5-Distances-Timeseries

September 18, 2016

1 Distances and Timeseries

Today we will start building some tools for making comparisons of data objects, with particular attention to timeseries.

Working with data, we can encounter a wide variety of different data objects:

- Records of users
- Graphs
- Images
- Videos
- Text (webpages, books)
- Strings (DNA sequences)
- Timeseries
- ...

How can we compare them?

1.1 Feature space representation

Usually a data object consists of a set of attributes.

These are also commonly called **features.**

- ("J. Smith", 25, $ 200,000)

If all d dimensions are real-valued then we can visualize each data object as a point in a vector space.

- (25, USD 200,000) $\rightarrow \begin{bmatrix} 25 \\ 200000 \end{bmatrix}$.

Likewise If all features are binary then we can think of each data object as a binary vector in vector space.

The space is called **feature space.**

We then are naturally interested in how **similar** or **dissimilar** two objects are.

A dissimilarity function takes two objects as input, and returns a large value when then two objects are not very similar.

Often we put restrictions on the dissimilarity function.

One of the most common is that it be a **metric.**

The dissimilarity $d(x, y)$ between two objects x and y is a **metric** if

- $d(i,j) = 0 \leftrightarrow i == j$. (identity of indiscernables)
- $d(i,j) = d(j,i)$ (symmetry)
- $d(i,j) \leq d(i,h) + d(h,j)$ (triangle inequality)

A metric is also commonly called a **distance**.

Sometimes we will use "distance" informally, ie, to refer to a dissimilarity function even if we are not sure it is a metric. We'll try to say "dissimilarity" in those cases though.

Definitions of distance or dissimilarity functions are usually diferent for real, boolean, categorical, and ordinal variables.

Weights may be associated with diferent variables based on applications and data semantics.

1.2 Matrix representation

Very often we will manage data conveniently in matrix form.

The standard way of doing this is:

$$\overbrace{\quad\quad}^{n \text{ features}}$$
$$m \text{ data objects} \left\{ \begin{bmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m1} & \dots & x_{mj} & \dots & x_{mn} \end{bmatrix} \right.$$

Where we typically use symbols m for number of rows (objects) and n for number of columns (features).

When we are working with distances, the matrix representation is:

$$\overbrace{\quad\quad}^{m \text{ data objects}}$$
$$m \text{ data objects} \left\{ \begin{bmatrix} 0 & & & \\ d(1,2) & 0 & & \\ d(1,3) & d(2,3) & \ddots & \\ \vdots & \vdots & \vdots & \\ d(1,m) & d(2,m) & \dots & 0 \end{bmatrix} \right.$$

1.3 Norms

Assume some function $p(\mathbf{v})$ which measures the "size" of the vector $\mathbf{v}$.

$p()$ is called a **norm** if:

- $p(a\mathbf{v}) = |a|\, p(\mathbf{v})$ (absolute scaling)
- $p(\mathbf{u} + \mathbf{v}) \leq p(\mathbf{u}) + p(\mathbf{v})$ (subadditivity)
- $p(\mathbf{v}) = 0 \leftrightarrow \mathbf{v}$ is the zero vector (separates points)

Norms are important for this reason:

Every norm defines a corresponding metric.

That is, if $p()$ is a norm, then $d(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} - \mathbf{y})$ is a metric.

ℓ_p Norm.

Defines the *Minkowski distance.*

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

A special case of this isthe ℓ_2 norm that we've already studied, which defines the **Euclidean** norm.

It is equal to the euclidean distance between the vectors.

$$L_1(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

Another important special case is the ℓ_1 norm.

This defines the **Manhattan** distance, or (for binary vectors), the **Hamming** distance:

$$L_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1} |x_i - y_i|$$

1.4 Similarity and Dissimilarity

We've already seen that the inner product of two vectors can be used to compute the **cosine of the angle** between them:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

Note that this value is **large** when $\mathbf{x} \approx \mathbf{y}$. So it is a **similarity** function.

We often find that we have a similarity function and need to convert it to a dissimilarity function. Two straightforward ways of doing that are:

$$d(x, y) = 1/s(x, y)$$

$$d(x, y) = k - s(x, y)$$

For some properly chosen k.

For cosine similarity, one often uses:

$$d(\mathbf{x}, \mathbf{y}) = 1 - \cos(\mathbf{x}, \mathbf{y})$$

Note however that this is **not a metric!.**

However if we recover the actual angle beween $\mathbf{x}$ and $\mathbf{y}$, that is a metric.

1.5 Bit vectors and Sets

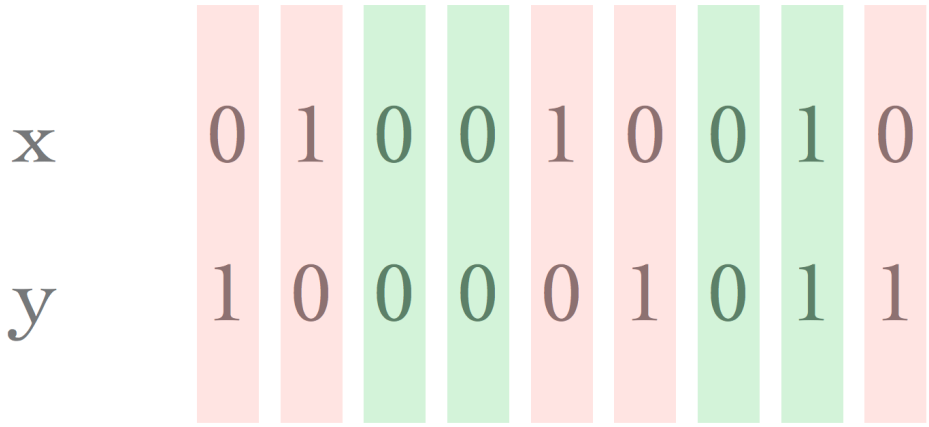
When working with bit vectors, the ℓ_1 metric is commonly used and called the **Hamming** distance.

x 0 1 0 0 1 0 0 1 0

y 1 0 0 0 0 1 0 1 1

$$L_1(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i| \right)$$

This has a natural interpretation: “how well do the two vectors match?”
Or: “What is the smallest number of bit flips that will convert one vector into the other?”

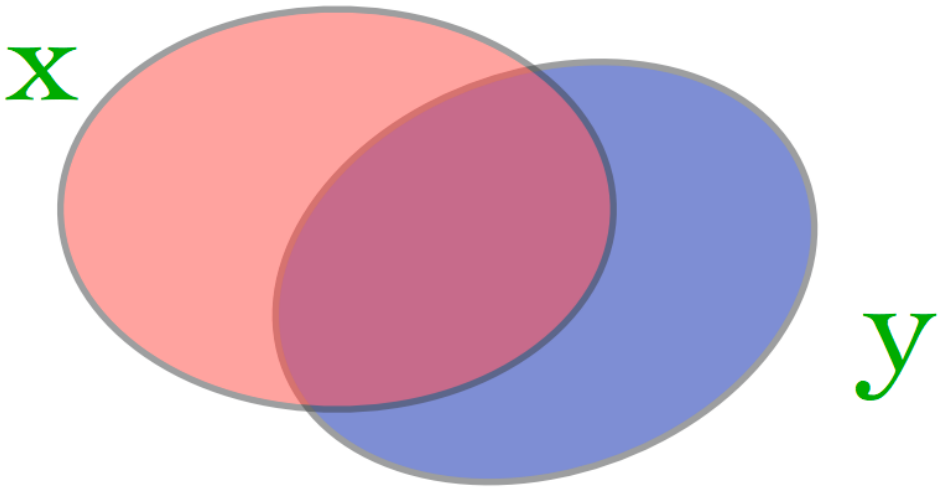


$$L_1(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i| \right)$$

In other cases, the Hamming distance is not a particular appropriate metric.
Consider the case in which the bit vector is being used to represent a set.
In that case, Hamming distance measures the **size of the set difference.**
For example, two documents. We will use bit vectors to represent the sets of words in each document.

- Case 1: both documents are large, almost identical, but differ in 10 words.
- Case 2: both documents are small, disjoint, have 5 words each.

The situation can be represented as this:



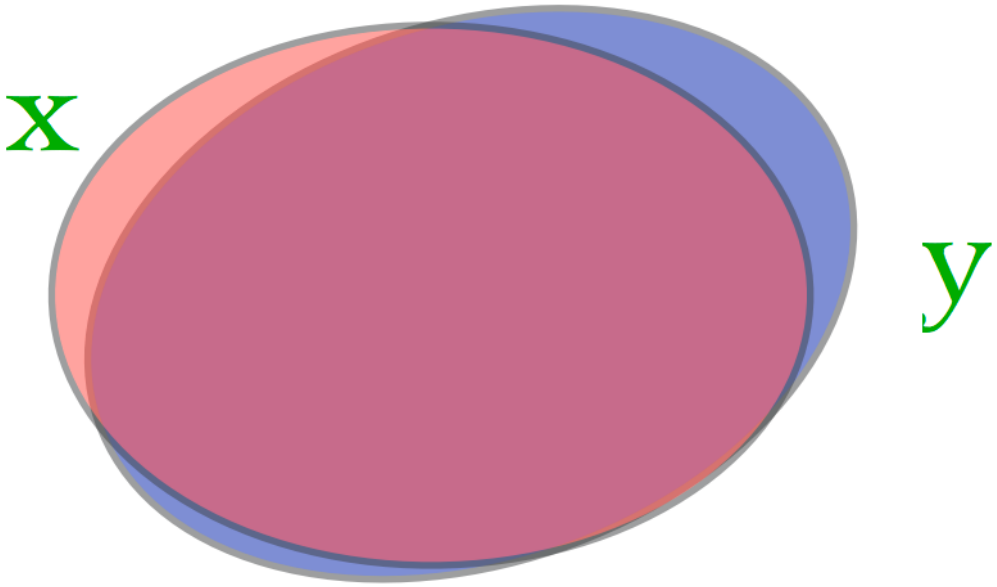
What matters is not just the size of the set difference, but the size of the intersection as well.
This leads to the *Jaccard* similarity:

$$JSim(\mathbf{x}, \mathbf{y}) = \frac{|\mathbf{x} \cap \mathbf{y}|}{|\mathbf{x} \cup \mathbf{y}|}$$

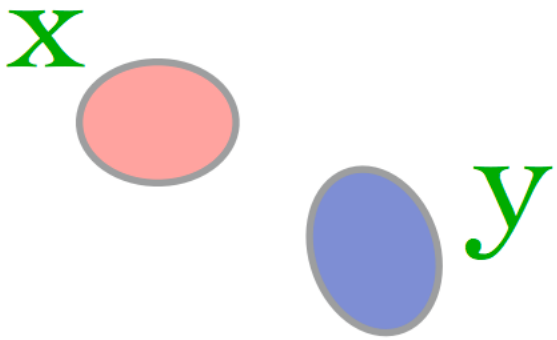
This takes on values from 0 to 1, so a natural dissimilarity metric is $1 - JSim()$.
In fact, this is a **metric!**:

$$JDist(\mathbf{x}, \mathbf{y}) = 1 - \frac{|\mathbf{x} \cap \mathbf{y}|}{|\mathbf{x} \cup \mathbf{y}|}$$

Consider our two cases:
Case 1: (very large almost identical documents)



Here $JSim(\mathbf{x}, \mathbf{y})$ is almost 1.
Case 2: (small disjoint documents)



Here $JSim(\mathbf{x}, \mathbf{y})$ is 0.