Distance metrics and data transformations

Jianxin Wu

LAMDA Group
National Key Lab for Novel Software Technology
Nanjing University, China
wujx2001@gmail.com

February 20, 2019

Contents

In this chapter, we deal with two seemingly disjoint topics: distance metrics and similarity measures, and data transformation and normalization. We will see, however, these topics are in fact related to each other.

In fact, in order for the distance metrics to make sense, good data transformation or normalization is required. And, in the design of many data transformation equations and normalization methods, the objective is usually to ensure that the computed distance metric or similarity measure will reflect the inherent distance or similarity of the data.

1 Distance metrics and similarity measures

The tasks we have dealt with in this book so far is to learn a mapping $f : \mathcal{X} \mapsto \mathcal{Y}$ with a set of training examples $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$. In problems we have studied up to now, $y_i \in \mathcal{Y}$ is categorical and $\boldsymbol{x}_i \in \mathcal{X}$ are always real vectors, i.e., $\mathcal{X} \subseteq \mathbb{R}^d$.

It is vital to measure whether two examples $\boldsymbol{x}$ and $\boldsymbol{y}$ are similar or not, and the preferred answer to this question is a real number. In k-nearest neighbor, principal component analysis and Fisher's linear discriminant, we use *distance* as a measure for the dissimilarity; while in support vector machines and kernel density estimation, we use dot-product or other kernels (such as the Gaussian kernel) as a measure for the similarity between any two vector $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. All these similarity and dissimilarity measures return a real number, whose magnitude is associated with the level of similarity or dissimilarity.

These two concepts, similarity and dissimilarity, are closely related: if similarity is high then dissimilarity is low, and vice versa. The RBF (Gaussian) kernel is a typical example of this relationship: the similarity is measured by a decreasing function of the dissimilarity (which is the Euclidean distance):

$$K_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{y}\|^2)\,, \tag{1}$$

in which the decreasing function is

$$g(d) = \exp(-\gamma d^2)$$

with $\gamma > 0$. Other types of decreasing functions, such as

$$g(d) = \frac{1}{d}\,,$$

can also be used. The exponential function, however, has an additional benefit that its range (i.e., computed similarity) is between 0 and 1, which fits our intuition about similarity. For example, when $K_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{y}) = 0.99$, it is reasonable to say that the similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$ is 99%.

1.1 Distance metrics

We first pay attention to the distances, or how to measure the dissimilarity. Formally speaking, a metric is a function $\mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_+$, in which $\mathbb{R}_+$ is the set of non-negative real numbers, i.e.,

$$\mathbb{R}_+ = \{x | x \geq 0\}\,.$$

A metric is also called a distance function or simply a distance. The most commonly used distance metric is the Euclidean distance, which is defined for two vectors $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^T$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_d)^T$, as

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

The Euclidean distance follows all the following properties for any $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{X}$:

$$\begin{array}{lll} \text{i)} & d(\boldsymbol{x}, \boldsymbol{y}) \geq 0 & \text{(non-negativity);} \\ \text{ii)} & d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x}) & \text{(symmetry);} \\ \text{iii)} & d(\boldsymbol{x}, \boldsymbol{y}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{y} & \text{(identity of indiscernibles);} \\ \text{iv)} & d(\boldsymbol{x}, \boldsymbol{z}) \leq d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z}) & \text{(triangle inequality).} \end{array} \tag{3}$$

These four properties specify that a distance should be non-negative (i.e., distance is always ≥ 0), symmetric (i.e., inward and outward trips should have the same distance), is 0 if and only if the two vectors are exactly the same (i.e., we did not move at all), and satisfy the triangle inequality (e.g., the shortest distance between two points is the length of the line segment connecting them)—these equations formalize our (maybe vague) intuition of what a distance should be.

These formal descriptions also generalize the concept *distance*. Any mapping $f : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{Y}$ satisfying all these four conditions is called a *metric*, which can be considered as a generalization (or more abstract version) of the distance concept. Except for the Euclidean distance, in this chapter we will also introduce two other metrics: the discrete metric and the ℓ_p metric.

The discrete metric is useful for comparing two categorical values. It is defined as

$$\rho(x, y) = \begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{if } x = y \end{cases}, \tag{4}$$

which simply indicates whether two categorical values are the same or not. It is easy to prove that ρ is a valid metric, i.e., satisfying all the four conditions for metrics.

1.2 Vector norm and metric

It is obvious that the Euclidean distance is closely related to the vector norm, because $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$. In linear algebra, a vector norm is defined more generally than $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T \boldsymbol{x}}$.

If we restrict our attention to real-valued vectors, a function $f : \mathbb{R}^d \mapsto \mathbb{R}_+$ is a *vector norm* if it satisfies the following three properties for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ and any value $c \in \mathbb{R}$:

$$\begin{array}{lll} \text{i)} & f(c\boldsymbol{x}) = |c| f(\boldsymbol{x}) & \text{(absolute scalability);} \\ \text{ii)} & f(\boldsymbol{x}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{0} & \text{(separates points);} \\ \text{iii)} & f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y}) & \text{(triangle inequality).} \end{array} \tag{5}$$

It is easy to show that

$$f(\boldsymbol{0}) = f(0\boldsymbol{x}) = 0 f(\boldsymbol{x}) = 0\,, \tag{6}$$

$$f(-\boldsymbol{x}) = f(-1 \times \boldsymbol{x}) = |-1| f(\boldsymbol{x}) = f(\boldsymbol{x})\,. \tag{7}$$

for an arbitrary vector $\boldsymbol{x}$ in $\mathbb{R}^d$. Hence, these properties not only imply symmetry, but also non-negativity: because

$$0 = f(\mathbf{0}) = f(\boldsymbol{x} + (-\boldsymbol{x})) \leq f(\boldsymbol{x}) + f(-\boldsymbol{x}) = 2f(\boldsymbol{x})$$

for arbitrary $\boldsymbol{x}$, we always have

$$f(\boldsymbol{x}) \geq 0\,.$$

That is, vector norm is always non-negative.

Simulating the Euclidean distance, if we define a mapping $\mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_+$ as $f(\boldsymbol{x} - \boldsymbol{y})$ for any vector norm f, it is trivial to verify that $f(\boldsymbol{x} - \boldsymbol{y})$ satisfies all the conditions for a metric. In other words, whenever we have a valid vector norm, we automatically obtain a corresponding metric. We say this metric is induced by the norm.

1.3 The ℓ_p norm and ℓ_p metric

In this section, we focus on the ℓ_p vector norm and its induced ℓ_p metric. The ℓ_p norm (or simply p-norm) is well-defined for $p \geq 1$ in $\mathbb{R}^d$, as

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{d} |x_i|^p\right)^{\frac{1}{p}}\,, \tag{8}$$

in which the ℓ_p norm of $\boldsymbol{x}$ is denoted as $\|\boldsymbol{x}\|_p$. When p is an irrational number and x is negative, x^p is not defined. Hence, the absolute value operator in $|x_i|$ cannot be omitted in this definition.

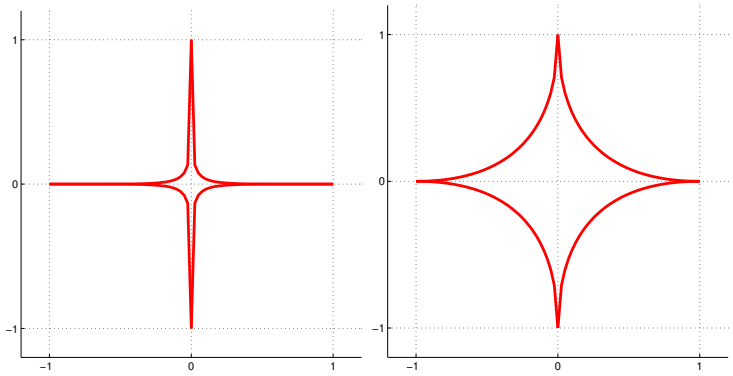
We usually omit the subscript for the ℓ_2 norm, simply writing $\|\boldsymbol{x}\|$ if $p = 2$. It is easy to prove that $\|\boldsymbol{x}\|_p$ satisfies properties i) and ii) in Equation 5. The property iii) is guaranteed by the Minkowski inequality, whose proof is omitted.[1] The Minkowski inequality states that for real numbers $x_i, y_i \geq 0$, $i = 1, 2, \ldots, d$, and $p > 1$,

$$\left(\sum_{i=1}^{d}(x_i + y_i)^p\right)^{1/p} \leq \left(\sum_{i=1}^{d} x_i^p\right)^{1/p} + \left(\sum_{i=1}^{d} y_i^p\right)^{1/p}\,. \tag{9}$$

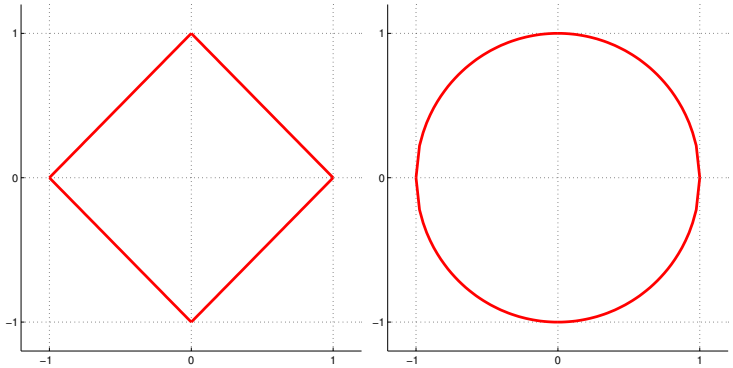
The equality holds if and only if there exists a $\lambda \in \mathbb{R}$ such that $x_i = \lambda y_i$ for all $1 \leq i \leq d$.

Why shall we specify $p \geq 1$ in the definition? Considering two unit vectors $\boldsymbol{x} = (1, 0, 0, \ldots, 0)^T$ and $\boldsymbol{y} = (0, 1, 0, \ldots, 0)^T$, we have $\|\boldsymbol{x} + \boldsymbol{y}\|_p = 2^{1/p}$ and $\|\boldsymbol{x}\|_p = \|\boldsymbol{y}\|_p = 1$. Hence, the triangle inequality requires $2^{1/p} \leq 2$, which specifies $p \geq 1$. When $0 < p < 1$, $\|\boldsymbol{x}\|_p$ is not a norm, but we can still use $\|\boldsymbol{x}\|_p$ to represent the mapping in Equation 8.

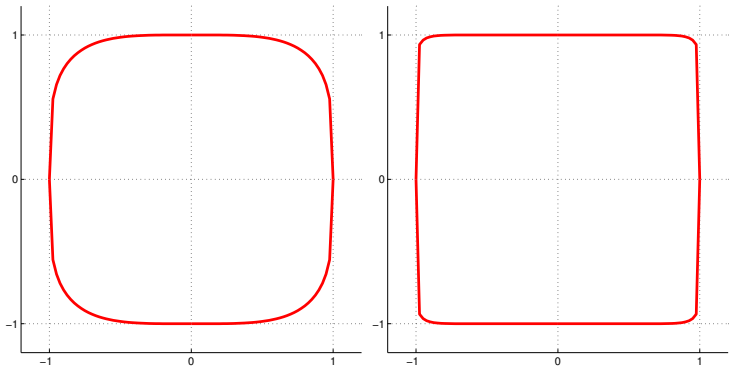
In Figure 1, we show the contours formed by points with $\|\boldsymbol{x}\|_p = 1$ for different p values. As shown by these figures, when $p \geq 1$ the regions enclosed



(a) $p = 0.25$ (b) $p = 0.5$



(c) $p = 1$ (d) $p = 2$



(e) $p = 4$ (f) $p = 16$

Figure 1: Illustration of 2D points that satisfy $\|\boldsymbol{x}\|_p = 1$.

by the contours (i.e., all points with $\|\boldsymbol{x}\|_p \leq 1$) are convex and $\|\boldsymbol{x}\|_p$ is a norm;

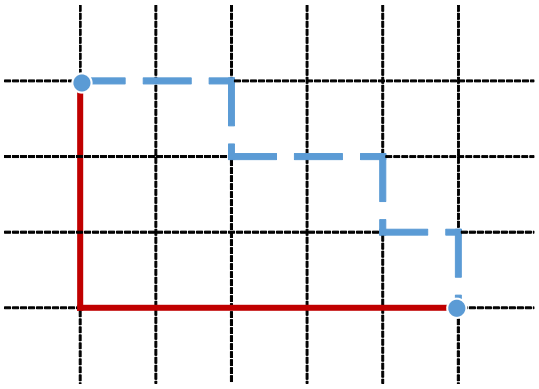


Figure 2: The Manhattan (city block) distance.

when $0 < p < 1$, the regions are non-convex and $\|\boldsymbol{x}\|_p$ is not a norm.

Figure 1 also hints us that the limit when $p \to \infty$ exists. When $p \to \infty$, we have

$$\ell_\infty \triangleq \lim_{p\to\infty} \|\boldsymbol{x}\|_p = \max\{|x_1|, |x_2|, \ldots, |x_d|\}\,. \tag{10}$$

When $p \to 0$, $\|\boldsymbol{x}\|_0$ is *not* a norm. However, in areas such as sparse learning methods, researchers use the term ℓ_0 *norm* to mean *the number of non-zero elements* in $\boldsymbol{x}$. We want to emphasize that this is not a valid norm. Sometimes a quotation mark is added to emphasize this fact (i.e., ℓ_0 "norm").

When $p \geq 1$, the ℓ_p norm induces an ℓ_p metric (which we denote as d_p)

$$d_p(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_p\,. \tag{11}$$

We also call the ℓ_p metric the ℓ_p distance. It is obvious that the ℓ_2 distance is the Euclidean distance.

The ℓ_1 distance and associated ℓ_1-norm is also widely used (e.g., in sparse learning methods). The ℓ_1 distance is also called the Manhattan distance or the city block distance, computed as

$$d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} |x_i - y_i|\,. \tag{12}$$

In a city whose roads form regular grids (such as those in Manhattan, New York city), the distance between two locations are the number of blocks that are between them, as shown in Figure 2, no matter the red or the blue path (or any other route along the roads) is followed..

The ℓ_∞ distance measures the largest distance in any single dimension:

$$d_\infty(\boldsymbol{x}, \boldsymbol{y}) = \max\{|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_d - y_d|\}\,.$$

[1]The Minkowski inequality is named after Hermann Minkowski, a Lithuanian-German mathematician.

When $p > q > 0$, we have

$$\|\boldsymbol{x}\|_p \leq \|\boldsymbol{x}\|_q \,. \tag{13}$$

In other words, as p increases, the ℓ_p norm will decrease or remain unchanged. We leave the proof of this inequality as an exercise.

1.4 Distance metric learning

Another type of vector norm induced metric is widely used. Let $\boldsymbol{x} \in \mathbb{R}^d$ and G be any $d \times d$ positive definite matrix, then the mapping

$$f(\boldsymbol{x}) = \|G\boldsymbol{x}\| \tag{14}$$

defines a valid vector norm. The distance metric associated with it is $\|G(\boldsymbol{x}-\boldsymbol{y})\|$. However, the squared distance is often used, as

$$d_A^2(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T A(\boldsymbol{x} - \boldsymbol{y}) \,, \tag{15}$$

in which $A = G^T G$ is a $d \times d$ positive definite square matrix.

This kind of distance is called the (squared) Mahalanobis distance. If our data follow a Gaussian distribution $N(\boldsymbol{\mu}, \Sigma)$ and $\Sigma \neq I_d$, then it is inappropriate to use $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$ to represent the distance between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. If both points are in the major axis (corresponding to the largest eigenvalue λ_1 of Σ, say $\lambda_1 = 100$), $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| = 1$ indicates the points are close to each other (roughly only 10% of the standard deviation in that direction). However, if these points are in the minor axis (corresponding to the smallest eigenvalue, e.g., $\lambda_d = 0.01$), $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| = 1$ in fact means a 10 times standard deviation distance in that direction!

We know the whitening transform $\Sigma^{-1/2}(\boldsymbol{x} - \bar{\boldsymbol{x}})$ translates, rotates and scales the distribution, such that it becomes a spherical Gaussian. In other words, all dimensions are independent and in the same scale. Hence, the squared Euclidean distance in the transformed space is a good distance metric, which is exactly the squared Mahalanobis distance, with $G = \Sigma^{-1/2}$ and $A = \Sigma^{-1}$.

The squared Mahalanobis distance is a special case of the squared distance in Equation 15, with A fixed to Σ^{-1}. When the data is not Gaussian, however, using the inverse covariance matrix is not optimal. Hence, *distance metric learning* tries to learn a good A matrix from the training data. In this case, we use $\|\boldsymbol{x} - \boldsymbol{y}\|_A$ to denote the learned distance between $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e.,

$$\|\boldsymbol{x} - \boldsymbol{y}\|_A^2 = (\boldsymbol{x} - \boldsymbol{y})^T A(\boldsymbol{x} - \boldsymbol{y}) \,.$$

If we are given a training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ in a binary classification problem, we can learn a good distance metric by optimizing the following problem:

$$\min_{A \in \mathbb{R}^d \times \mathbb{R}^d} \quad \sum_{\substack{1 \leq i < j \leq n \\ y_i = y_j}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_A^2 \tag{16}$$

$$\text{s.t.} \quad \sum_{\substack{1 \le i < j \le n \\ y_i \neq y_j}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_A^2 \ge 1 \tag{17}$$

$$A \succeq 0\,. \tag{18}$$

The learned distance metric should make the distances between examples in the same class to be as small as possible (via minimizing the objective), while keeping the distances between examples in different classes to be large (via the first constraint.) Hence, the learned optimal distance metric (or equivalently, the optimal positive semi-definite matrix A) is useful for classification of the two classes.

In distance metric learning, A is required to be positive semi-definite, rather than positive definite. With this relaxation, $\|\boldsymbol{x} - \boldsymbol{y}\|_A$ is no longer a metric in the strict sense, because $\|\boldsymbol{x} - \boldsymbol{y}\|_A = 0$ is possible even if $\boldsymbol{x} \neq \boldsymbol{y}$. However, this relaxation allows us to learn inherently lower dimensional representations. For example, if $G \in \mathbb{R}^k \times \mathbb{R}^d$ and $k < d$, then $A = G^T G$ is positive semi-definite but not positive definite, and $G\boldsymbol{x}$ has fewer dimensions than $\boldsymbol{x}$.

1.5 The mean as a similarity measure

Now we turn our attention to similarity measures. We have seen different similarity measures, e.g., the dot product and those transformed from a distance metric (such as the RBF kernel). Another group of similarity measures are in fact the average values (i.e., the mean). For example, if we want to find the similarity of two *distributions*, as illustrated in Figure 3, a natural idea is to use the area of their common region (i.e., the blue region) as a numerical similarity measure. Because the area under each distribution is 1 ($\int p(x)\,\mathrm{d}x = 1$ for a valid distribution $p(x)$), the similarity will always be between 0 and 1, which fits our intuition well.

Given two distributions $p_X(x)$ and $p_Y(y)$, a particular value v will contribute

$$\min(p_X(v), p_Y(v))$$

to the similarity in Figure 3, which means the similarity is

$$\int \min(p_X(v), p_Y(v))\,\mathrm{d}v\,.$$

The minimum between two non-negative values, $\min(x, y)$, in fact, is a special type of mean value between x and y, called the generalized mean.

The generalized mean (also called the power mean) with exponent p of a set of *positive* values $x_1, x_2, \ldots, x_n$ are defined as

$$M_p(x_1, x_2, \ldots, x_n) = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^p\right)^{\frac{1}{p}}\,. \tag{19}$$

The power mean is very similar to the p-norm (Equation 8). However, there are a few important differences.

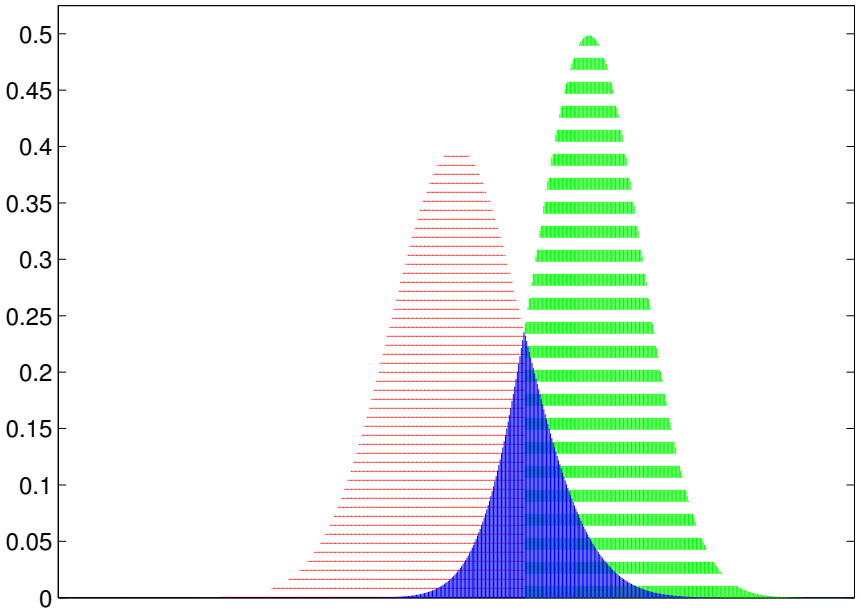


Figure 3: Illustration of the similarity of two distributions.

- First, power mean requires x_i to be positive but p-norm uses the absolute value of x_i;
- Second, the $\frac{1}{n}$ term in power mean does *not* appear in the p-norm; and,
- More importantly, power mean is well defined for all real numbers, while p-norm requires $p \geq 1$!

The power mean for the following special p values are given their designated names:

$$M_{-\infty}(x_1, \ldots, x_n) = \lim_{p \to -\infty} M_p(x_1, \ldots, x_n) = \min\{x_1, \ldots, x_n\}\,, \tag{20}$$

(minimum value)

$$M_{-1}(x_1, \ldots, x_n) = \frac{n}{x_1^{-1} + \cdots + x_n^{-1}}\,, \tag{21}$$

(harmonic mean)

$$M_0(x_1, \ldots, x_n) = \lim_{p \to 0} M_p(x_1, \ldots, x_n) = \sqrt[n]{\prod_{i=1}^{n} x_i}\,,$$

(geometric mean)
(22)

$$M_1(x_1, \ldots, x_n) = \frac{x_1 + \cdots + x_n}{n}\,,$$

(arithmetic mean)
(23)

$$M_2(x_1, \ldots, x_n) = \sqrt{\frac{x_1^2 + \cdots + x_n^2}{n}}\,,$$

(square mean) (24)

$$M_\infty(x_1, \ldots, x_n) = \lim_{p \to \infty} M_p(x_1, \ldots, x_n) = \max\{x_1, \ldots, x_n\}\,,$$

(maximum value)
(25)

The minimum, geometric, and maximum mean values are defined using limits. The harmonic mean has been used previously: the F1 score is the harmonic mean of precision and recall. The arithmetic mean is the most commonly used mean value.

The power mean with exponent p and the p-norm differ only by a factor $n^{1/p}$. This slight difference, however, leads to very different properties. When $p_1 < p_2$, we have

$$M_{p_1}(x_1, x_2, \ldots, x_n) \le M_{p_2}(x_1, x_2, \ldots, x_n)\,, \tag{26}$$

that is, M_p is a non-decreasing function of p in the entire real domain. The equality holds if and only if $x_1 = x_2 = \cdots = x_n$. In contrast, $\|\boldsymbol{x}\|_p \ge \|\boldsymbol{x}\|_q$ if $0 < p < q$!

Figure 4 illustrates various power mean values for two positive numbers a and b, which clearly shows that

$$M_{-\infty} < M_{-1} < M_0 < M_1 < M_2 < M_\infty$$

when $a \ne b$.

1.6 Power mean kernel

Some similarity functions, such as dot-product or RBF, can be used as kernel functions in kernel methods. The same argument also applies to the power mean, but *only* when $p \le 0$. Given two *non-negative*[2] vectors $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^T$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_d)^T$ in which $x_i, y_i \ge 0$ for all $1 \le i \le d$, the power mean kernel is defined (for $p \le 0$) as

$$M_p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} M_p(x_i, y_i)\,. \tag{27}$$

[2] We assume $0^p = 0$ even for $p \le 0$.

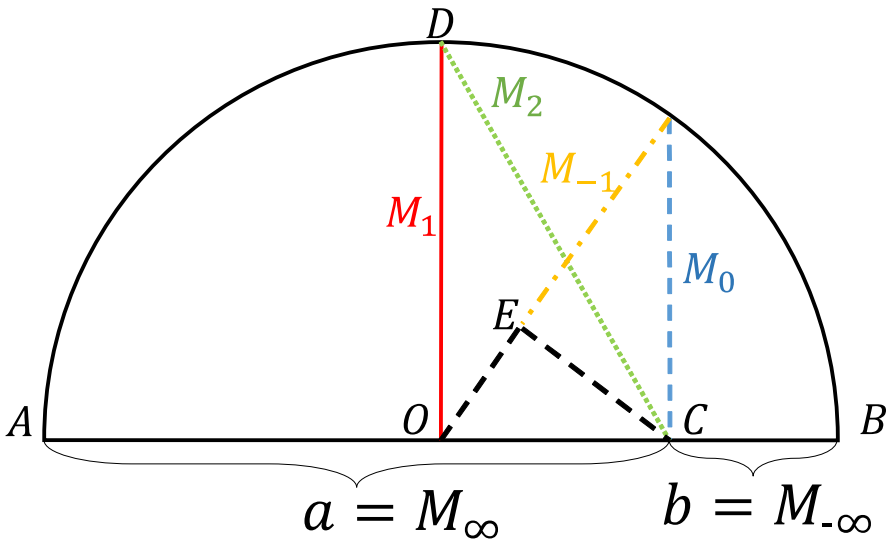


Figure 4: Illustration of different power mean values.

When we want to compare two distributions, e.g., when the two distributions are represented as two histograms whose values are non-negative, the power mean kernel family usually produce better similarity measure than commonly used kernels such as dot-product, RBF, or polynomial. A few special p values in the power mean kernel family also lead to kernels that are defined in statistics:

$$\begin{aligned} &M_0(\boldsymbol{x}, \boldsymbol{y}) = \textstyle\sum_{i=1}^{d} \sqrt{x_i y_i}\,, && \text{(Hellinger's kernel)} \\ &M_{-1}(\boldsymbol{x}, \boldsymbol{y}) = \textstyle\sum_{i=1}^{d} \frac{2x_i y_i}{x_i + y_i}\,, && (\chi^2 \text{ kernel}) \\ &M_{-\infty}(\boldsymbol{x}, \boldsymbol{y}) = \textstyle\sum_{i=1}^{d} \min(x_i, y_i)\,. && \text{(histogram intersection kernel)} \end{aligned} \tag{28}$$

Hellinger's kernel is the similarity measure corresponding to the Hellinger distance (also called the Bhattacharyya distance), which is an established measure for quantifying the distance (or dissimilarity) between two distributions. The χ^2 kernel is closely related to the χ^2-squared test (or written as chi-squared test) and the χ^2-distance, which is a widely used measure to quantify the difference between two distributions. And, the histogram intersection kernel (HIK), as we mentioned above, is an intuitive way to measure the similarity between two distributions, which is illustrated in Figure 3.

The power mean kernel family is a generalization of all these special kernels, which are suitable for the comparison of distributions (or histograms). One benefit of the power mean kernel is that there is efficient algorithms for learning and testing the entire power mean family, and it can also provide smooth alternatives to the histogram intersection kernel (which is in fact not differentiable).

Figure 5a shows the curve for $M_{-\infty}(0.2, x)$ (i.e., the HIK between x and a fixed value 0.2) and $M_{-8}(0.2, x)$, while Figure 5b shows the power mean approx-

imation of HIK when $p = -32$. When $p = -32$, the power mean approximation to HIK is already fairly accurate.

2 Data transformation and normalization

When we seek a mapping $f : \mathcal{X} \mapsto \mathcal{Y}$ and $\mathcal{Y} = \mathbb{R}$, the task is called regression. Although we will not introduce regression methods in detail, let us motivate the need for data transformation and normalization using linear regression as an example.

In a regression task, we are given a set of training examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. We want to find a function f such that for any point $(\boldsymbol{x}, y)$ sampled from the same distribution as the training set, $f(\boldsymbol{x}) \approx y$. This approximation is usually performed by minimizing the residue $f(\boldsymbol{x}) - y$, e.g., by minimizing $\mathbb{E}[(f(\boldsymbol{x}) - y)^2]$.

2.1 Linear regression

In linear regression, we assume f approximates y using a linear combination of components of $\boldsymbol{x}$, i.e.,

$$y_i = \boldsymbol{x}_i^T \boldsymbol{\beta} + \epsilon_i \,, \tag{29}$$

in which $\boldsymbol{\beta} \in \mathbb{R}^d$ is the parameter of linear regression and ϵ_i is the residue for approximating y_i using $\boldsymbol{x}_i$. Hence, the training objective is to minimize the residues, by

$$\boldsymbol{\beta}^\star = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(\boldsymbol{x}_i^T \boldsymbol{\beta} - y_i\right)^2 . \tag{30}$$

This objective can be further simplified using the matrix notation. Using the block matrix notation, we define

$$X = \begin{bmatrix} \boldsymbol{x}_1^T \\ \hline \boldsymbol{x}_2^T \\ \hline \vdots \\ \hline \boldsymbol{x}_n^T \end{bmatrix} \in \mathbb{R}^n \times \mathbb{R}^d, \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n \,, \tag{31}$$

and the linear regression optimization becomes

$$\boldsymbol{\beta}^\star = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - X\boldsymbol{\beta}\|^2 = \arg\min_{\boldsymbol{\beta}} \left(\boldsymbol{\beta}^T X^T X \boldsymbol{\beta} - 2\boldsymbol{y}^T X \boldsymbol{\beta}\right) . \tag{32}$$

In the last equality we have dropped the term $\boldsymbol{y}^T \boldsymbol{y}$ because it does not depend on $\boldsymbol{\beta}$.

Because $\frac{\partial\left(\boldsymbol{\beta}^T X^T X \boldsymbol{\beta} - 2\boldsymbol{y}^T X \boldsymbol{\beta}\right)}{\partial \boldsymbol{\beta}} = 2X^T X \boldsymbol{\beta} - 2X^T \boldsymbol{y}$, we know

$$X^T X \boldsymbol{\beta} = X^T \boldsymbol{y} \tag{33}$$

is a necessary condition for the optimal value. Hence, the solution to linear regression is

$$\boldsymbol{\beta}^\star = (X^T X)^{-1} X^T \boldsymbol{y} \,. \tag{34}$$

When the matrix $X^T X$ is not invertible, a practical way is to use

$$\boldsymbol{\beta}^\star = X^+ \boldsymbol{y} \,,$$

in which X^+ is the Moore-Penrose pseudoinverse of X. Given any testing example $\boldsymbol{x}$, its associated y is approximated as

$$y \approx \boldsymbol{x}^T \boldsymbol{\beta}^\star \,. \tag{35}$$

Now we consider a problem with 10 examples in 2D: $\boldsymbol{x} = (x_1, x_2)^T$. The two dimensions correspond to the height and waist circumference of a male person. The heights (x_1) are measured in centimeter, and the ten heights are generated as

$$x_{i,1} = 169 + i \quad 1 \le i \le 10 \,.$$

The waist-to-height ratio for a healthy adult man is considered to be in the range $[0.43, 0.52]$. Hence, we generate the waist circumference (x_2) as

$$x_{i,2} = (0.46 + 0.02v) x_{i,1} \,,$$

where the variation $v \sim N(0,1)$ is randomly sampled for each example independently. We expect the waist-to-height ratio to be between 0.40 and 0.52 (i.e., within the "3σ" range).

The labels y is generated using $\boldsymbol{\beta} = (1,1)^T$ and a white noise $0.0001N(0,1)$, that is,

$$y_i = x_{i,1} + x_{i,2} + \epsilon_i \tag{36}$$

$$\epsilon_i \sim 0.0001N(0,1) \,. \tag{37}$$

The estimated $\boldsymbol{\beta}$ will be slightly different from run to run because of the noise. However, because the label noise $0.0001N(0,1)$ is very small compared to the scale of x_1 and x_2, the estimation is fairly accurate, e.g., $\boldsymbol{\beta}^\star = (0.9999, 1.0197)^T$ in one of our runs.

Suppose, however, the waist circumference and the height are measured by two different individuals, using centimeter for heights by one person and meter for waist circumference by another. Then, an example (x_1, x_2) becomes $(x_1, 0.01x_2)$. The linear regression model has much larger estimation errors, and $\boldsymbol{\beta}^\star = (0.9992, 1.1788)^T$ is one example. We want to point out that: even though the second dimension is now much smaller than the first, it is still 50 to 100 times larger than the noise $0.0001N(0,1)$. Hence, wrong scales for some feature dimensions may cause serious troubles in machine learning and pattern recognition, if the difference in scale is not underpinned by certain properties in their data generation process.

2.2 Feature normalization

Data normalization can solve this problem. We first find the minimum and maximum value of the j-th dimension ($1 \le j \le d$) in the training set, that is,

$$x_{\min,j} = \min_{1 \le i \le n} x_{i,j}, \qquad \text{and} \qquad x_{\max,j} = \max_{1 \le i \le n} x_{i,j}\,. \tag{38}$$

Then, we can normalize the range of the j-th dimension to $[0, 1]$ by

$$\hat{x}_{i,j} = \frac{x_{i,j} - x_{\min,j}}{x_{\max,j} - x_{\min,j}}\,. \tag{39}$$

Note that $\min_{1 \le i \le n} \hat{x}_{i,j} = 0$ and $\max_{1 \le i \le n} \hat{x}_{i,j} = 1$.

This per-dimension normalization is also a learning process, because it learns how to preprocess the data using the training set. Hence, we need to retain this model for normalization, which includes $x_{\min,j}$ and $x_{\max,j}$ for all $1 \le j \le d$, i.e., $2d$ numbers as the normalization model's parameters. Finally, the normalized data are used to learn a model f for classification, regression, or other tasks.

When a test example $\boldsymbol{x}$ is presented to us, we first use the normalization parameters to convert $\boldsymbol{x}$ to the normalized version $\hat{\boldsymbol{x}}$, and then output the prediction using $f(\hat{\boldsymbol{x}})$. A commonly occurring error for novices is to learn the maximum and minimum values for dimensions in the test set, and use these values to normalize the test examples. Test data cannot be used apart from being tested.

Similarly, in a cross-validation process, we shall learn the normalization parameters for each fold separately. The available examples are divided into different training and validation sets in different folds. In one fold, we learn the normalization parameters using the training set for this fold, and apply it to all examples. We have to learn different normalization parameters in different folds.

This seemingly trivial normalization trick has more factors to weigh in.

- If we want a different range than $[0, 1]$, e.g., $[-1, +1]$, it is also possible, by

$$\hat{x}_{i,j} = 2\left(\frac{x_{i,j} - x_{\min,j}}{x_{\max,j} - x_{\min,j}} - 0.5\right)\,. \tag{40}$$

  In fact, we can stretch the feature values to any range $[a, b]$ ($a < b$).

- If $x_{\max,j} = x_{\min,j}$ for some dimension j, it means all values in this dimension are the same and the normalization equation is not well defined. However, a constant dimension means a useless one. Hence, we can simply discard all such dimensions.

- A vector is called *sparse* if many of its dimensions are zeros. A sparse dataset not only has many zeros in every row, it will also have many zeros in each dimension (column in X). Equation 39 will normalize a zero value to

$$-\frac{x_{\min,j}}{x_{\max,j} - x_{\min,j}}\,,$$

which is nonzero if $x_{\min,j} \neq 0$. However, if a zero value in the original data means "null" or "no information" based on its data generation process, this normalization is not welcome. In that case, we can specify 0 is always normalized to 0.

- After normalization, the values in test examples could be smaller than 0 or larger than 1. In some problems the $[0, 1]$ range is required by the algorithm to learn f, e.g., the power mean kernels require all feature values to be non-negative. We can set all negative values to 0 and all values larger than 1 to 1. We can still use this strategy if the learning algorithm does not require a strict $[0, 1]$ range. However, we can also leave these values as they are.

If there is a reason to believe a dimension j is a Gaussian, then it is probably better to normalize that dimension to a standard normal distribution, rather than to the range $[0, 1]$. We can first find μ_j and σ_j as the mean and standard deviation of that dimension based on the training set, and normalize it as

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j} \,. \tag{41}$$

Of course, we will use the μ_j and σ_j computed from the training set to normalize all examples, including the test ones. We have seen this type of normalization done implicitly in the whitening transform.

In some applications, we have reasons to require the scale of examples $\boldsymbol{x}$ to be roughly equal, rather than requiring the scale of feature dimensions to match. The ℓ_2 normalization normalizes any (training or testing) example $\boldsymbol{x}$ to be a unit vector, that is,

$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} \,. \tag{42}$$

Another commonly used per-example normalization is the ℓ_1 normalization, which is

$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_1} \,. \tag{43}$$

After the ℓ_1 normalization, the dimensions of every example will sum to 1 if the values are non-negative. It is a good idea to normalize a histogram using the ℓ_1 normalization. If necessary, other ℓ_p norm can be used to normalize our data, too.

In short, proper normalization is essential in many learning and recognition methods and systems, no matter they are deep or not. However, which type of normalization is the most suitable for a particular dataset or problem? There is no silver bullet in making this choice. Visualizing and studying the properties (e.g., distributions) of your input, interim, and output data should be a useful strategy in general. After properly understanding your data's properties, you will be able to find a matching normalization strategy for them.

2.3 Data transformation

Feature normalization is a type of data transformation. Converting a distance metric to a similarity measure is also a transformation. In this section, we briefly introduce another type of transformation that converts arbitrary real numbers into a limited range. This type of transformation is useful in various methods, e.g., to convert the value of a discriminant function (which is a real number) to $[0, 1]$ and then we can interpret the transformed value as a probability.

One type of conversion can convert a categorical dimension to a vector, such that the Euclidean distance on the vector is proportional to the discrete distance on the categorical data. For example, if a categorical dimension has three possible values `a,b,c`. Then, we can transform `a` to a short vector $\boldsymbol{t}_a = (1, 0, 0)$, `b` to $\boldsymbol{t}_b = (0, 1, 0)$ and `c` to $\boldsymbol{t}_c = (0, 0, 1)$. In other words, $\boldsymbol{t}_x$ is the histogram of a singleton set $\{\mathtt{x}\}$, which only contains one element `x`.

It is easy to prove that $\rho(\mathtt{x}, \mathtt{y}) = \frac{1}{\sqrt{2}}\|\boldsymbol{t}_x - \boldsymbol{t}_y\|$, where $\mathtt{x}, \mathtt{y} \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, and

$$1 - \rho(\mathtt{x}, \mathtt{y}) = \boldsymbol{t}_x^T \boldsymbol{t}_y \,.$$

For example, if a problem has both categorical and real-valued features, we can convert every categorical dimension to a short vector, and the SVM method can be applied on the transformed data for both training and testing.

To convert a set of values to another set of values that could form a probability mass function, the *softmax* transform is popular. The softmax transformation is in fact a nonlinear normalization, too. Given a vector of arbitrary values $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^T$, the softmax function converts it to another d-dimensional vector $\boldsymbol{z}$. Components of the new vector are non-negative and sum to 1, hence, could be interpreted as probabilities $\Pr(y = i|\boldsymbol{x})$. The softmax transformation is defined as

$$z_i = \frac{\exp(x_i)}{\sum_{j=1}^{d} \exp(x_j)} \,. \tag{44}$$

The data could be scaled to form a better probability distribution, e.g.,

$$z_i = \frac{\exp(\gamma x_i)}{\sum_{j=1}^{d} \exp(\gamma x_j)} \,, \quad \gamma > 0 \,.$$

To convert one value to another value in a limited range, there is also the logistic sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \,, \tag{45}$$

which is illustrated in Figure 6. Its range is $(0, +1)$; $\sigma(\infty) = \lim_{x\to\infty} \sigma(x) = 1$ and $\sigma(-\infty) = \lim_{x\to-\infty} \sigma(x) = 0$.

The logistic sigmoid function σ is a special case of the logistic function family, which has the form

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \,, \tag{46}$$

where L, k and x_0 are the maximum value, steepness (of the logistic curve) and the midpoint, respectively.

A transformation similar to the logistic sigmoid is the hyperbolic tangent function tanh,

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \,. \tag{47}$$

The curve of tanh is similar to that of logistic sigmoid. However, its range is $(-1, +1)$. Both logistic sigmoid and hyperbolic tangent have been widely used in neural network models.

In linear regression, if we transform the linear combination $\boldsymbol{x}^T\boldsymbol{\beta}$ using a logistic function, we arrive at a commonly used classification model in statistics, called the logistic regression.[3] Unlike linear regression, logistic regression finds nonlinear classification boundaries.

In a binary problem, we assume the probability of $\boldsymbol{x}$ belonging to the positive class has the following form

$$\Pr(y = 1|\boldsymbol{x}) \approx f(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{x}^T\boldsymbol{\beta}}} \,, \tag{48}$$

and use methods such as maximum likelihood estimation to find a good $\boldsymbol{\beta}$ value. It is convenient to use $\mathcal{Y} = \{0, 1\}$ in logistic regression, because we can write the objective as maximizing

$$\prod_{i=1}^{n} f(\boldsymbol{x}_i)^{y_i} \left(1 - f(\boldsymbol{x}_i)\right)^{1-y_i} \,, \tag{49}$$

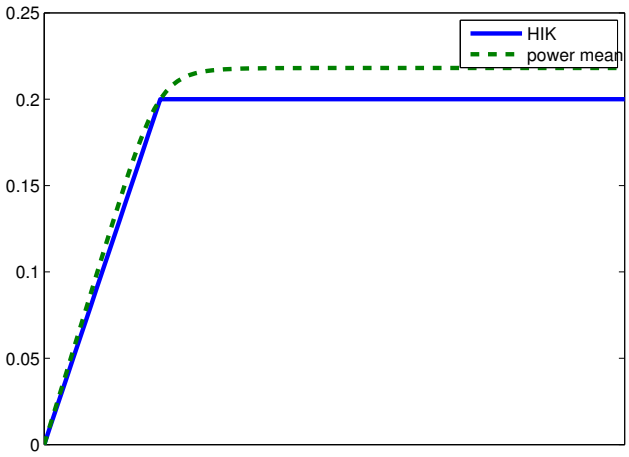
in which $\Pr(y_i = 0|\boldsymbol{x}_i) = 1 - \Pr(y_i = 1|\boldsymbol{x}_i) \approx 1 - f(\boldsymbol{x}_i)$.

Considering the i-th training example $(\boldsymbol{x}_i, y_i)$, the above objective says if $y_i = 1$ we want $\Pr(y_i = 1|\boldsymbol{x}_i) \approx f(\boldsymbol{x}_i)$ to be large; and when $y_i = 0$, we want $\Pr(y_i = 0|\boldsymbol{x}_i) = 1 - \Pr(y_i = 1|\boldsymbol{x}_i) \approx 1 - f(\boldsymbol{x}_i)$ to be large. Overall, we want the probability distribution predicted by our model to match the distribution computed from the training set. In the next chapter, we will also very briefly introduce multinomial logistic regression, which extends the logistic regression model to multiclass classification problems.

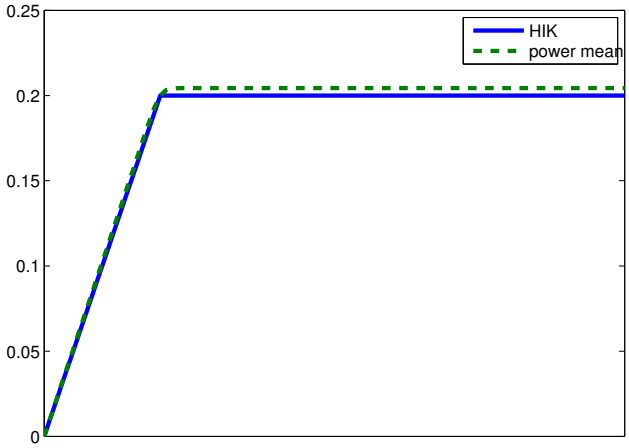
Logistic regression is a representative *discriminative* probabilistic model, because it directly models the posterior distribution $\Pr(y|\boldsymbol{x})$.

To solve the logistic regression optimization is beyond the scope of this book. However, before we conclude this chapter, we want to add that logistic regression is a popular approach in classification, especially when probability estimates are required. Through some transformations, other classifiers such as SVM can also output a probability estimation, but that estimation is usually inferior to the probability estimated by logistic regression.

[3] Although the word "regression" is used in its name, logistic regression is a classification method.



(a) $p = -8$



(b) $p = -32$

Figure 5: Use the power mean kernel to approximate the histogram intersection kernel.

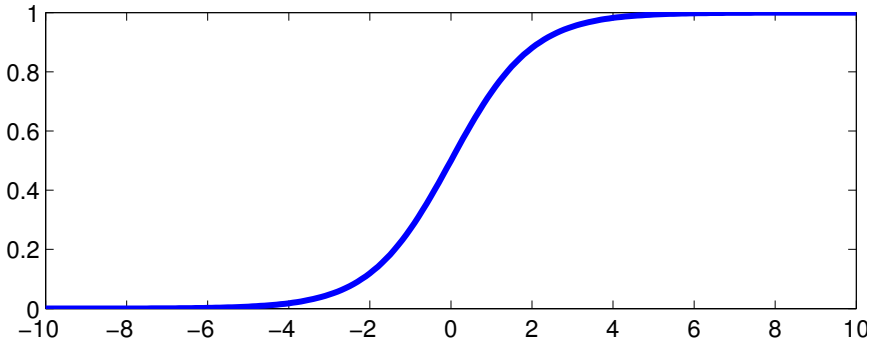


Figure 6: The logistic sigmoid function

Exercises

1. Principal component analysis (PCA) transforms a vector $\boldsymbol{x} \in \mathbb{R}^D$ to a lower dimensional vector $\boldsymbol{y} \in \mathbb{R}^d$ ($d < D$) using

$$\boldsymbol{y} = E_d^T (\boldsymbol{x} - \bar{\boldsymbol{x}}) ,$$

in which $\bar{\boldsymbol{x}}$ is the sample mean of $\boldsymbol{x}$, and E_d is a $D \times d$ matrix formed by the top d eigenvectors of the sample covariance matrix of $\boldsymbol{x}$ (cf. Chapter 5).

Let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be any two samples of $\boldsymbol{x}$, and $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ be the PCA transformed version of them. Show that

$$d_A^2(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2$$

is a valid distance metric in the family defined by Equation 15. What shall be assigned to be the matrix A?

2. (Locally linear embedding) PCA is a *linear* transformation that approximately keeps the dissimilarity (aka, distance) between any two data points: $\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2 \approx \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2$ because $E_d^T E_d \approx I$. It is particularly useful when *all* data points approximately lie on a linear subspace of $\mathbb{R}^D$ (i.e., a global subspace).

However, the global linear subspace assumption often breaks down in real-world data that are complex in nature. In that case, it is reasonable to assume *local* linear relationships and use these relationships for dimensionality reduction or better similarity (dissimilarity) measure. Locally linear embedding (LLE), proposed by Roweis and Saul , is one such method.

(a) **Local geometry.** The local geometry around an example $\boldsymbol{x}_i$ is represented by a local linear reconstruction. Let there be n examples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$. For any example $\boldsymbol{x}_i$, LLE firstly finds $\boldsymbol{x}_i$'s nearest neighbors, and use a linear combination of these nearest neighbors to reconstruct $\boldsymbol{x}_i$. That is, for $\boldsymbol{x}_i$, LLE wants to minimize the reconstruction error

$$e_i = \left\| \boldsymbol{x}_i - \sum_{j=1}^{n} w_{ij} \boldsymbol{x}_j \right\|^2 , \tag{50}$$

in which w_{ij} is the linear weight for $\boldsymbol{x}_j$ in the reconstruction. Note that $w_{ij} = 0$ if $\boldsymbol{x}_j$ is not among the nearest neighbors of $\boldsymbol{x}_i$ (hence $w_{ii} = 0$). An additional constraint is that

$$\sum_{j=1}^{n} w_{ij} = 1 \tag{51}$$

for any $1 \leq i \leq n$. This constraint makes the solution for w unique.

On the entire dataset, LLE seeks a matrix W (with $[W]_{ij} = w_{ij}$) that both satisfies all these constraints and minimizes the total error $\sum_{i=1}^{n} e_i$. Suppose K nearest neighbors are used, find the optimal solution for w_{ij} $(1 \leq i, j \leq n)$.

(b) **Invariance.** We can apply the same operation to all examples in the dataset.

i. *Rotation*: $\boldsymbol{x}_i \leftarrow Q\boldsymbol{x}_i$ with $QQ^T = Q^TQ = I$ for all $1 \leq i \leq n$.

ii. *Translation*: $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \boldsymbol{t}$ with $\boldsymbol{t} \in \mathbb{R}^D$ for all $1 \leq i \leq n$.

iii. *Scaling*: $\boldsymbol{x}_i \leftarrow s\boldsymbol{x}_i$ with $s \neq 0$ for all $1 \leq i \leq n$.

Show that the optimal solution for w_{ij} is invariant to any one of these three operations.

(c) **New representation: formulation.** In the next step, LLE seeks a shorter vector $\boldsymbol{y}_i \in \mathbb{R}^d$ with $d \ll D$ as a new representation for $\boldsymbol{x}_i$. The main purpose is that the local geometry (i.e., w_{ij}) is preserved *and* to reduce unnecessary degrees of freedom in the new representation. To find optimal $\boldsymbol{y}_i$, the following optimization is solved.

$$\underset{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n}{\arg\min} \quad \sum_{i=1}^{n} \left\| \boldsymbol{y}_i - \sum_{j=1}^{n} w_{ij} \boldsymbol{y}_j \right\|^2 \tag{52}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \boldsymbol{y}_i = \boldsymbol{0} \tag{53}$$

$$\sum_{i=1}^{n} \boldsymbol{y}_i \boldsymbol{y}_i^T = I\,, \tag{54}$$

in which $\boldsymbol{0}$ is a vector of all zeros and I is an identity matrix with appropriate dimensions. w_{ij} are the optimal values learned in the previous step.

Explain (intuitively) why this optimization keeps the local geometry, and what are the effects of the two conditions. In terms of rotation, translation and scaling, which of these degrees of freedom are eliminated? If there are some degrees of freedom still alive, are they adversely affecting the new representation?

(d) **New representation: simplification.** Let W be an $n \times n$ matrix with $[W]_{ij} = w_{ij}$, and let M be an $n \times n$ matrix defined by $M = (I - W)^T(I - W)$. Show that the above optimization objective is equivalent to

$$\sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij} \boldsymbol{y}_i^T \boldsymbol{y}_j\,. \tag{55}$$

(e) **New representation: solution.** First, show that 1) M is positive semi-definite; and, 2) $\boldsymbol{1}$ is an eigenvector of M.

Then, let $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots$ be eigenvectors of M sorted in the *ascending* order of their corresponding eigenvalues, and E_d be a $n \times d$ matrix defined by $E_d = [\boldsymbol{\xi}_2 | \boldsymbol{\xi}_3 | \ldots | \boldsymbol{\xi}_{d+1}]$. Matrix analysis results tell us that if we set $\boldsymbol{y}_i$ to be the i-th row of E_d, the objective is minimized (which is easy to prove). Hence, the i-th eigenvector contains the $(i-1)$-th dimension of the new representation for all n examples.

Show that the two constraints are satisfied, hence the rows in E_d are indeed the optimal solutions for $\boldsymbol{y}_i$. Why $\boldsymbol{\xi}_1$ is discarded?

(f) The page https://www.cs.nyu.edu/~roweis/lle/ lists some useful resources for LLE, including publications, codes, visualizations and demonstrations. Browse the resources in this page and play with the codes and demos.

3. In this exercise, we prove that $\|\boldsymbol{x}\|_p$ $(p > 0)$ is a non-increasing function of p. In other words, if $0 < p < q$, prove

$$(|x_1|^p + |x_2|^p + \cdots + |x_d|^p)^{\frac{1}{p}} \geq (|x_1|^q + |x_2|^q + \cdots + |x_d|^q)^{\frac{1}{q}} \ . \tag{56}$$

(a) Show that to prove Equation 56 is equivalent to prove

$$(x_1^p + x_2^p + \cdots + x_d^p)^{\frac{1}{p}} \geq (x_1^q + x_2^q + \cdots + x_d^q)^{\frac{1}{q}} \ . \tag{57}$$

under the additional constraint $x_i \geq 0$ for all $1 \leq i \leq d$.

(b) Denote $r = \frac{q}{p}$ $(0 < p < q)$ and assume $x_i \geq 0$ for all $1 \leq i \leq d$, prove that Equation 57 is equivalent to

$$(y_1 + y_2 + \cdots + y_d)^r \geq (y_1^r + y_2^r + \cdots + y_d^r) \ , \tag{58}$$

in which $y_i = x_i^p$.

(c) Prove that Equation 58 holds when $r > 1$ and $y_i \geq 0$ $(i = 1, 2, \ldots, d)$. (Hint: use Taylor's expansion for $d = 2$ and mathematical induction for $d > 2$.)

(d) The above three steps prove Equation 56. Note that one simplification (assuming non-negative numbers) and one change of variables $(y_i = x_i^p)$ transformation are used in this proof. Both simplification do not change the property of the original problem. After the simplifications, Equation 58 is easy (if not trivial) to prove. Try to prove Equation 56 without using these simplifications, e.g., by showing

$$\frac{\mathrm{d}\|\boldsymbol{x}\|_p}{\mathrm{d}p} \leq 0$$

when $p > 0$ and $d \geq 1$. How do you compare the level of difficulty of these two different ways of proof?

4. Prove $\|G\boldsymbol{x}\|$ $(\boldsymbol{x} \in \mathbb{R}^d)$ is a valid vector norm when $G \in \mathbb{R}^d \times \mathbb{R}^d$ is a positive definite matrix.

5. (Ridge regression) In this problem, we pay attention to linear regression models. The model produced by Equation 34 is called the *ordinary least squares* model. This solution method, however, is problematic when noise exists in the data or the labels, as shown by the example below.

   We use the Matlab / Octave command

   ```
   x=-7:1:7;
   ```

   to generate 15 1-dimensional examples, and the linear regression model is

   $$y = 0.3x + 0.2\,.$$

   In order to handle the bias term (0.2), we use the command

   ```
   xb = [x; ones(size(x))];
   ```

   to transform the input x to two-dimensions, where the second dimension is constant and will handle the bias term. We suppose the labels are corrupted by white noise, as

   ```
   rng(0) % make it repeatable                    (59)
   noise = randn(size(y))*0.2;                    (60)
   z = y + noise;                                 (61)
   ```

   (a) We want to use `x` (or `xb`) and `z` to estimate a linear model such that $z = wx + b$, in which true values are $w = 0.3$ and $b = 0.2$. Write a program to find the ordinary least square estimation for w and b. Are there errors in these estimations? Are the estimations for w and b equally accurate? What causes the errors in the OLS estimation?

   (b) The loss function in ordinary linear regression is shown in Equation 30, as

   $$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(\boldsymbol{x}_i^T \boldsymbol{\beta} - y_i\right)^2 ,$$

   in which $\boldsymbol{x}_i$ and y_i are the i-th training features and label, respectively. An alternative linear regression method, *ridge regression*, minimizes the following objective

   $$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(\boldsymbol{x}_i^T \boldsymbol{\beta} - y_i\right)^2 + \lambda \|\boldsymbol{\beta}\|^2 , \tag{62}$$

   in which $\lambda > 0$ is a hyperparameter. The regularization term $\lambda\|\boldsymbol{\beta}\|^2$ will change the solution of OLS. What is the effect of this regularization term?

   (c) Now try ridge regression with $\lambda = 9.3$, what are the estimates in this setup? Are these estimates better or worse than OLS ones?

   (d) Try different λ values, and fill the ridge regression estimates into Table 1. What have you learned from this table?

Table 1: Ridge regression under different λ values.

λ	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
w					
b					

6. In this problem, we will use the LIBLINEAR software and try one particular data transformation.

   (a) Download the LIBLINEAR software (https://www.csie.ntu.edu.tw/~cjlin/liblinear/) and learn how to use it. You can also use the Matlab / Octave binding and use it along with Matlab / Octave.

   (b) Download the MNIST dataset from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#mnist. Use the non-scaled version, which includes a training and a test set. Using the default parameters of LIBLINEAR, what is the accuracy?

   (c) For every feature value (including those for both training and test examples), perform the following data transformation:

$$x \leftarrow \sqrt{x}\,.$$

   What is the new accuracy rate after this transformation?

   (d) Why the square root transformation changes the accuracy this way?

7. (Sigmoid) The logistic sigmoid function has been widely used in machine learning and pattern recognition, especially in the neural networks community. Let

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

   denote the logistic sigmoid function. We study this function in this problem.

   3 (a) Prove that $1 - \sigma(x) = \sigma(-x)$.

   3 (b) Prove that $\sigma'(x) = \sigma(x)(1-\sigma(x))$, in which $\sigma'(x)$ means $\frac{\mathrm{d}}{\mathrm{d}x}\sigma(x)$. Draw a figure to show the curves of $\sigma(x)$ and $\sigma'(x)$ simultaneously.

   4 (c) A neural network is often a layer-by-layer processing machine. For example, the output $\boldsymbol{y}$ for an input $\boldsymbol{x}$ can be produced as

$$\boldsymbol{y} = f^{(L)}\left(f^{(L-1)}\left(\cdots f^{(2)}\left(f^{(1)}(\boldsymbol{x})\right)\right)\right)\,,$$

   in which $f^{(i)}$ is a mathematical function describing the i-th layer of processing. When the number of processing layers, L, is large, it is called a deep neural network (cf. Chapter 15).

   Stochastic gradient descent is often the choice of optimization algorithm for neural networks. Let $\boldsymbol{\theta}$ be the current values of all parameters in the

network, and $\boldsymbol{g}$ the gradient of the loss function with respect to $\boldsymbol{\theta}$, then $\boldsymbol{\theta}$ is updated as

$$\boldsymbol{\theta}^{new} \leftarrow \boldsymbol{\theta} - \lambda \boldsymbol{g}\,, \tag{63}$$

in which λ is a positive learning rate.

The gradient $\boldsymbol{g}$ are computed using a chain rule. Let $\boldsymbol{\theta}^{(i)}$ be the parameters in the i-th layer, and $\boldsymbol{y}^{(i)}$ be the output after the first i layers. Then,

$$\frac{\partial \ell}{\partial (\boldsymbol{\theta}^{(i)})^T} = \frac{\partial \ell}{\partial (\boldsymbol{y}^{(i)})^T} \frac{\partial \boldsymbol{y}^{(i)}}{\partial (\boldsymbol{\theta}^{(i)})^T}\,, \tag{64}$$

in which ℓ is the overall loss function to be minimized. This computation is called *error back-propagation*, because the errors in ℓ propagates from the last layer towards the first layer in a backward order.

This learning strategy, however, often suffers from the *diminishing gradient* problem, which means that for some i, the gradient at this layer $\frac{\partial \ell}{\partial (\boldsymbol{\theta}^{(i)})^T}$ becomes very small, or $\left\| \frac{\partial \ell}{\partial (\boldsymbol{\theta}^{(i)})^T} \right\| \to 0$ quickly when i changes from L toward 1. The sigmoid function $\sigma(x)$ was popular in neural networks. Several layers $f^{(i)}$ apply the sigmoid function individually to every element in its input.

Show that the sigmoid function easily leads to the vanishing gradient difficulty. (Hint: You can look at only one element in the gradient. Refer to the figure you created for the last sub-problem.)