LN3. K-Nearest Neighbors

Kun He

Data Mining and Machine Learning Lab
(John Hopcroft Lab)
Huazhong University of Science & Technology

brooklet60@hust.edu.cn

2022年04月29日

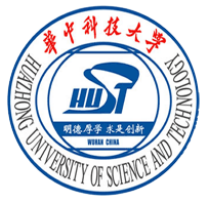


Table of contents

1. The k-NN algorithm
   - Paramatric and Non-paramatric Model:
   - Formal definition of k-NN:
   - Examples of kNN
2. Parameter selection
   - Distance function
   - K value
3. Special k-nearest neighbor classifier
   - 1-nearest neighbor classifier
   - *weighted nearest neighbor classifier
4. Curse of Dimensionality
   - Distances between points
   - Distances to hyperplanes
   - Data with low dimensional structure
5. How to solve the curse of dimensionality
   - Dimension reduction
   - Data reduction

Table of Contents

1. The k-NN algorithm
   - Paramatric and Non-paramatric Model:
   - Formal definition of k-NN:
   - Examples of kNN
2. Parameter selection
   - Distance function
   - K value
3. Special k-nearest neighbor classifier
   - 1-nearest neighbor classifier
   - *weighted nearest neighbor classifier
4. Curse of Dimensionality
   - Distances between points
   - Distances to hyperplanes
   - Data with low dimensional structure
5. How to solve the curse of dimensionality
   - Dimension reduction
   - Data reduction

Formal definition of k-NN:

Paramatric Model:

The model has a fixed mumber of parameters.

Non-paramatric Model:

The mumber of parameters grow with the amount of training data.

- Paramatric Model: faster to use, but has stronger assumptions on the data distribution.
- Non-paramatric Model: more flexible, but higher computational cost. E.g., k-NN

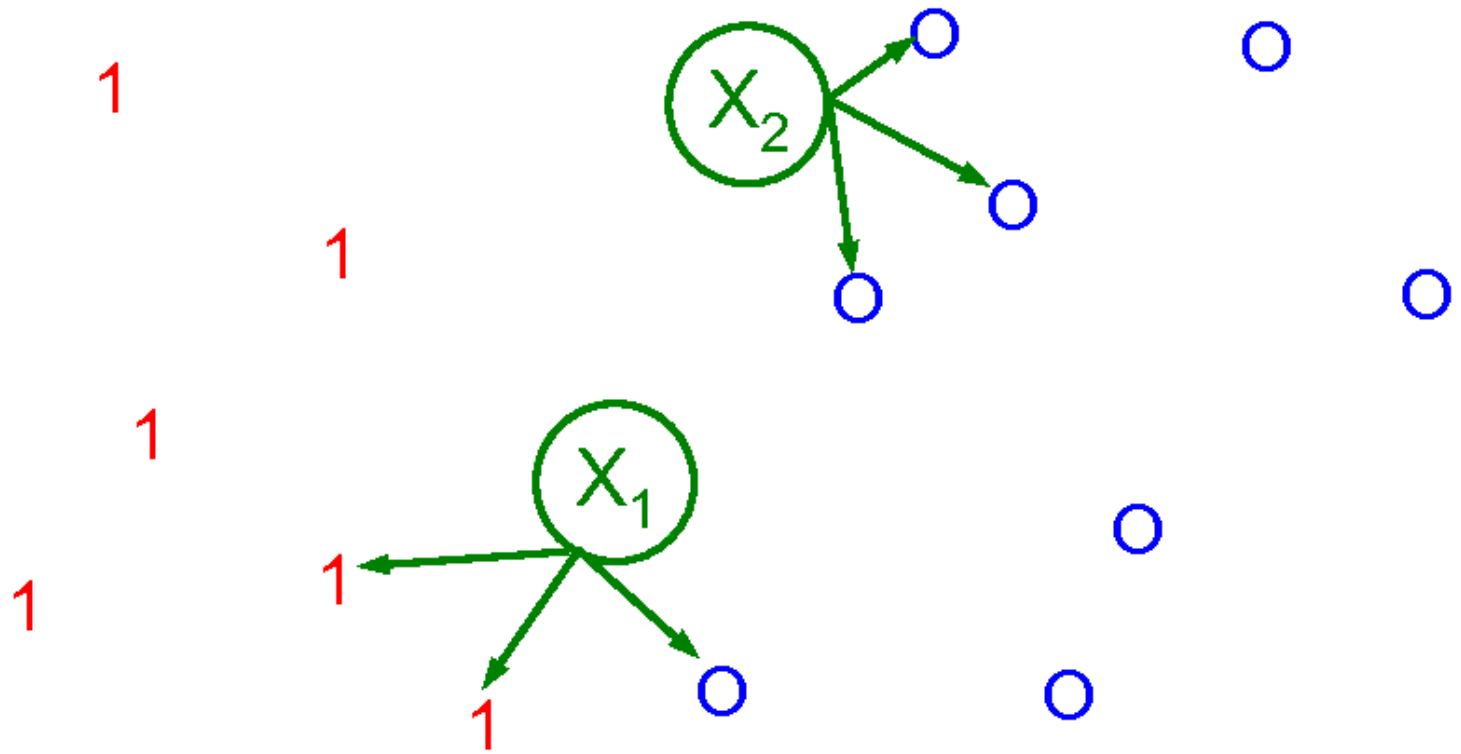
Formal definition of k-NN:

Basic idea:

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression.

- Assumption:Similar inputs have similar outputs
- Classification rule: For a test input **x**, assign the most common label amongest its k most similar training inputs
- Regression rule: The output is the property value for the object. This value is the average of the values of **k** nearest neighbors.

Examples of kNN, k = 3



Formal definition of k-NN:

Formal definition

- Test point: $\mathbf{x}$
- Denote the set of the k nearest neighbors of $\mathbf{x}$ as $S_{\mathbf{x}}$. Formally $S_{\mathbf{x}}$ is defined as $S_{\mathbf{x}} \subseteq D$ s.t. $|S_{\mathbf{x}}| = k$ and $\forall (\mathbf{x}', y') \in D \backslash S_{\mathbf{x}}$,

$$\text{dist}(\mathbf{x}, \mathbf{x}') \geq \max_{(\mathbf{x}'', y'') \in S_{\mathbf{x}}} \text{dist}(\mathbf{x}, \mathbf{x}''),$$

(i.e. every point in D but *not* in $S_{\mathbf{x}}$ is at least as far away from $\mathbf{x}$ as the furthest point in $S_{\mathbf{x}}$). We can then define the classifier $h()$ as a function returning the most common label in $S_{\mathbf{x}}$:

$$h(\mathbf{x}) = \text{mode}(\{y'' : (\mathbf{x}'', y'') \in S_{\mathbf{x}}\}),$$

where mode$(\cdot)$ means to select the label of the highest occurrence.

A binary classification example

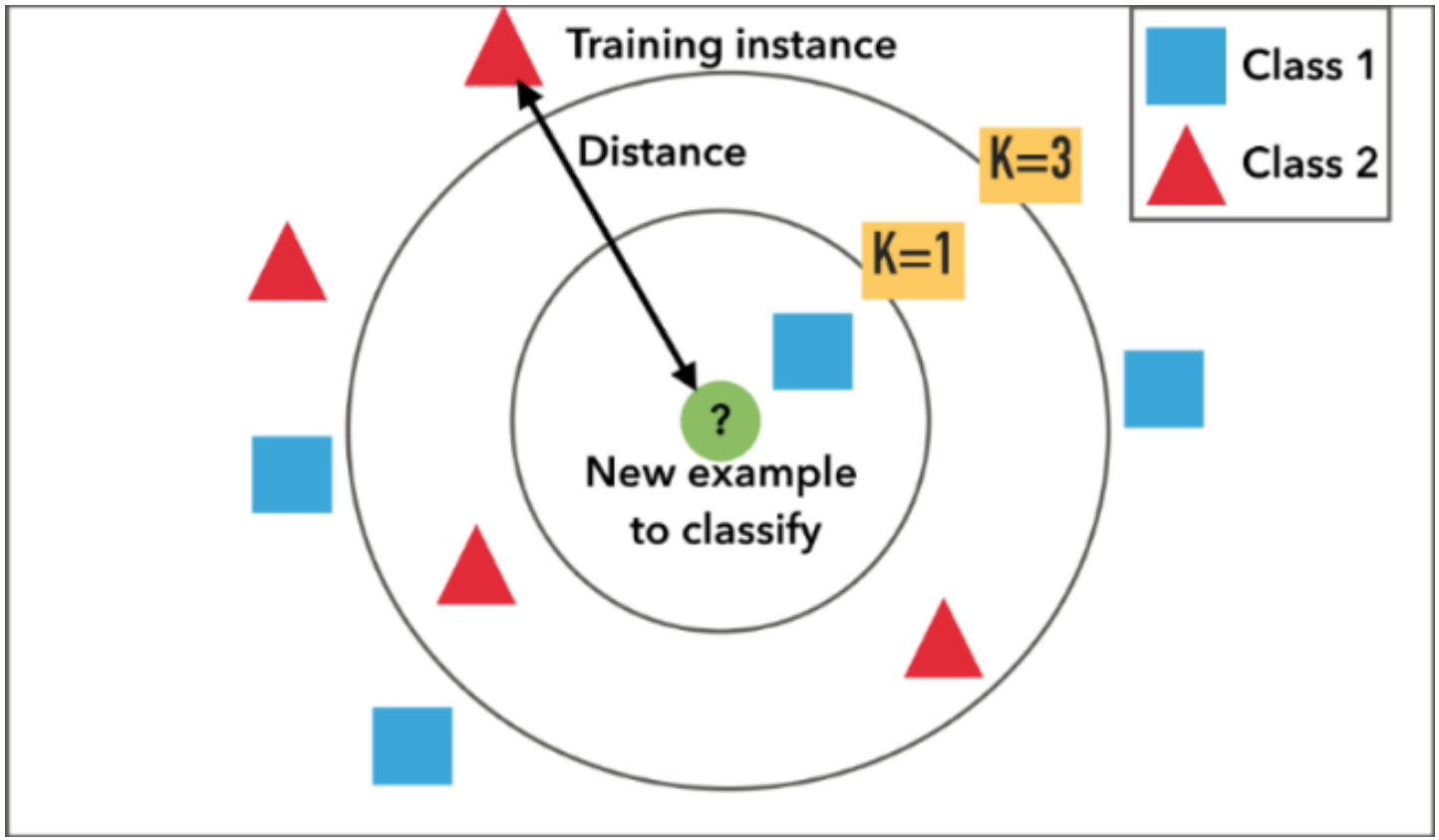


图: Example of k-NN classification. The test sample (inside circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (outside circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If, for example k = 5 it is assigned to the first class (3 squares vs. 2 triangles outside the outer circle).

Table of Contents

1. The k-NN algorithm
   - Paramatric and Non-paramatric Model:
   - Formal definition of k-NN:
   - Examples of kNN
2. Parameter selection
   - Distance function
   - K value
3. Special k-nearest neighbor classifier
   - 1-nearest neighbor classifier
   - *weighted nearest neighbor classifier
4. Curse of Dimensionality
   - Distances between points
   - Distances to hyperplanes
   - Data with low dimensional structure
5. How to solve the curse of dimensionality
   - Dimension reduction
   - Data reduction

Distance function

The k-nearest neighbor classifier fundamentally relies on a distance metric. The better that metric reflects label similarity, the better the classified will be. The most common choice is the **Minkowski distance**

$$\text{dist}(\mathbf{x}, \mathbf{z}) = \left(\sum_{r=1}^{d} |x_r - z_r|^p \right)^{1/p} .$$

Quiz

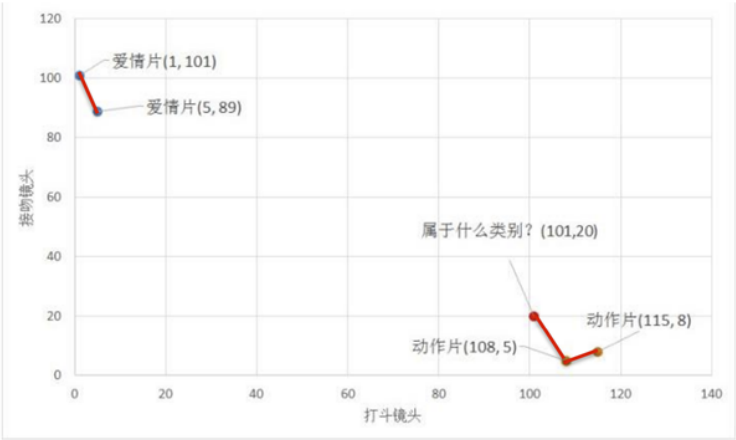
This distance definition is pretty general and contains many well-known distances as special cases. Can you identify the following candidates?

- $p = 1$
- $p = 2$
- $p \to \infty$

p = 2

欧氏距离(Euclidean distance)

电影分类

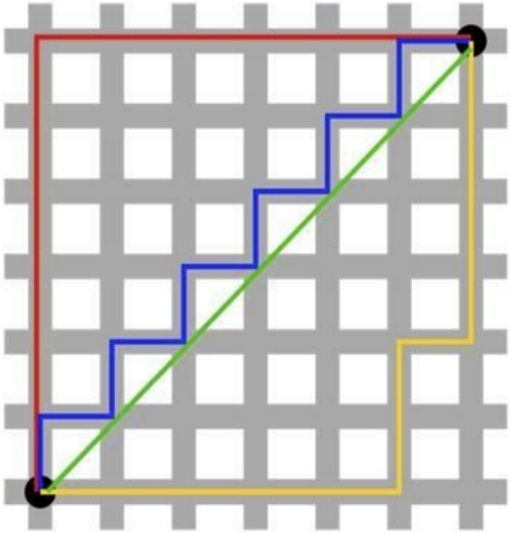


$$d(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$$

欧几里得度量（Euclidean Metric）（也称欧氏距离）是一个通常采用的距离定义，指在m维空间中两个点之间的真实距离，或者向量的自然长度（即该点到原点的距离）。在二维和三维空间中的欧氏距离就是两点之间的实际距离。

p = 1

曼哈顿距离(Manhattan distance)

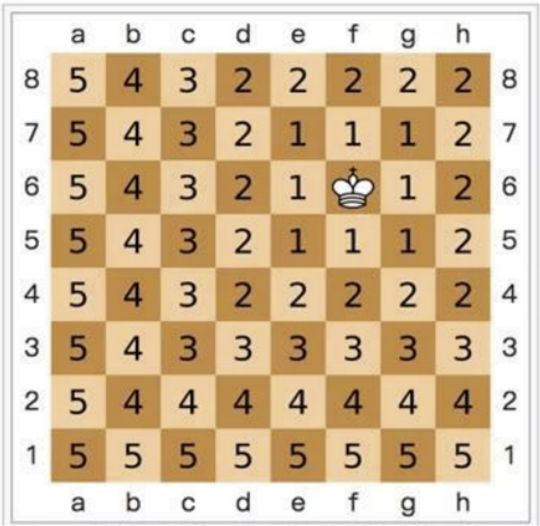


$$d(x,y) = \sum_i |x_i - y_i|$$

想象你在城市道路里，要从一个十字路口开车到另外一个十字路口，驾驶距离是两点间的直线距离吗？显然不是，除非你能穿越大楼。实际驾驶距离就是这个“曼哈顿距离”。而这也是曼哈顿距离名称的来源， 曼哈顿距离也称为城市街区距离(City Block distance)。

p = infinity

切比雪夫距离(Chebyshev distance)



	a	b	c	d	e	f	g	h	
8	5	4	3	2	2	2	2	2	8
7	5	4	3	2	1	1	1	2	7
6	5	4	3	2	1	♔	1	2	6
5	5	4	3	2	1	1	1	2	5
4	5	4	3	2	2	2	2	2	4
3	5	4	3	3	3	3	3	3	3
2	5	4	4	4	4	4	4	4	2
1	5	5	5	5	5	5	5	5	1
	a	b	c	d	e	f	g	h	

$$d(x,y) = \max_i |x_i - y_i|$$

二个点之间的距离定义是其各坐标数值差绝对值的最大值。

国际象棋棋盘上二个位置间的切比雪夫距离是指王要从一个位子移至另一个位子需要走的步数。由于王可以往斜前或斜后方向移动一格，因此可以较有效率的到达目的的格子。上图是棋盘上所有位置距f6位置的切比雪夫距离。

p = 1,2,...,infinity

闵可夫斯基距离(Minkowski distance)

p取1或2时的闵氏距离是最为常用的

$p = 2$即为欧氏距离，

$p = 1$时则为曼哈顿距离。

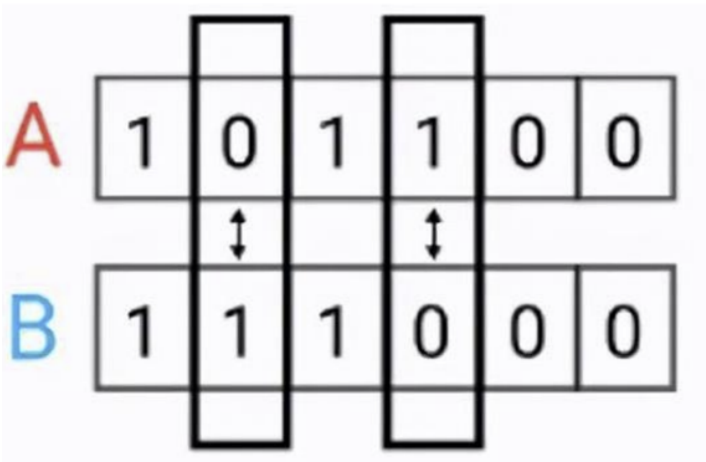
当p取无穷时的极限情况下，可以得到切比雪夫距离

$$d(x,y) = \left(\sum_i |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Other distance

汉明距离(Hamming distance)

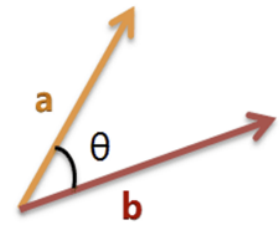
$$d(x,y) = \frac{1}{N}\sum_{i} 1_{x_i \neq y_i}$$

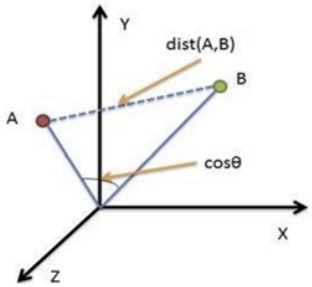


汉明距离是使用在数据传输差错控制编码里面的，汉明距离是一个概念，它表示两个（相同长度）字对应位不同的数量，我们以表示两个字之间的汉明距离。对两个字符串进行异或运算，并统计结果为1的个数，那么这个数就是汉明距离。

Other distance

余弦相似度





两个向量有相同的指向时，余弦相似度的值为1；两个向量夹角为90°时，余弦相似度的值为0；两个向量指向完全相反的方向时，余弦相似度的值为-1。

假定A和B是两个n维向量，A是$[A_1, A_2, \ldots, A_n]$，B是$[B_1, B_2, \ldots, B_n]$，则A和B的夹角的余弦等于：

$$\cos(\theta) = \frac{A \cdot B}{\| A \| \| B \|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

The best K

How to choose a suitable K

A smaller K

- Reduce the approximation error of the learning [Training data]
- Enlarge the estimation error of the learning [Test data]
- A more complex model, easy to overfitting

A larger K:

- Reduce the estimation error of the learning
- Enlarge the approximation error of the learning
- A simpler model

The best K

How to choose a suitable K

The best choice of **k** depends upon the data.

Generally, larger values of **k** reduces effect of the noise on the classification, but make boundaries between classes less distinct.

A good **k** can be selected by various heuristic techniques (see hyperparameter optimization).

In binary (two class) classification problems, it is helpful to choose **k** to be an odd number to avoid tie.

Some popular ways of choosing the empirically optimal **k** includes :

- bootstrap method
- cross validation
- Bayes method

Table of Contents

1. The k-NN algorithm
   - Paramatric and Non-paramatric Model:
   - Formal definition of k-NN:
   - Examples of kNN
2. Parameter selection
   - Distance function
   - K value
3. Special k-nearest neighbor classifier
   - 1-nearest neighbor classifier
   - *weighted nearest neighbor classifier
4. Curse of Dimensionality
   - Distances between points
   - Distances to hyperplanes
   - Data with low dimensional structure
5. How to solve the curse of dimensionality
   - Dimension reduction
   - Data reduction

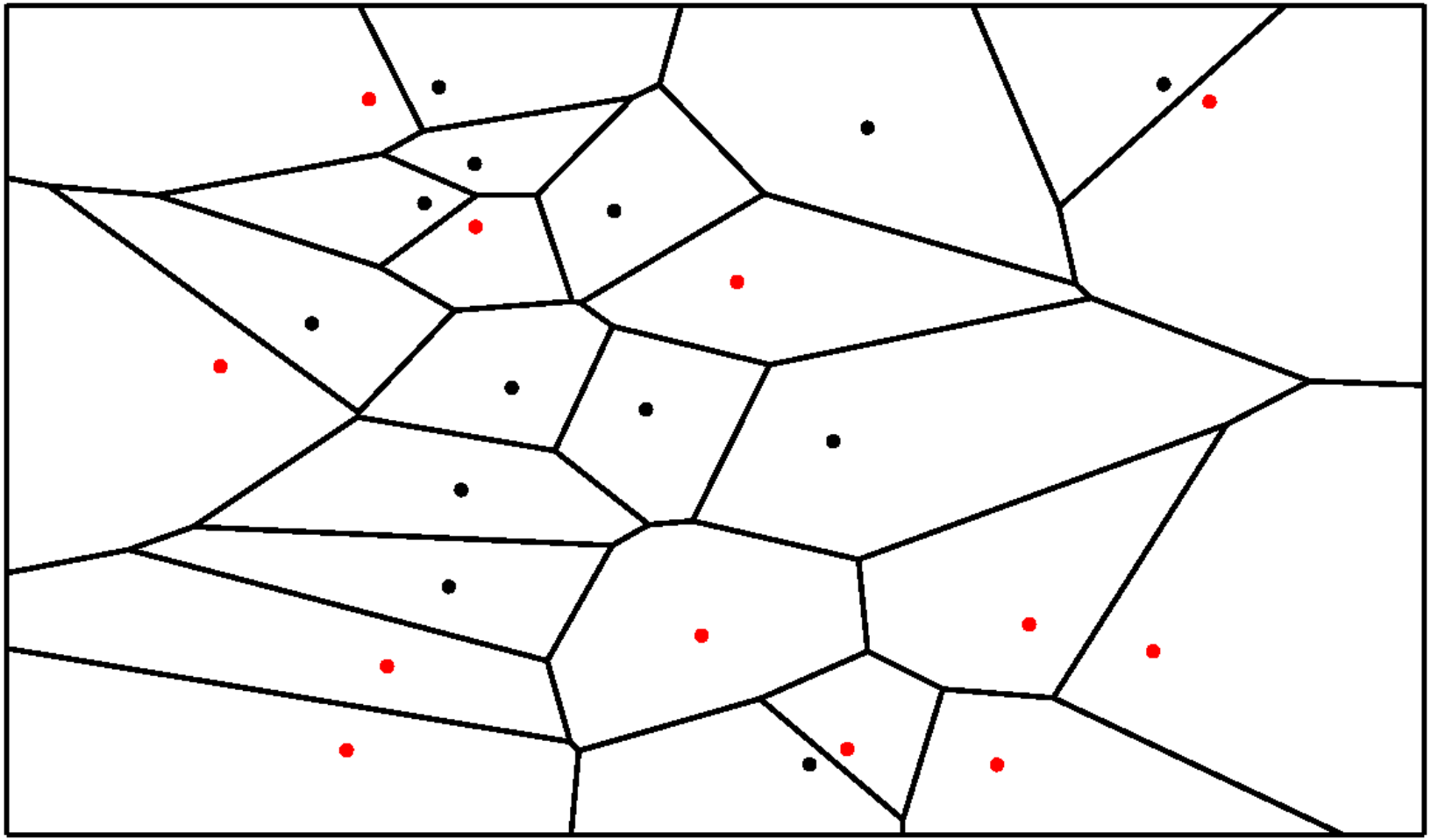
1-NN classifier

1-NN classifier

The most intuitive nearest neighbour type classifier is the one nearest neighbour classifier**(k=1)** that assigns a point **x** to the class of its closest neighbour in the feature space.

As the size of training data set approaches infinity, the one nearest neighbour classifier guarantees an error rate of no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data).

Examples of kNN, k = 1



Bayes optimal classifier

Example: Assume (and this is almost never the case) you knew $\mathrm{P}(y|\mathbf{x})$, then you would simply predict the most likely label.

$$\text{The Bayes optimal classifier predicts: } y^* = h_{\text{opt}}(\mathbf{x}) = \underset{y}{\operatorname{argmax}}\, P(y|\mathbf{x})$$

Although the Bayes optimal classifier is as good as it gets, it still can make mistakes. It is always wrong if a sample does not have the most likely label. We can compute the probability of that happening precisely (which is exactly the error rate):

$$\epsilon_{BayesOpt} = 1 - \mathrm{P}(h_{\text{opt}}(\mathbf{x})|\mathbf{x}) = 1 - \mathrm{P}(y^*|\mathbf{x})$$

Assume for example an email $\mathbf{x}$ can either be classified as spam $(+1)$ or ham (-1). For the same email $\mathbf{x}$ the conditional class probabilities are:

$$\mathrm{P}(+1|\mathbf{x}) = 0.8$$
$$\mathrm{P}(-1|\mathbf{x}) = 0.2$$

In this case the Bayes optimal classifier would predict the label $y^* = +1$ as it is most likely, and its error rate would be $\epsilon_{BayesOpt} = 0.2$.

1-NN Convergence

[Cover, Hart, 1967]

As $n \to \infty$, **the 1-NN error is no more than twice the error of the Bayes Optimal classifier.** (Similar guarantees hold for $k > 1$.)

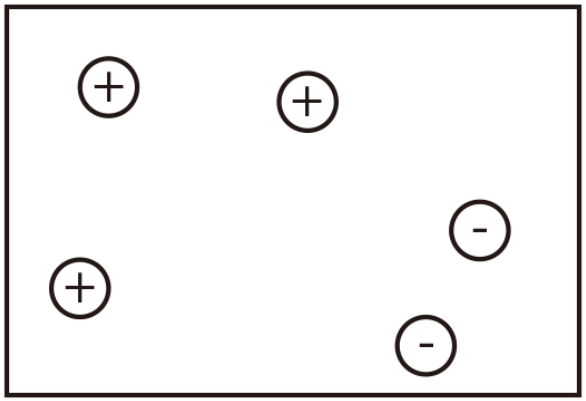


图: n small

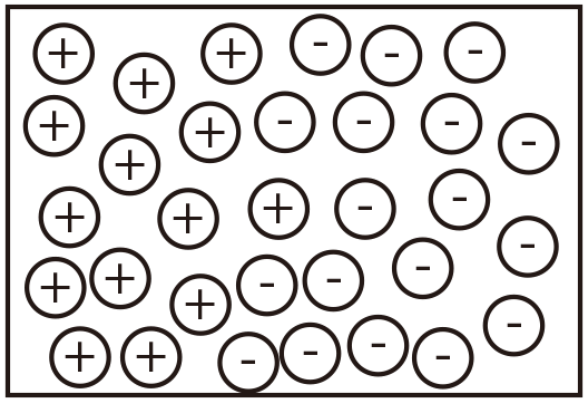


图: n large

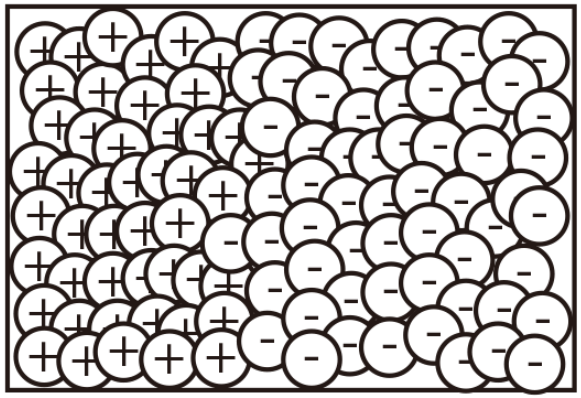


图: $n \to \infty$

1-NN Convergence Proof

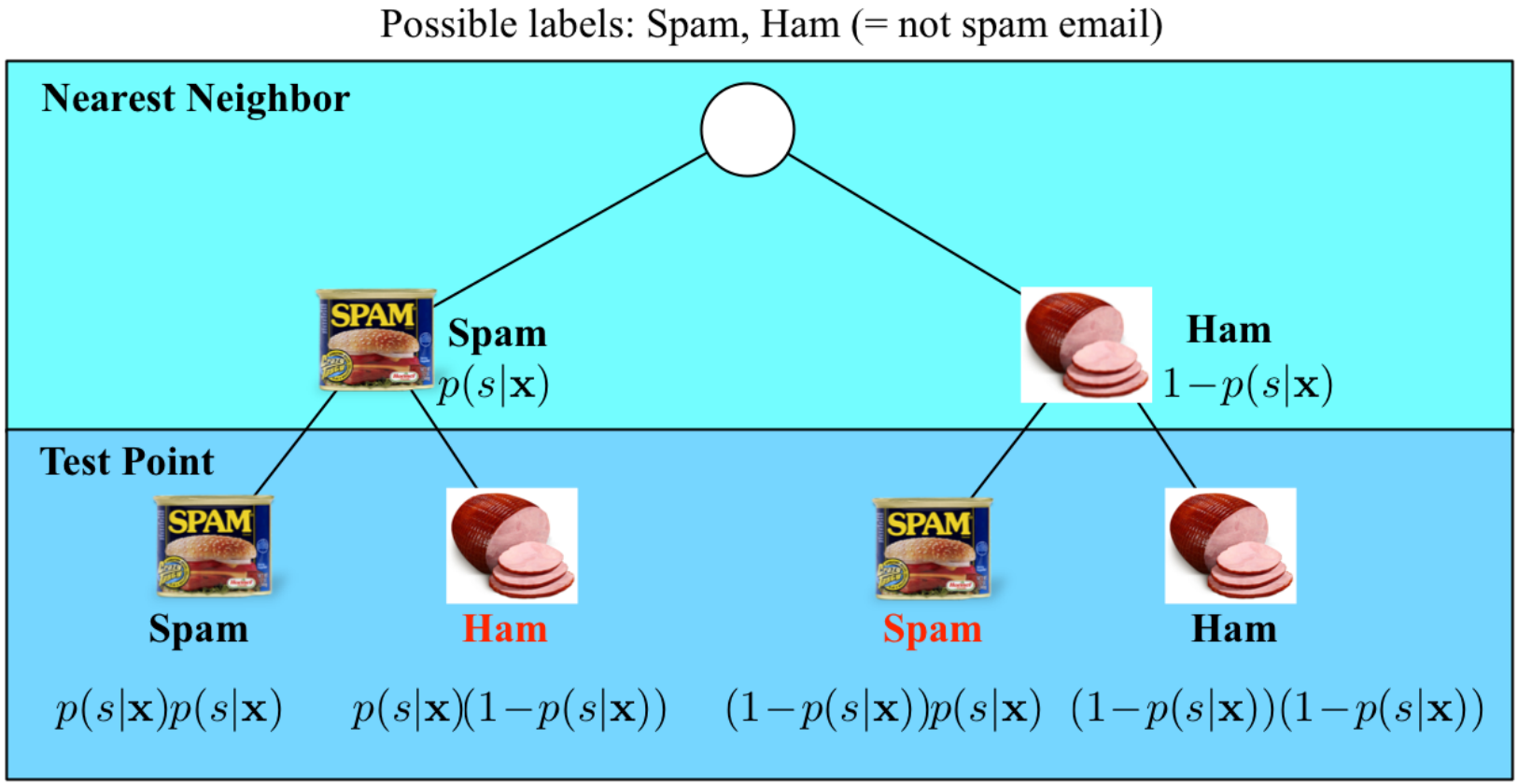


图: In the limit case, the test point and its nearest neighbor are identical. There are exactly two cases when a misclassification can occur: when the test point and its nearest neighbor have different labels. The probability of this happening is the probability of the two red events: $(1-p(s|\mathbf{x}))p(s|\mathbf{x}) + p(s|\mathbf{x})(1-p(s|\mathbf{x})) = 2p(s|\mathbf{x})(1-p(s|\mathbf{x}))$.

1-NN Convergence Proof

Let $\mathbf{x}_{\mathrm{NN}}$ be the nearest neighbor of our test point $\mathbf{x}_{\mathrm{t}}$. As $n \to \infty$, $\mathrm{dist}(\mathbf{x}_{\mathrm{NN}}, \mathbf{x}_{\mathrm{t}}) \to 0$, i.e. $\mathbf{x}_{\mathrm{NN}} \to \mathbf{x}_t$. (This means the nearest neighbor is identical to $\mathbf{x}_{\mathrm{t}}$.)
You return the label of $\mathbf{x}_{\mathrm{NN}}$. What is the probability that this is not the label of $\mathbf{x}_{\mathrm{t}}$?
(This is the probability of drawing two different label of $\mathbf{x}$)

$$\begin{aligned}\epsilon_{NN} &= \mathrm{P}(y^*|\mathbf{x}_t)(1-\mathrm{P}(y^*|\mathbf{x}_{NN})) + \mathrm{P}(y^*|\mathbf{x}_{NN})(1-\mathrm{P}(y^*|\mathbf{x}_t)) \\ &\leq (1-\mathrm{P}(y^*|\mathbf{x}_{NN})) + (1-\mathrm{P}(y^*|\mathbf{x}_t)) = 2(1-\mathrm{P}(y^*|\mathbf{x}_t) = 2\epsilon_{\mathrm{BayesOpt}},\end{aligned}$$

where the inequality follows from $\mathrm{P}(y^*|\mathbf{x}_t) \leq 1$ and $\mathrm{P}(y^*|\mathbf{x}_{NN}) \leq 1$. We also used that $\mathrm{P}(y^*|\mathbf{x}_t) = \mathrm{P}(y^*|\mathbf{x}_{NN})$.

*weighted nearest neighbor classifier

The k-nearest neighbour classifier can be viewed as assigning the **k** nearest neighbours a weight **1/k** and all others 0 weight.

This can be generalised to weighted nearest neighbour classifiers.

That is, where the i-th nearest neighbour is assigned a weight w_{ni},with $\sum_{i=1}^{n} w_{ni} = 1$.

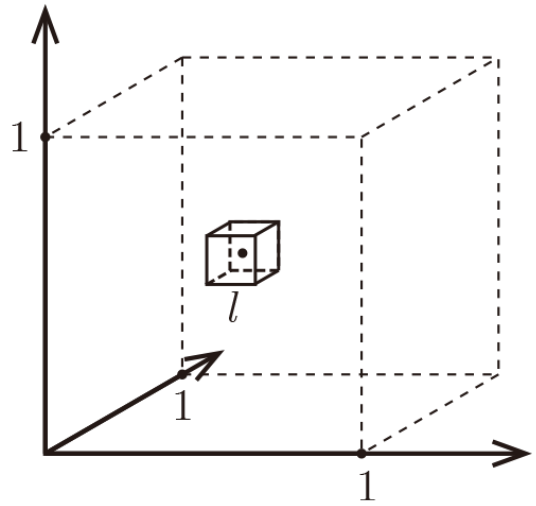
An analogous result on the strong consistency of weighted nearest neighbour classifiers also holds.

Table of Contents

1. The k-NN algorithm
   - Paramatric and Non-paramatric Model:
   - Formal definition of k-NN:
   - Examples of kNN
2. Parameter selection
   - Distance function
   - K value
3. Special k-nearest neighbor classifier
   - 1-nearest neighbor classifier
   - *weighted nearest neighbor classifier
4. Curse of Dimensionality
   - Distances between points
   - Distances to hyperplanes
   - Data with low dimensional structure
5. How to solve the curse of dimensionality
   - Dimension reduction
   - Data reduction

Distances between points

Formally, imagine the unit cube $[0,1]^d$. All training data is sampled *uniformly* within this cube, i.e. $\forall i, x_i \in [0,1]^d$, and we are considering the $k = 10$ nearest neighbors of such a test point.



Let ℓ be the edge length of the smallest hyper-cube that contains all k-nearest neighbor of a test point. Then $\ell^d \approx \frac{k}{n}$ and $\ell \approx \left(\frac{k}{n}\right)^{1/d}$.

Distances between points

For $k = 10$, $n = 1000$:

d	ℓ
2	0.1
10	0.63
100	0.955
1000	0.9954

So as $d \gg 0$ almost the entire space is needed to find the 10-NN.

This breaks down the k-NN assumptions, because the k-NN are not particularly closer (and therefore more similar) than any other data points in the training set.

Why would the test point share the label with those k-nearest neighbors, if they are not actually similar to it?

Distances between points

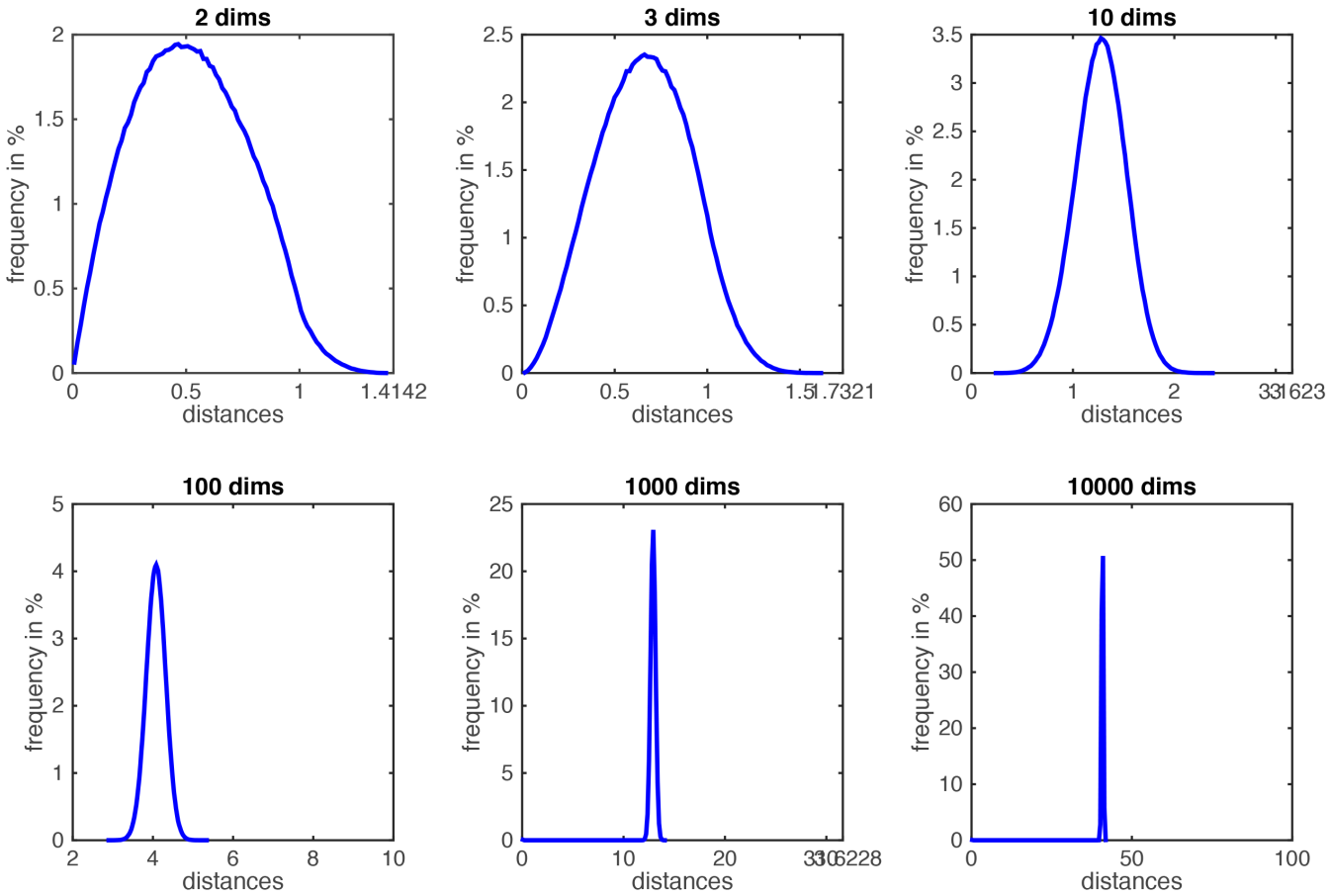


图: Figure demonstrating "the curse of dimensionality". The histogram plots show the distributions of all pairwise distances between randomly distributed points within d-dimensional unit squares. As the number of dimensions d grows, all distances concentrate within a very small

Distances to hyperplanes

Thinking

The distance between two randomly drawn data points increases drastically with their dimensionality. How about the distance to a hyperplane?

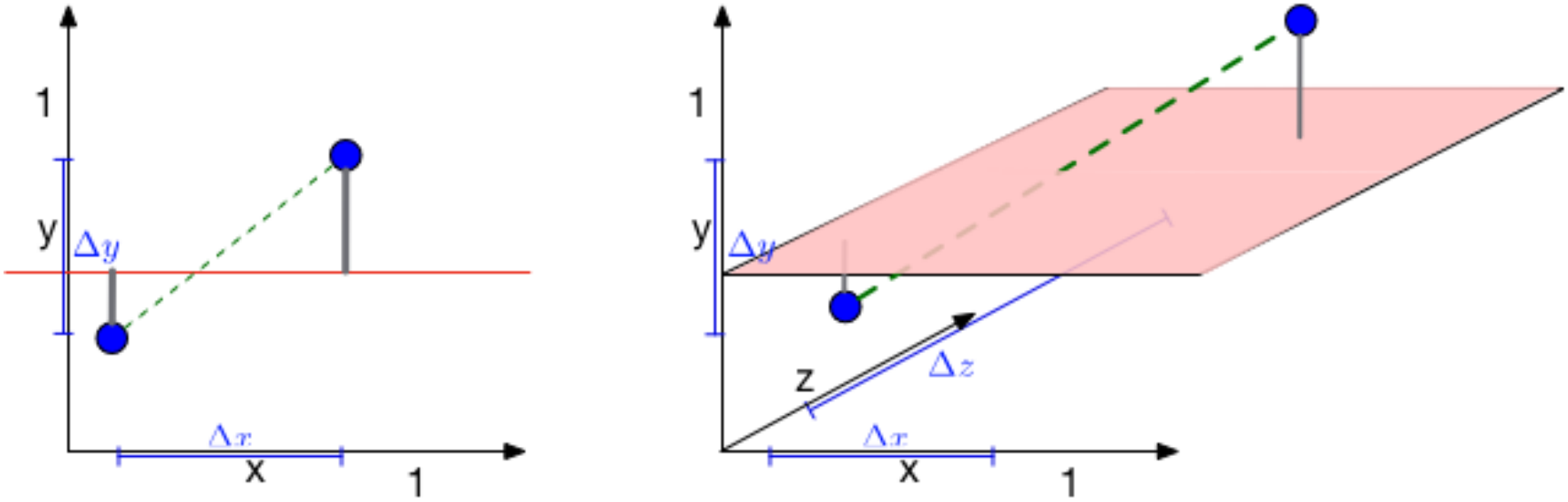
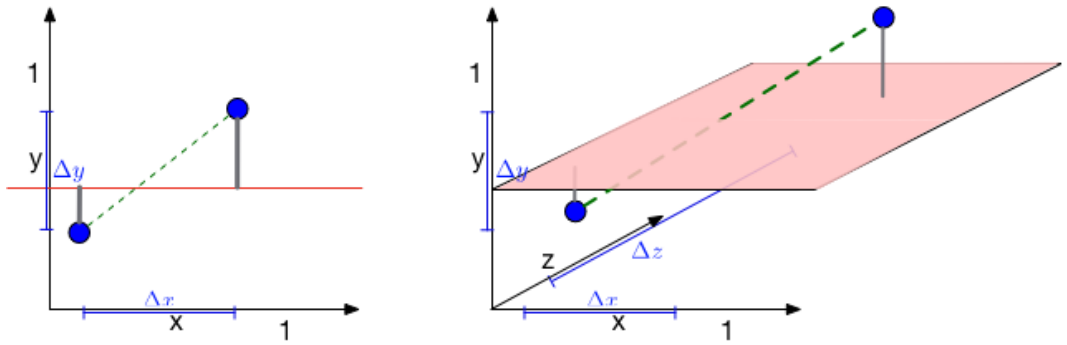


图: The curse of dimensionality has different effects on distances between two points and distances between points and hyperplanes.

Distances to hyperplanes



In d dimensions, $d-1$ dimensions will be orthogonal to the normal of any given hyper-plane. Movement in those dimensions cannot increase or decrease the distance to the hyperplane — the points just shift around and remain at the same distance. As distances between pairwise points become very large in high dimensional spaces, distances to hyperplanes become comparatively tiny.

For machine learning algorithms, this is highly relevant. As we will see later on, many classifiers (e.g. the **Perceptron** or **SVMs**) place hyper planes between concentrations of different classes.

One consequence of the curse of dimensionality is that most data points tend to be very close to these hyperplanes and it is often possible to perturb input slightly (and often imperceptibly) in order to change a classification outcome. This practice has recently

Data with low dimensional structure

However, not all is lost. Data may lie in low dimensional subspace or on sub-manifolds.

Example: natural images (digits, faces).

Although an image of a face may require 18M pixels, a person may be able to describe this person with less than 50 attributes (e.g. male/female, blond/dark hair, ...) along which faces vary.

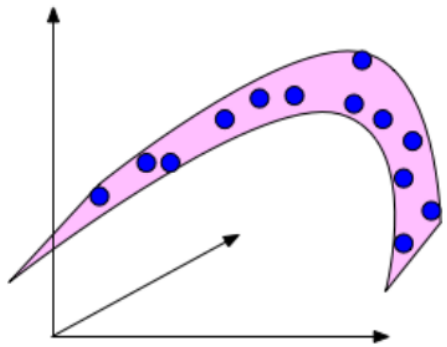


图: An example of a data set in 3d that is drawn from an underlying 2-dimensional manifold. The blue points are confined to the pink surface area, which is embedded in a 3-dimensional ambient space.

Table of Contents

1. The k-NN algorithm
   - Paramatric and Non-paramatric Model:
   - Formal definition of k-NN:
   - Examples of kNN
2. Parameter selection
   - Distance function
   - K value
3. Special k-nearest neighbor classifier
   - 1-nearest neighbor classifier
   - *weighted nearest neighbor classifier
4. Curse of Dimensionality
   - Distances between points
   - Distances to hyperplanes
   - Data with low dimensional structure
5. How to solve the curse of dimensionality
   - Dimension reduction
   - Data reduction

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.

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**.

Data reduction

Data reduction is one of the most important problems for work with huge data sets. Usually, only some of the data points are needed for accurate classification. Those data are called the **prototypes** and can be found as follows:

1. Select the **class-outliers**, that is, training data that are classified incorrectly by k-NN (for a given k)
2. Separate the rest of the data into two sets:
   (i) the prototypes that are used for the classification decisions and
   (ii) the absorbed points that can be correctly classified by k-NN using prototypes. The absorbed points can then be removed from the training set.

Selection of class-outliers

A training example surrounded by examples of other classes is called a **class outlier**. Causes of class outliers include:

- random error
- insufficient training examples of this class (an isolated example appears instead of a cluster)
- missing important features (the classes are separated in other dimensions which we do not know)
- too many training examples of other classes (unbalanced classes) that create a "hostile" background for the given small class

Class outliers with k-NN produce noise. They can be detected and separated for future analysis. Given two natural numbers, $k>r>0$, a training example is called a **(k,r)**NN class-outlier if its **k** nearest neighbors include more than **r** examples of other classes.

Table of Contents

1. The k-NN algorithm
   - Paramatric and Non-paramatric Model:
   - Formal definition of k-NN:
   - Examples of kNN
2. Parameter selection
   - Distance function
   - K value
3. Special k-nearest neighbor classifier
   - 1-nearest neighbor classifier
   - *weighted nearest neighbor classifier
4. Curse of Dimensionality
   - Distances between points
   - Distances to hyperplanes
   - Data with low dimensional structure
5. How to solve the curse of dimensionality
   - Dimension reduction
   - Data reduction

Quick summary of k-NN

- k-NN is a simple and effective classifier if distances reliably reflect a semantically meaningful notion of the dissimilarity. (It becomes truly competitive through metric learning)
- As $n \to \infty$, k-NN becomes provably very accurate, but also very slow.
- As $d \gg 0$, points drawn from a probability distribution stop being similar to each other, and the k-NN assumption breaks down.
- k-NN stores the entire training dataset which it uses as its representation.
- k-NN does not learn any model.
- k-NN makes predictions just-in-time by calculating the similarity between an input sample and each training instance.

Pros and cons of k-NN

Some pros and cons of k-NN:

Pros

- No assumptions about data — useful, for example, for nonlinear data
- Simple algorithm — to explain and understand/interpret
- High accuracy (relatively) — it is pretty high but not competitive in comparison to better supervised learning models
- Versatile — useful for classification or regression

Cons

- Computationally expensive — because the algorithm stores all of the training data
- High memory requirement
- Stores all (or almost all) of the training data
- Prediction stage might be slow (in O(N))
- Sensitive to irrelevant features and the scale of the data

Table of Contents

1. The k-NN algorithm
   - Paramatric and Non-paramatric Model:
   - Formal definition of k-NN:
   - Examples of kNN
2. Parameter selection
   - Distance function
   - K value
3. Special k-nearest neighbor classifier
   - 1-nearest neighbor classifier
   - *weighted nearest neighbor classifier
4. Curse of Dimensionality
   - Distances between points
   - Distances to hyperplanes
   - Data with low dimensional structure
5. How to solve the curse of dimensionality
   - Dimension reduction
   - Data reduction

Reference

Cover, Thomas, and, Hart, Peter. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 1967, 13(1): 21-27.

The End