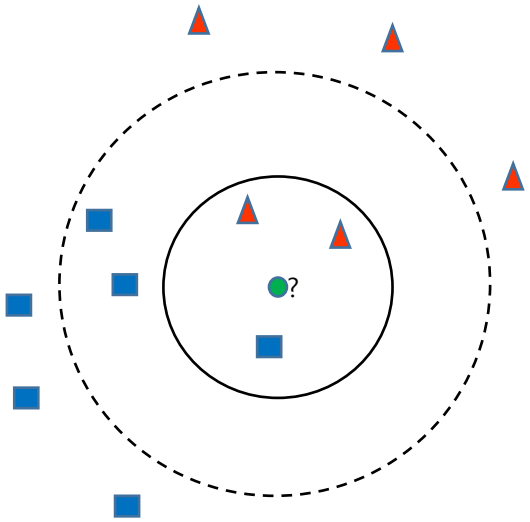
Each input vector $\mathbf{x}$ has a corresponding class label, $c^n \in \{1, \ldots, C\}$. Given a dataset of N train examples, $\mathcal{D} = \{\mathbf{x}^n, c^n\}, n = 1, \ldots, N$, and a novel $\mathbf{x}$, we aim to return the correct class $c(\mathbf{x})$.

For novel $\mathbf{x}$, find the nearest input in the training set and use the class of this nearest input.

- Example of k-NN classifier

  Green Circle(The test sample) should be classified either to the first class(blue squares) or to the second class(red triangles).

  - If *k = 3* , there are 2 triangles vs. only 1 square inside the inner circle.
  - If *k = 5* , there are 3 squares vs. 2 triangles inside the outer circle.

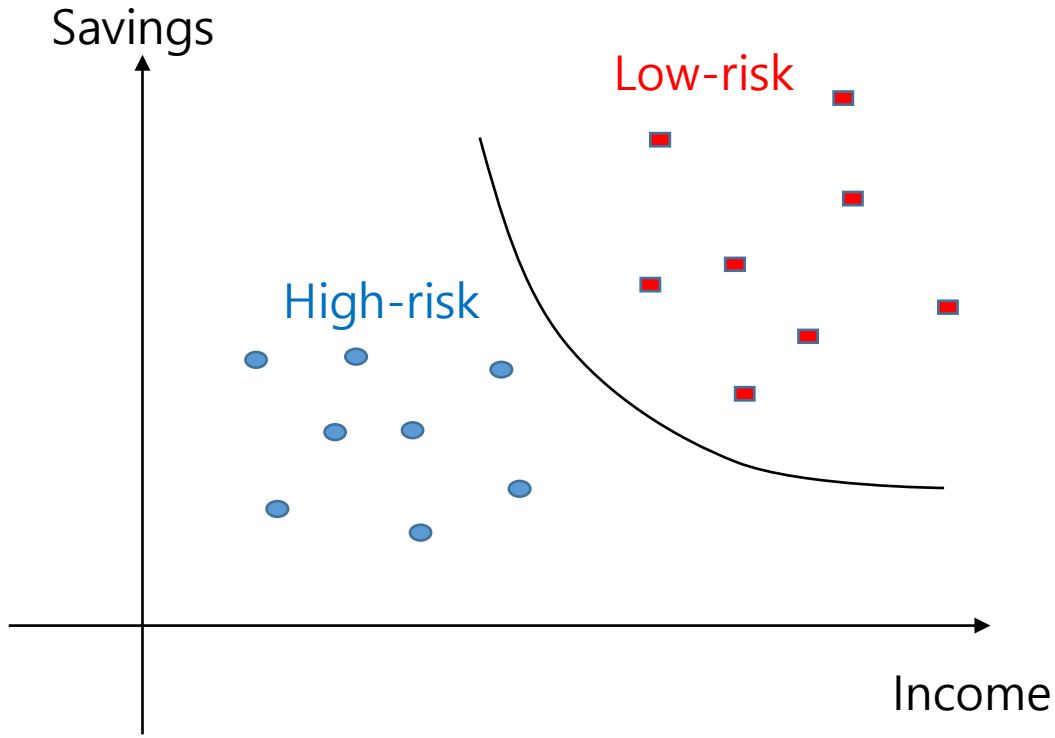


2.2. Classification by *k*-Nearest Neighbors Algorithm

- Classification
  - Ex) Credit scoring
    - Low-risk and high-risk customers from their incomes and savings

  - Class membership
  - Input Features

  - Other examples
    - Handwritten digit recognition
    - Speech Recognition
    - Fraud detection



k-Nearest Neighbors Algorithm

- In pattern recognition, the ***k*-Nearest Neighbors algorithm** (or ***k*-NN** for short) is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

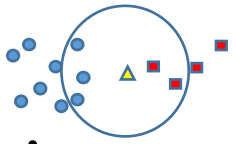
- **In *k-NN classification***, the output is a **class membership**. An object is classified by a **majority vote of its neighbors**, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.

- *k*-NN is a type of **instance-based learning**, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. **The *k*-NN algorithm is among the simplest of all machine learning algorithms.**

k-NN: Algorithm

- **The training examples** are vectors in a multidimensional feature space, each with **a class label**. The training phase of the algorithm consists only of **storing the feature vectors and class labels of the training samples.**

- **In the classification phase**, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.
- A commonly used **distance metric for continuous variables is Euclidean distance**. For discrete variables, such as for text classification, another metric can be used, such as the **overlap metric** (or Hamming distance).



- **A drawback of the basic "majority voting" classification occurs when the class distribution is skewed**. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the *k* nearest neighbors due to their large number. **One way to overcome this problem is to weigh the classification, taking into account the distance from the test point to each of its *k* nearest neighbors.** The class (or value, in regression problems) of each of the *k* nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation.

k-NN: Algorithm

참조:

두 벡터 $\boldsymbol{x}^*$ 과 $\boldsymbol{x}$ 사이의 Euclidean Distance는

$$D(\boldsymbol{x}^*, \boldsymbol{x}) = \left(\sum_i (x^*{}_i - x_i)^2 \right)^{1/2} \tag{2.2.1}$$

와 같이 정의된다. Mahalanobis Distance는 벡터의 공분산 행렬 (Covariance Matrix) $\boldsymbol{S}$를 고려하여

$$D(\boldsymbol{x}^*, \boldsymbol{x}) = \left((\boldsymbol{x}^* - \boldsymbol{x})^T \boldsymbol{S}^{-1} (\boldsymbol{x}^* - \boldsymbol{x}) \right)^{1/2} \tag{2.2.2}$$

로 정의되며, $\boldsymbol{S} = \boldsymbol{I}$ 이면 Mahalanobis Distance는 Euclidean Distance와 같다. 만약, $\boldsymbol{S}$가 대각행렬이면

$$D(\boldsymbol{x}^*, \boldsymbol{x}) = \left(\sum_i \frac{(x^*{}_i - x_i)^2}{s_i^2} \right)^{1/2} \tag{2.2.3}$$

이 되며, 이를 Normalized Euclidean Distance라고 한다. 여기서, s_i는 표준편차이다.

두 벡터 $\boldsymbol{x}^*$ 과 $\boldsymbol{x}$ 의 요소가 이진수일 경우 Hamming Distance는 다른 비트 수의 합으로 정의된다. 예를 들면, $\boldsymbol{x}^* = [11001]$이고 $\boldsymbol{x} = [10101]$이면 Hamming Distance는 2이다.

예제 2.2-1

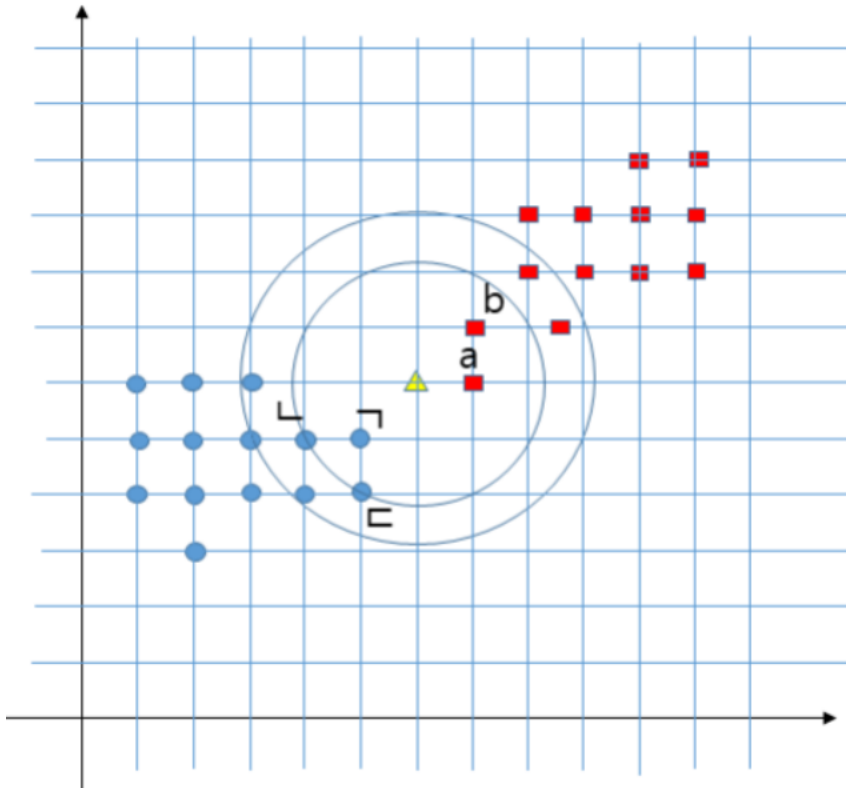
아래 그림과 같이 두 클래스(사각형 혹은 원)에 속하는 데이터가 2차원 공간에 분포하고 있는 경우, 입력이 삼각형으로 주어지면 k-NN에서 k=5로 두고서 이 삼각형은 사각형과 원 중 어느 클래스에 속하는지 구하라? 투표에 거리에 반비례하는 가중치를 적용한 경우와 다수 투표만을 적용한 경우의 결과를 비교하여 보아라.

$d_a = 1,\ d_b = \sqrt{2}$

$1 + 1/\sqrt{2}$ 표

$d_{\ulcorner} = \sqrt{2},\ d_{\llcorner} = \sqrt{5},\ d_{\sqsubset} = \sqrt{5}$

$1/\sqrt{2} + 1/\sqrt{5} + 1/\sqrt{5}$ 표



참조: 역행렬

$\boldsymbol{W}=(w_{ij})$를 역행렬을 구할 수 있는 $n\times n$ 차원이라고 하자. 그러면,

$$\boldsymbol{W}^{-1}=\frac{1}{\det \boldsymbol{W}}adj(\boldsymbol{W}) \tag{2.2.4}$$

이 된다. 여기서, $\det \boldsymbol{W}$는 행렬식을 나타내며, $adj(\boldsymbol{W})$는 수반행렬(Adjoint)이다. 수반행렬은

$$adj(\boldsymbol{W})=\begin{pmatrix} W_{11} & \cdots & W_{n1} \\ \vdots & & \vdots \\ W_{1n} & \cdots & W_{nn} \end{pmatrix} \tag{2.2.5}$$

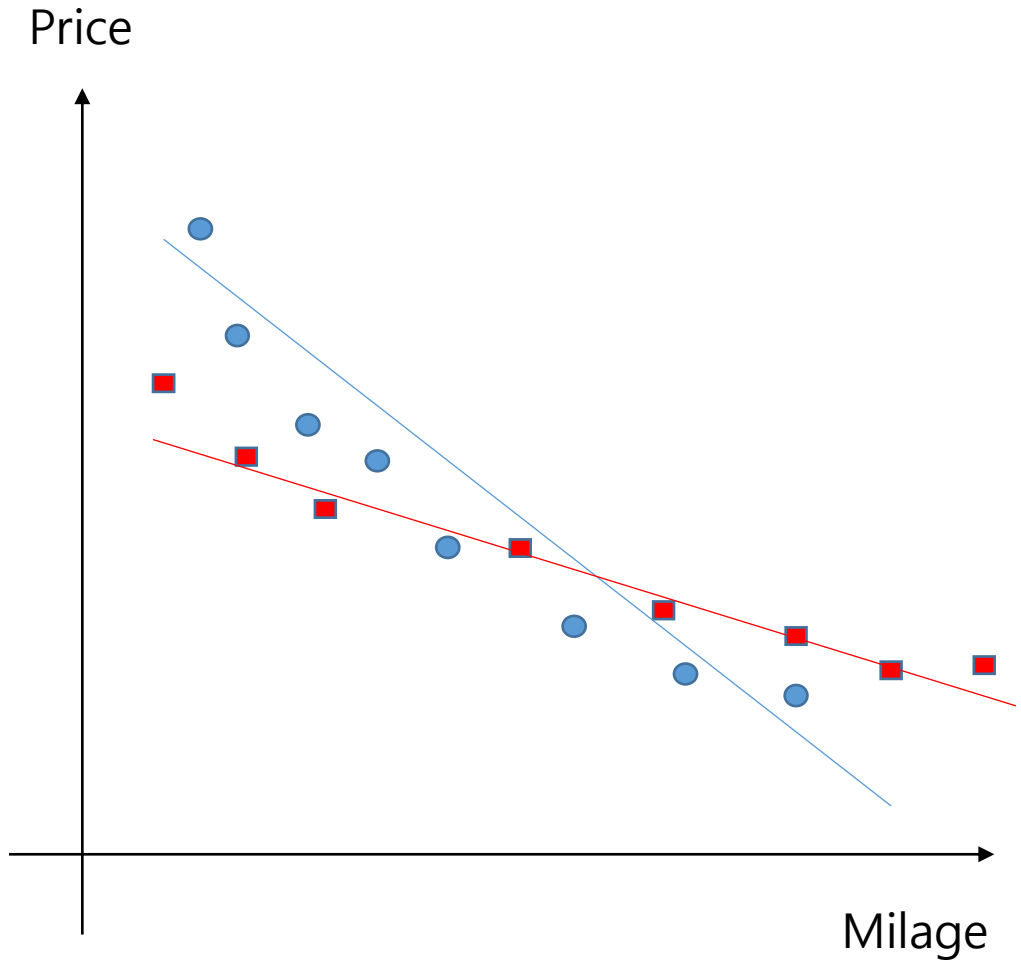
이며, 여기서 W_{ij}는 여인수(Cofactor)이다. 이는 행렬 $\boldsymbol{W}$에서 i번째 열과 j번째 행을 제외한 나머지 $(n-1)\times(n-1)$ 행렬식을 취한 후 $(-1)^{i+j}$를 곱하여 얻어진다. 행렬식은 여인수의 함수로

$$\det \boldsymbol{W}=\sum_{k=1}^{n} w_{ik}W_{ik} \tag{2.2.6}$$

와 같이 표현된다.

2.3. Regression by *k*-Nearest Neighbors Algorithm

- Regression
  - Ex) The price of a used car
  - Output values: continuous(regression) vs. discrete (classification)
  - Other examples
    - Stock price prediction
    - Oil price prediction
    - Water level prediction



- In pattern recognition, the ***k*-Nearest Neighbors algorithm** (or ***k*-NN** for short) is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

- **In *k-NN regression***, the output is the property value for the object. This value is **the average of the values of its *k* nearest neighbors**.

- Both for classification and regression, **it can be useful to assign weight to the contributions of the neighbors**, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of $1/d$, where d is the distance to the neighbor.

예제 2.3-1

k-NN으로 회귀 문제를 해결하고자 한다. k=3로 두고서 새로운 입력에 가장 가까운 세 벡터에 해당하는 출력 값이 y_1, y_2, y_3이고 이들의 거리는 각각 d_1, d_2, d_3이다. 평균에 의해 출력을 구하는 경우와 거리에 반비례하는 가중치를 적용한 경우(가중치 평균)의 출력 값 계산을 하여라. 그리고, 가중치 평균의 적용 시 가중치의 합은 1이 됨을 증명하여라.

2.4. *k*-NN: Parameter selection

- **The best choice of *k* depends upon the data**; generally, larger values of *k* reduce the effect of noise on the classification,[5] but make boundaries between classes less distinct. **A good *k* can be selected by various heuristic techniques** (see hyperparameter optimization). The special case where the class is predicted to be the class of the closest training sample (i.e. when $k = 1$) is called **the nearest neighbor algorithm**.

- **The accuracy of the *k*-NN algorithm can be severely degraded by the presence of noisy or irrelevant features**, or if the feature scales are not consistent with their importance. Much research effort has been put into selecting or scaling features to improve classification. A particularly popular approach is the use of evolutionary algorithms to optimize feature scaling. Another popular approach is to scale features by the mutual information of the training data with the training classes.

- **In binary (two class) classification problems, it is helpful to choose *k* to be an odd number as this avoids tied votes**. One popular way of choosing the empirically optimal *k* in this setting is via bootstrap method.

k-NN: Feature Extraction

- **When the input data to an algorithm is too large to be processed and it is suspected to be redundant** (e.g. the same measurement in both feet and meters) **then the input data will be transformed into a reduced representation set of features** (also named features vector).

- Transforming the input data into the set of features is called **feature extraction.**

- If the features extracted are carefully chosen, it is expected that **the features set will extract the relevant information from the input data in order to perform the desired task** using this reduced representation instead of the full size input.
- Feature extraction is performed on raw data prior to applying *k*-NN algorithm on the transformed data in feature space.

k-NN: Dimension reduction

- For high-dimensional data (e.g., with number of dimensions more than 10) **dimension reduction is usually performed prior to applying the *k*-NN algorithm in order to avoid the effects of the curse of dimensionality.**

- The curse of dimensionality in the *k*-NN context basically means that Euclidean distance is unhelpful in high dimensions because all vectors are almost equidistant to the search query vector (imagine multiple points lying more or less on a circle with the query point at the center; the distance from the query to all data points in the search space is almost the same).

- Feature extraction and dimension reduction can be combined in one step using principal component analysis (PCA), linear discriminant analysis (LDA), or canonical correlation analysis (CCA) techniques as a pre-processing step, followed by clustering by *k*-NN on feature vectors in reduced-dimension space. In machine learning this process is also called low-dimensional embedding.

k-NN Algorithm

k-NN 알고리즘

① 학습패턴 $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ 를 저장한다.

② 새로운 입력 $\boldsymbol{x}^*$이 주어지면 입력과 학습패턴 사이의 거리 d를 계산한다.

③ 거리가 가장 가까운 이웃 k개를 선정한다.

$$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), (\boldsymbol{x}_3, y_3), \ldots, (\boldsymbol{x}_k, y_k)$$

④ 분류의 문제이면 다수투표를 수행하고, 회귀의 문제이면 평균을 수행한다.

⑤ 다수 투표 혹은 평균의 과정에 가중치를 거리의 역수 $1/d$로 적용할 수 있다.