CS-E4600 — Lecture notes #3
Distance embeddings

Aristides Gionis

September 18, 2018

As we have already discussed, it is often convenient to map objects from one space to another. For example, when processing data of complex types (photos, documents, time-series) it is common to map those objects to vectors in a high-dimensional space. Another typical situation is when we want to map a vector space of high dimensionality to a vector space of lower dimensionality, so as to be able to perform more efficient computation, or visualize the data easier.

Another recent example of a mapping between different data types is the `word2vec` mapping, which was proposed recently by researchers in Google [4] and has received a fair amount of attention. In `word2vec` English words (or words from other languages) are mapped to vectors in a way that logical relations among words can be expressed by vector operations. For instance, it was shown by the authors of the original paper that the mapping can capture linguistic analogies, e.g., the vector `word2vec`('Paris')−`word2vec`('France')+`word2vec`('Italy') results in a vector that is very close to `word2vec`('Rome'). Note here that + and − are just vector operations. Similar mappings have been proposed for other applications, for example, `item2vec` for recommendation systems [2], `bioVectors` for computational biology [1], `node2vec` for graph mining [3], etc. The study of such sophisticated mappings goes beyond the scope of these lecture notes. Here we consider more abstract mappings between metric spaces, and we discuss some of the mathematical theory associated with them.

A mapping between two metric spaces is called an *embedding*. More formally, let (X, d) and (X', d') be two metric spaces. Recall that d is a *distance function* over pairs of objects in X, i.e., $d : X \times X \to \mathbb{R}$, with d being *non-negative* and *symmetric*, and satisfying the *isolation* property and the *triangle inequality*. The same properties hold for (X', d'). An embedding f is simply a mapping from objects of X to objects in X', that is,

$$f : X \to X',$$

so that an object $x \in X$ maps to an object $f(x) \in X'$.

Note that, as our goal is to allow for easier computations, we typically consider that the space X' is "simpler" than X.

We are interested in embeddings that *preserve distances* between the underlying metric spaces. Intuitively, if x and y are two objects in X, which via the embedding are mapped to objects $f(x)$ and $f(y)$ in X', we want the distance of the original objects $d(x, y)$, measured in X, to be *related* to the distance of their images $d'(f(x), f(y))$, measured in X'.

An embedding $f : X \to X'$ is called *isometric* or *distance preserving* if for all pairs of objects $x, y \in X$ it holds

$$d(x, y) = d'(f(x), f(y)).$$

An isometric embedding is the best we can hope as all distances are kept intact. In many cases, however, it is not possible to obtain such exact mappings, at least without increasing too much the complexity of the embedding function f or the dimension of the "host" space X'.

On the other hand, quite often it is sufficient to devise an embedding that maintains distances *approximately*. To quantify the degree to which an embedding approximates distances from X to X' we define the notion of *embedding distortion*. In particular, given an embedding $f : X \to X'$ between metric spaces (X, d) and (X', d') we define the *embedding contraction* c_f by

$$c_f = \max_{x,y \in X} \frac{d(x,y)}{d'(f(x), f(y))},$$

and the *embedding expansion* e_f by

$$e_f = \max_{x,y \in X} \frac{d'(f(x), f(y))}{d(x,y)}.$$

As their names suggest the contraction and expansion of an embedding capture the largest ratio that the distance of two objects in the original space may shrink or expand, respectively, in the host space, under the embedding. The *embedding distortion* α_f of f is then defined as

$$\alpha_f = c_f \cdot e_f.$$

We can also define the embedding distortion as the smallest number α_f for which it holds

$$r \cdot d(x,y) \leq d'(f(x), f(y)) \leq \alpha_f \cdot r \cdot d(x,y),$$

for some $r > 0$ and for all $x, y \in X$.

It is left as exercise to verify that the two definitions of embedding distortion are equivalent.

We next present an example of an isometric embedding.

Embedding L_1 to L_∞. We present an embedding from $(\mathbb{R}^m, L_1)$ to $(\mathbb{R}^{2^m}, L_\infty)$, that is, we embed m-dimensional vectors with L_1 distance to 2^m-dimensional vectors with L_∞ distance. Mapping to a vector space with exponentially larger dimension seems not a particularly smart thing to do, however, the counter effect is that L_∞ is a simpler norm to work with, and for certain tasks the embedding may lead to more efficient computation.

Let $\mathbf{x} = [x_1, \ldots, x_m]$ be an m-dimensional vector. The L_1 norm of $\mathbf{x}$ can be written as

$$||\mathbf{x}||_1 = \sum_i |x_i| = \sum_i \text{sgn}(x_i) x_i,$$

where the sign function sgn is defined by

$$\text{sgn}(x) = \begin{cases} +1 & \text{if } x \geq 0, \\ -1 & \text{if } x < 0. \end{cases}$$

Considering any set of numbers $y_i \in \{-1, +1\}$ gives

$$||\mathbf{x}||_1 = \sum_i \text{sgn}(x_i) x_i \geq \sum_i y_i x_i,$$

where the inequality becomes tight when $y_i = \text{sgn}(x_i)$, for all $i = 1, \ldots, m$.

Let $\mathscr{Y}$ be the set of all m-dimensional vectors with coordinates -1 or $+1$; so the set $\mathscr{Y}$ has 2^m vectors. From the previous discussion it follows that

$$||\mathbf{x}||_1 = \max_{\mathbf{y}\in\mathscr{Y}} \{\mathbf{y}\cdot\mathbf{x}\}. \tag{1}$$

We are now ready define the embedding $f : \mathbb{R}^m \to \mathbb{R}^{2^m}$. For a vector $\mathbf{x} \in \mathbb{R}^m$ we define

$$f(\mathbf{x}) = [\mathbf{y}_1 \cdot \mathbf{x}, \ldots, \mathbf{y}_{2^m} \cdot \mathbf{x}],$$

where $\mathbf{y}_1, \ldots, \mathbf{y}_{2^m}$ are all the 2^m vectors of $\mathscr{Y}$. In other words, each vector $\mathbf{x} \in \mathbb{R}^m$ is mapped to the 2^m-dimensional space by taking as the i-th coordinate the inner product of $\mathbf{x}$ with the i-th vector $\mathbf{y}_i$ of $\mathscr{Y}$.

We show that the embedding f is distance preserving. For any two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$ we have

$$\begin{aligned}
||f(\mathbf{u}) - f(\mathbf{v})||_\infty &= ||[\mathbf{y}_1 \cdot \mathbf{u}, \ldots, \mathbf{y}_{2^m} \cdot \mathbf{u}] - [\mathbf{y}_1 \cdot \mathbf{v}, \ldots, \mathbf{y}_{2^m} \cdot \mathbf{v}]||_\infty \\
&= ||[\mathbf{y}_1 \cdot (\mathbf{u}-\mathbf{v}), \ldots, \mathbf{y}_{2^m} \cdot (\mathbf{u}-\mathbf{v})]||_\infty \\
&= ||f(\mathbf{u}-\mathbf{v})||_\infty \\
&= \max_{\mathbf{y}\in\mathscr{Y}} \{\mathbf{y}\cdot(\mathbf{u}-\mathbf{v})\} \\
&= ||\mathbf{u}-\mathbf{v}||_1,
\end{aligned}$$

where the penultimate equality follows from the definition of f and the definition of L_∞, while the last equality follows from Equation (1).

The fact that $||\mathbf{u}-\mathbf{v}||_1 = ||f(\mathbf{u}) - f(\mathbf{v})||_\infty$ establishes that f is a distance-preserving embedding of $(\mathbb{R}^m, L_1)$ to $(\mathbb{R}^{2^m}, L_\infty)$.

Next we provide an application of the L_1 to L_∞ embedding. Consider a dataset $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n vectors in $\mathbb{R}^m$. Assume that we want to compute the pair of furthest vectors in X, according to L_1 distance. Why do we want to do that? Perhaps we are working with an application where our vectors represent customers and we want to find the most dissimilar customers as a way to obtain an understanding of our customer base, thinking that observing two extreme cases will help us understand everything "in between." Alternatively, as we will see in one of the next lectures, finding furthest points is a primitive used in clustering algorithms.

Now, a simple algorithm to find the pair of furthest vectors in X is to evaluate the L_1 distance between all pairs of vectors and report the pair with the largest distance. There are $n(n-1)/2$ pairs of vectors, and an L_1 distance operation requires $\mathcal{O}(m)$ operations, so the running time of this simple algorithm is $\mathcal{O}(n^2 m)$.

Assume now that we map the n vectors of X from $\mathbb{R}^m$ to $\mathbb{R}^{2^m}$, using the embedding f discussed above. Interestingly, given n vectors in $\mathbb{R}^d$, the pair of furthest vectors according to the L_∞ distance can be computed in time $\mathcal{O}(nd)$.[1] It follows that for our application, the pair of furthest vectors can be found in time $\mathcal{O}(n2^m)$.

It follows that if m is small compared to n, i.e., if $m = o(\log n)$, solving the furthest-pair problem in the embedded space $(\mathbb{R}^{2^m}, L_\infty)$ is more efficient than solving it in original space $(\mathbb{R}^m, L_1)$.

[1] The proof of this claim is left as exercise.

Exercises

1. Consider a finite metric space (X, d), with $|X| = n$. Show that (X, d) can be isometrically embedded to $(\mathbb{R}^{n-1}, L_\infty)$.

2. Let (X, d) be a finite metric space, with $|X| = n$. We say that (X, d) is a *tree metric* if it can be isometrically embedded into a tree T. In other words, for a tree metric we assume that there exists a tree T, possibly with weighted edges, and a mapping $f : X \to T$, so that each element $x \in X$ can be mapped to a node $f(x) \in T$, and for each pair of elements $x, y \in X$ the distance $d(x, y)$ is equal to the length of the unique path on T that connects $f(x)$ with $f(y)$ (where the length of a path is the sum of the weights of all edges along this path).

Show that if (X, d) is a tree metric, then it can be isometrically embedded into $(\mathbb{R}^{\log n}, L_\infty)$.

References

[1] E. Asgari and M. R. Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11), 2015.

[2] O. Barkan and N. Koenigstein. Item2vec: Neural item embedding for collaborative filtering. *CoRR*, abs/1603.04259, 2016.

[3] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.