Artificial Intelligence - Knowledge Representation and Planning - Assignment 3

Lorenzo Soligo - 875566

Academic Year 2018-2019

1 Requirements

Read this article presenting a way to improve the disciminative power of graph kernels. Choose one graph kernel among

- Shortest-path Kernel
- Graphlet Kernel
- Random Walk Kernel
- Weisfeiler-Lehman Kernel

Choose one manifold learning technique among

- Isomap
- Diffusion Maps
- Laplacian Eigenmaps
- Local Linear Embedding

Compare the performance of an SVM trained on the given kernel, with or without the manifold learning step, on the following datasets:

- PPI: this is a Protein-Protein Interaction dataset. Here proteins (nodes) are connected by an edge in the graph if in actuality they are so physically or functionally.
- Shock: representing 2D shapes.

Note: the datasets are contained in Matlab files. The variable G contains a vector of cells, one per graph. The entry am of each cell is the adjacency matrix of the graph. The variable labels, contains the class-labels of each graph.

2 Background

2.1 Kernel functions

A positive-definite kernel is a generalization of a positive-definite function or a positive-definite matrix.

Let $\mathcal{X}$ be a nonempty set. A *symmetric* function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive-definite kernel on $\mathcal{X}$ if

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0 \qquad \forall n \in \mathbb{N}, \quad \forall x_1, ..., x_n \in \mathcal{X}, \quad \forall c_1, ..., c_n \in \mathbb{R}$$

Notice that positive definite kernels require $c_i = 0 \ \forall i$, while positive semi-definite kernels do not impose this condition. This relates to the spectrum of a finite matrix constructed by pairwise evaluation $\mathbf{K}_{ij} = K(x_i, x_j)$: in the former case we have only positive eigenvalues; in the latter we have non-negative eigenvalues.

2.1.1 The kernel trick

Kernel methods (namely SVMs and many more) exploit kernel functions to work on high-dimensional, implicit feature spaces without having to compute the coordinates of the data in that space. This is achieved by performing inner products between the images of all pairs of data in the feature space. This operation is called *kernel trick*. It is extremely useful in the case the dataset is not linearly separable, but can be easily separated by an hyperplane in a higher-dimensional space.

Formally, a kernel maps two objects x and x' via a mapping ϕ into the feature space $\mathcal{H}$, measuring their similarity in $\mathcal{H}$ as $\langle \phi(x), \phi(x') \rangle$. The kernel trick is nothing but computing the inner product in $\mathcal{H}$ as kernel in the input space: $k(x, x') = \langle \phi(x), \phi(x') \rangle$.

2.2 Graph Comparison Problem

Given two graphs G and G' from the space of graphs $\mathcal{G}$, the problem of graph comparison is to find a mapping

$$s : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$$

such that $s(G, G')$ quantifies the similarity (or dissimilarity) of G and G'.

2.3 Graph isomorphism

Given two graphs G_1 and G_2, find a mapping f of the vertices of G_1 to the vertices of G_2 such that G_1 and G_2 are identical, i.e. (x, y) is an edge of G_1 iff $(f(x), f(y))$ is an edge of G_2. Then f is an isomorphism, and G_1 and G_2 are said to be isomorphic.
At the moment we do not know a polynomial time algorithm for graph isomorphism, but we also do not know whether the problem is NP-complete.
On the other hand, we know that subgraph isomorphism is NP-complete. Subgraph isomorphism checks whether there is a subset of edges and vertices of G_1 that is isomorphic to a smaller graph G_2.

2.3.1 Graph edit distances

The idea is to count the number of operations that is necessary to transform G_1 into G_2, assigning different costs to the different types of operations (e.g. edge/node insertion/deletion, modification of labels, ...). This allows us to capture (partial) similarities between graphs, but contains a check for subgraph isomorphism (which is NP-complete) as an intermediate step.

2.3.2 Topological descriptors

The idea here is to map each graph to a feature vector and then using distances and metrics on vectors for learning on graphs. In this case the clear advantage is that known, efficient tools for feature vectors can be reused, but the feature vector transformation either leads to a loss of topological information or still includes subgraph isomorphism as one step.

3 Manifold Learning

3.1 Introduction and motivations

Manifold learning is an approach to non-linear dimensionality reduction. Algorithms for this task are based on the idea that the dimensionality of many data sets is only artificially high.

High-dimensional datasets can be very difficult to visualize. In order to visualize the structure of a dataset, the dimension must be reduced in some way.

The simplest way to accomplish this dimensionality reduction is by taking a random projection of the data. Though this allows some degree of visualization of the data structure, the randomness of the choice leaves much to be desired. In a random projection, it is likely that the more interesting structure within the data will be lost.

To address this concern, a number of supervised and unsupervised linear dimensionality reduction frameworks have been designed, such as Principal Component Analysis (PCA) and many others. These algorithms define specific rubrics to choose an "interesting" linear projection of the data. These methods can be powerful, but often miss important non-linear structure in the data.

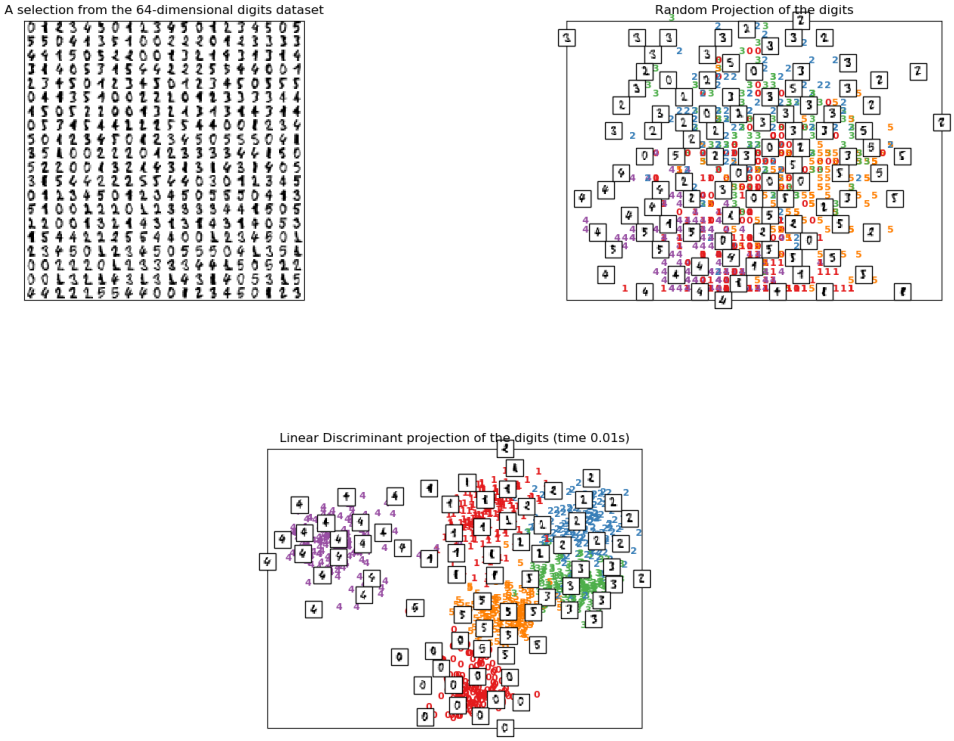

Figure 1: The representation drastically improves using dimensionality reduction techniques

3.2 Defining manifold learning

Manifold Learning can be thought of as an attempt to generalize linear frameworks like PCA to be sensitive to non-linear structure in data. Though supervised variants exist, the typical manifold learning problem is unsupervised: it learns the high-dimensional structure of the data from the data itself, without the use of predetermined classifications. Intuitively, the "curvier" is the considered manifold, the denser the data must be.

3.3 Isomap

3.3.1 Description

Isomap (short for ISOmetric MAPping) seeks a low-dimensional representation of the data which maintains geodesic (namely, the shortest path between two points on a surface/manifold) distances between all points. In Isomap, the long-range distances become more important than the local structure, and this makes it quite sensitive to noise. Isomap is computational expensive due to the heavy work on matrices to be done.

3.3.2 Analysis

Isomap consists of three stages:

1. Nearest-neighbor search
2. Shortest-path graph search
3. Partial eigenvalue decomposition

3.4 Locally Linear Embedding (LLE)

Locally linear embedding (LLE) seeks a lower-dimensional projection of the data which preserves distances within local neighborhoods. It can be thought of as a series of local Principal Component Analyses which are globally compared to find the best non-linear embedding. LLE is faster than Isomap and consists in simple linear algebra operations. Since it focuses on preserving distances locally, it can distort the global structure of the data.

3.4.1 Analysis

The standard LLE algorithm consists of three stages:

1. Nearest-neighbor search
2. Weight matrix construction
3. Partial eigenvalue decomposition

3.4.2 Modified Locally Linear Embeddng

One well-known issue with LLE is the regularization problem. When the number of neighbors is greater than the number of input dimensions, the matrix defining each local neighborhood is rank-deficient. Standard LLE addresses this problem by applying an arbitrary regularization parameter r, which is chosen relative to the trace of the local weight matrix. It can be proved that for $r \to 0$ the solution converges to the desired embedding, but there is no guarantee that the optimal solution will be found for $r > 0$. This results in a distortion of the underlying geometry of the manifold.

One method to address the regularization problem is to use multiple weight vectors in each neighborhood. This is the essence of modified locally linear embedding (MLLE).

The steps taken are the same as standard LLE, but the weight matrix construction takes more time because we need to construct the weight matrix from multiple weights. In practice, however, this increase in the cost is negligible with respect to the cost of steps 1 and 3.

4 Graph kernels

4.1 Moving towards polynomial times

From the background, we have understood that computing whether two graphs are isomorphic is usually expensive, often becoming infeasible for “big” graphs. Therefore it would be grat to have a polynomial time similarity measure for graphs. Graph kernels allow us to compare substructures of graphs that are computable in polynomial time. We want a graph kernel to be expressive, efficient to compute, positive definite and applicable to a wide range of graphs.

4.2 Kernel 1

4.3 Kernel 2

5 Experiments

6 Conclusions

7 Resources

- Professor's slides
- http://www.dsi.unive.it/~atorsell/AI/graph/Unfolding.pdf
- http://www.dsi.unive.it/~atorsell/AI/graph/kernels.pdf
- https://en.wikipedia.org/wiki/Kernel_method
- https://en.wikipedia.org/wiki/Positive-definite_kernel
- https://www.ethz.ch/content/dam/ethz/special-interest/bsse/borgwardt-lab/documents/slides/CA10_GraphKer
- https://scikit-learn.org/stable/modules/manifold.html
- https://scikit-learn.org/stable/auto_examples/manifold/plot_lle_digits.html