Summary: Dimensionality Reduction and Visualization of Representations

Younesse Kaddar

Studied articles: Visualizing MNIST, Visualizing Representations (C. Olah), Visualizing Data using t-SNE (L. Van der Maaten & G. Hinton) + added Isomap and LLE to the dimensionality reduction overview

Dimensionality reduction

The bottom line of dimensionality reduction (DR) is: how to turn *high-dimensional data* into *lower-dimensional data*, so that we can *vizualize* it more easily? The usefulness of DR relies on the *manifold hypothesis*, according to which real-world high-dimensional data vectors lie in a lower-dimensional embedded manifold. First, we'll make an overview of a handful of DR methods. Assume that we are given n data points $x_1, \cdots, x_n$.

Multidimensional Scaling (MDS)

Goal: Preserve the distances between points in the data space, so as to conserve the geometry of the data. We denote by d^* (resp. d) the distance in the original space (resp. our visualization). Then we resort to *gradient descent* to minimize the cost function $\sum_{x_i \neq x_j \text{ data points}} \Big(d^*(x_i, x_j) - d(x_i, x_j)\Big)^2$

Sammon's mapping

It is a tweak of MDS to ensure that the smaller the distance between two data points is in the original data space, the more it is preserved. The cost function to be minimized is $\sum_{x_i \neq x_j \text{ data points}} \left(\frac{d^*(x_i,x_j) - d(x_i,x_j)}{d^*(x_i,x_j)}\right)^2$

Force-directed Graph Drawing

Goal: Use physical intuition! Build the graph whose the vertices are the data points, where each point is connected to its k nearest neighbors in original space. The vertices are then seen as repelling charged particles, and the edges as springs. As with electric potential energy and spring energy, we minimize the (potential) energy function $\sum_{x_i \neq x_j \text{ data points}} \frac{1}{d^*(x_i,x_j)} + \sum_{x_i, x_j \text{ data points}} \frac{1}{2}\Big(d^*(x_i, x_j) - d(x_i, x_j)\Big)^2$

Principal component analysis (PCA)

Goal: Find a few orthogonal vectors/axes onto which the variance of the data points under projection is maximal, i.e. find the best possible "angles" from which the data points are the most spread out.

Isomap

Goal: Similar to MDS, except that we take into account the *curvature* of the data space: the distance used is no longer the euclidean one but the *geodesic* one, where the length of paths on curved manifold surfaces are measured as if the surfaces were flat.

Locally-Linear Embedding (LLE)

Goal: Preserve the relationship between neighboring points.

1. Find the weight matrix $(W_{i,j})_{i,j}$ - whose rows sum to 1 - that minimizes the cost function:
   $$\sum_{x_i \text{ data point}} \left| x_i - \sum_{x_j \text{ neighbor of } x_i} W_{i,j} x_j \right|^2$$
2. Map each data point x_i to a point y_i in the vizualization, such that the y_k's minimize the cost function
   $$\sum_{y_i \text{ data point}} \left| y_i - \sum_{y_j \text{ neighbor of } y_i} W_{i,j} y_j \right|^2$$

t-Distributed Stochastic Neighbor Embedding (t-SNE)

This very popular technique is similar to the graph drawing one, except that the relation "being neighbors" is here turned into a "continuous range of neighborness".

1. Compute conditional probabilities $p_{j|i}$ that x_i has x_j as its neighbor if neighbors were chosen according to a Gaussian distribution centered at x_i
2. Then, we define the joint probabilities $p_{i,j}$ as the symmetrized conditional probabilities $\frac{p_{j|i}+p_{i|j}}{2n}$
3. Similarities $q_{i,j}$ between two points in the visualization are computed with resort to a Cauchy distribution. This distribution approaches an inverse square law for large distances in the visualization, which leads to (almost) invariance to changes of scale, for points that are far apart.
4. Modify the y_i's (visualization points) with gradient descent to minimize the Kullback–Leibler divergence: $\sum_{i \neq j} p_{ij} \log \frac{p_{i,j}}{q_{i,j}}$, while recomputing the $q_{i,j}$'s at each step, up to the desired level of precision.

Dimensionality reduction to visualize high-dimensional representations

In a neural network, the input data has its shape changed from a layer to another: a *representation* is the reshaped data at a given layer. Since representations are high-dimensional, we can use DR methods to visualize them: it can prove useful to grasp the inner workings of neural networks, and better understand the data itself.

meta-SNE to visualize the space of representations

By building their matrices of pairwise distances, we vectorize visualized representations (note that representations that are equal up to isometries (rotations, switching of dimensions, …) give rise to isometric matrices). Then, taking a step up the ladder of abstraction, we can visualize our vectorized representations with t-SNE: this process is called *meta-SNE*.

> *Regarding neural networks, meta-SNE enables us no longer to confine ourselves to comparing their outcome only, but to take a step toward comparing how they operate internally.*