Stat260/CS294: Spectral Graph Methods **Lecture 14 - 03/10/2015**

Lecture: Some Practical Considerations (3 of 4)

Lecturer: Michael Mahoney *Scribe: Michael Mahoney*

Warning: these notes are still very rough. They provide more details on what we discussed in class, but there may still be some errors, incomplete/imprecise statements, etc. in them.

14 Non-linear dimension reduction methods

Today, we will describe several related methods to identify structure in data. The general idea is to do some sort of "dimensionality reduction" that is more general than just linear structure that is identified by a straightforward application of the SVD or truncated SVD to the input data. The connection with what we have been discussing is that these procedures construct graphs from the data, and then they perform eigenanalysis on those graphs in order to construct "low-dimensional embeddings" of the data. These (and many other related) methods are often called *non-linear dimension reduction methods*. In some special cases they identify structure that is meaningfully non-linear; but it is best to think of them as constructing graphs to then construct data-dependent representations of the data that (like other kernel methods) is linear in some nonlinearly transformed version of the data.

14.1 Some general comments

The general framework for these methods is the following.

- Derive some (typically sparse) graph from the data, *e.g.*, by connecting nearby data points with an ϵ-NN or k-NN rule.
- Derive a matrix from the graph (viz. adjacency matrix, Laplacian matrix).
- Derive an embedding of the data into $\mathbb{R}^d$ using the eigenvectors of that matrix.

Many algorithms fit this general outline. Here are a few things worth noting about them.

- They are not really algorithms (like we have been discussion) in the sense that there is a well-defined objective function that one is trying to optimize (exactly or approximately) and for which one is trying to prove running time or quality-of-approximation bounds.
- There exists theorems that say when each of these method "works," but those theoretical results have assumptions that tend to be rather strong and/or unrealistic.
- Typically one has some sort of intuition and one shows that the algorithm works on some data in certain specialized cases. It is often hard to generalize beyond those special cases, and so it is probably best to think of these as "exploratory data analysis" tools that construct data-dependent kernels.

- The intuition underlying these methods is often that the data "live" on a low-dimensional manifold. Manifolds are very general structures; but in this context, it is best to think of them as being "curved" low-dimensional spaces. The idealized story is that the processes generating the data have only a few degrees of freedom, that might not correspond to a linear subspace, and we want to reconstruct or find that low-dimensional manifold.
- The procedures used in constructing these data-driven kernels depend on relatively-simple algorithmic primitives: shortest path computations; least-squares approximation; SDP optimization; and eigenvalue decomposition. Since these primitives are relatively simple and well-understood and since they can be run relatively quickly, non-linear dimensionality reduction methods that use them are often used to explore the data.
- This approach is often of interest in semi-supervised learning, where there is a lot of unlabeled data but very little labeled data, and where we want to use the unlabeled data to construct some sort of "prior" to help with predictions.

There are a large number of these and they have been reviewed elsewhere; here we will only review those that will be related to the algorithmic and statistical problems we will return to later.

14.2 ISOMAP

ISOMAP takes as input vectors $x_i \in \mathbb{R}^D, i = 1, \ldots, n$, and it gives as output vectors $y_i \in \mathbb{R}^d$, where $d \ll D$. The stated goal/desire is to make near (resp. far) points stay close (resp. far); and the idea to achieve this is to preserve geodesic distance along a submanifold. The algorithm uses geodesic distances in an MDS computation:

1. **Step 1.** Build the nearest neighbor graph, using k-NN or ϵ-NN. The choice here is a bit of an art. Typically, one wants to preserve properties such as that the data are connected and/or that they are thought of as being a discretization of a submanifold. Note that the k-NN here scales as $O(n^2 D)$.
2. **Step 2.** Look at the shortest path or geodesic distance between all pairs of points. That is, compute geodesics. Dijkstra's algorithm for shortest paths runs in $O(n^2 \log n + n^2 k)$ time.
3. **Step 3.** Do Metric Multi-Dimensional scaling (MDS) based on A, the shortest path distance matrix. The top k eigenvectors of the Gram matrix then give the embedding. They can be computed in $\approx O(n^d)$ time.

Advantages

- Runs in polynomial time.
- There are no local minima.
- It is non-iterative.
- It can be used in an "exploratory" manner.

Disadvantages

- Very sensitive to the choice of ϵ and k, which is an art that is coupled in nontrivial ways with data pre-processing.
- No immediate "out of sample extension"' since so there is not obvious geometry to a graph, unless it is assumed about the original data.
- Super-linear running time—computation with all the data points can be expensive, if the number of data points is large. A solution is to choose "landmark points," but for this to work one needs to have already sampled at very high sampling density, which is often not realistic.

These strengths and weaknesses are not peculiar to ISOMAP; they are typical of other graph-based spectral dimensionality-reduction methods (those we turn to next as well as most others).

14.3 Local Linear Embedding (LLE)

For LLE, the input is vectors $x_i \in \mathbb{R}^D, i = 1, \ldots, n$; and the output is vectors $y_i \in \mathbb{R}^d$, with $d \ll D$.

Algorithm

Step 1 : Construct the Adjacency Graph There are two common variations:

1. ϵ neighborhood
2. K Nearest neighbor Graph

Basically, this involves doing some sort of NN search; the metric of closeness or similarity used is based on prior knowledge; and the usual (implicit or explicit) working assumption is that the neighborhood in the graph $\approx$ the neighborhood of the underlying hypothesized manifold, at least in a "local linear" sense.

Step 2 : Choosing weights That is, construct the graph. Weights W_{ij} must be chosen for all edges ij. The idea is that each input point and its k-NN can be viewed as samples from a small approximately linear patch on a low-dimensional submanifold and we can choose weights W_{ij} to get small reconstruction error. That is, weights are chosen based on the projection of each data point on the linear subspace generated by its neighbors.

$$\begin{aligned} \min \textstyle\sum_i ||x_i - \sum_j W_{ij} x_j||_2^2 \quad &= \Phi(W) \\ \text{s.t. } W_{ij} = 0 \text{ if } (ij) \notin E & \\ \textstyle\sum_j W_{ij} = 1, \forall i & \end{aligned}$$

Step 3 : Mapping to Embedded Co-ordinates Compute output $y \in \mathbb{R}^d$ by solving the same equations, but now with the y_i as variables. That is, let

$$\Psi(y) = \sum_i ||y_i - \sum_j W_{ij} y_j||^2,$$

for a fixed W, and then we want to solve

$$\begin{aligned} &\min \Psi(y) \\ &\text{s.t. } \textstyle\sum_i y_i = 0 \\ &\frac{1}{N} \textstyle\sum_i y_i y_i^T = I \end{aligned}$$

To solve this minimization problem reduces to finding eigenvectors corresponding to the $d+1$ lowest eigenvalues of the the positive definite matrix $(I-W)'(I-W) = \Psi$.

Of course, the lowest eigenvalue is uninteresting for other reasons, so it is typically not included. Since we are really computing an embedding, we could keep it if we wanted, but it would not be useful (it would assign uniform values to every node or values proportional to the degree to every node) to do downstream tasks people want to do.

14.4 Laplacian Eigenmaps (LE)

For LE, (which we will see has some similarities with LE), the input is vectors $x_i \in \mathbb{R}^D, i = 1, \ldots, n$; and the output is vectors $y_i \in \mathbb{R}^d$, with $d \ll D$. The idea is to compute a low-dimensional representation that preserves proximity relations (basically, a quadratic penalty on nearby points), mapping nearby points to nearby points, where "nearness" is encoded in G.

Algorithm

Step 1 : Construct the Adjacency Graph Again, there are two common variations:

1. ϵ-neighborhood
2. k-Nearest Neighbor Graph

Step 2 : Choosing weights We use the following rule to assign weights to neighbors:

$$W_{ij} = \begin{cases} e^{-\frac{||x_i - x_j||^2}{4t}} & \text{if vertices i \& j are connected by an edge} \\ 0 & \text{otherwise} \end{cases}$$

Alternatively, we could simply set $W_{ij} = 1$ for vertices i and j that are connected by an edge—it should be obvious that this gives similar results as the above rule under appropriate limits, basically since the exponential decay introduces a scale that is a "soft" version of this "hard" 0-1 rule. As a practical matter, there is usually a fair amount of "cooking" to get things to work, and this is one of the knobs to turn to cook things.

Step 3 : Eigenmaps Let

$$\Psi(y) = \sum_{i,j} \frac{w_{ij}||y_i - y_j||^2}{\sqrt{D_{ii} \cdot D_{jj}}}$$

where $D = \text{Diag}\{\sum_i w_{ij} : j = 1(1)n\}$, and we compute $y \in \mathbb{R}^d$ such that

$$\begin{gathered} \min \Psi(y) \\ \text{s.t. } \textstyle\sum_i y_i = 0 \quad \text{centered} \\ \text{and } \frac{1}{N}\textstyle\sum_i y_i y_i^T = I \quad \text{unit covariance} \end{gathered}$$

that is, s.t. that is minimized for each connected component of the graph. This can be computed from the bottom $d+1$ eigenvectors of $\mathcal{L} = I - D^{-1/2}WD^{-1/2}$, after dropping the bottom eigenvector (for the reason mentioned above).

LE has close connections to analysis on manifold, and understanding it will shed light on when it is appropriate to use and what its limitations are.

- Laplacian in $\mathbb{R}^d$: $\Delta f = -\sum_i \frac{a\partial^2 f}{\partial x_i^2}$
- Manifold Laplacian: change is measured along tangent space of the manifold.

The weighted graph $\approx$ discretized representation of the manifold. There are a number of analogies, *e.g.*, Stokes Theorem (which classically is a statement about the integration of differential forms which generalizes popular theorems from vector calculus about integrating over a boundary versus integration inside the region, and which thus generalizes the fundamental theorem of calculus):

- Manifold: $\int_M ||\nabla f||^2 = \int_M f\Delta f$
- Graph: $\sum_{ij}(f_i - f_j)^2 = f^T L f$

An extension of LE to so-called Diffusion Maps (which we will get to next time) will provide additional insight on these connections.

Note that the Laplacian is like a derivative, and so minimizing it will be something like minimizing the norm of a derivative.

14.5 Interpretation as data-dependent kernels

As we mentioned, these procedures can be viewed as constructing data-dependent kernels. There are a number of technical issues, mostly having to do with the discrete-to-continuous transition, that we won't get into in this brief discussion. This perspective provides light on why they work, when they work, and when they might not be expected to work.

- ISOMAP. Recall that for MDS, we have $\delta_{ij}^2 = (x_i - x_j)^T(x_i - x_j) =$ dissimilarities. Then A is a matrix s.t. $A_{ij} = -\delta_{ij}$, and $B = HAH$, with $H = I_n - \frac{1}{n}11^T$, a "centering" or "projection" matrix. So, if the kernel $K(x_i, x_j)$ is stationary, i.e., if it is a function of $||x_i - x_j||^2 = \delta_{ij}^2$, as it is in the above construction, then $K(x_i, x_j) = r(\delta_{ij})$, for some r that scales s.t. $r(0) = 1$. Then $\tilde{\delta}_{ij}^2$ is the Euclidean distance in feature space, and if A s.t. $A_{ij} = r(\delta_{ij}) - 1$, then $A = K - 11^T$. The "centering matrix" H annihilates 11^T, so $HAH = HKH$. (See the Williams paper.) So,

  $$K_{ISOMAP} = HAH = H\mathcal{D}H = (I - 11^T)\mathcal{D}(I - 11^T)$$

  where $\mathcal{D}$ is the squared geodesic distance matrix.

- LE. Recall that LE minimizes $\psi^T L\psi = \frac{1}{2}\sum_{ij}(psi_i - \psi_j)^2 W_{ij}$, and doing this involved computing eigenvectors of L or $\mathcal{L}$, depending on the construction. The point is that L has close connections to diffusions on a graph—think of it in terms of a continuous time Markov chain:

  $$\frac{\partial\psi(t)}{\partial t} = -L\psi(t).$$

  The solution is a Green's function or heat kernel, related to the matrix exponential of L:

  $$K_t = \exp(-Lt) = \sum_\xi \phi_\xi \phi_\xi^T e^{-\lambda_\xi t},$$

where ϕ_ξ, λ_ξ are the eigenvectors/eigenvalues of L. Then, $\psi(t) = K_t\psi(0)$ is the general solution, and under assumptions one can show that

$$K_L = \frac{1}{2}L^+$$

is related to the "commute time" distance of diffusion:

$$C_{ij} \sim L^+_{ii} + L^+_{jj} - L^+_{ij} = L^+_{ji}.$$

For the difference between the commute time in a continuous time Markov chain and the geodesic distance on the graph, think of the dumbbell example; we will get to this in more detail next time.

- LLE. Recall that this says that we are approximating each point as a linear combination of neighbors. Let $(W_n)_{ij}, i, j \in [n]$ be the weight of a point x_j in the expansion of x_i; then one can show that
$$K_n(x_i, x_j) = \big((I - W_n)^T(I - W_n)\big)_{ij}$$
is PD on the domain $\mathcal{X} : x_i, \ldots, x_n$. Also, one can show that if λ is the largest eigenvalue of $(I - W)^T(I - W) = M$, then
$$K_{LLE} = \big((\lambda - 1)I + W^T + W - W^TW\big)$$
is a PSD matrix, and thus a kernel. Note also that, under assumptions, we can view $M = (I - W^T(I - W)$ as $\mathcal{L}^2$.

14.6 Connection to random walks on the graph: more on LE and diffusions

Recall that, give a data set consisting of vectors, we can construct a graph $G = (V, E)$ in one of several ways. Given that graph, consider the problem of mapping G to a *line* s.t. connected points stay as close together as possible. Let $\vec{y} = (y_1, \ldots, y_n)^T$ be the map, and by a "good" map we will mean one that minimized $\sum_{ij}(y_i - y_j)^2 W_{ij}$ under appropriate constraints, *i.e.*, it will incur a big penalty if W_{ij} is large and y_i and y_j are far apart. This has important connections with diffusions and random walks that we now discuss.

We start with the following claim:

Claim 1. $\frac{1}{2}\sum_{i,j} w_{ij}(y_i - y_j)^2 = y^T L y$

Proof. Recall that W_{ij} is symmetric and that $D_{ii} = \sum_j W_{ij}$. Then:

$$\begin{aligned}\frac{1}{2}\sum_{i,j} w_{ij}(y_i - y_j)^2 &= \frac{1}{2}\sum_{i,j} w_{ij}y_i^2 + w_{ij}y_j^2 - 2w_{ij}y_iy_j \\ &= \sum_i D_{ii}y_i^2 - \sum_{ij} w_{ij}y_iy_j \\ &= y^T L y\end{aligned}$$

where $L = D - W$, W is the weight matrix and L is Laplacian, a symmetric, positive semidefinite matrix that can be thought of as an operator on functions defined on vertices of G. □

The following, which we have seen before, is a corollary:

Claim 2. *L is SPSD.*

Proof. Immediate, since $W_{ij} > 0$. □

Recall that the solution to

$$\begin{aligned} &\arg\min y^T L y \\ &\text{s.t.} y^T D y = 1, \end{aligned}$$

where D gives a measure of the importance of vertices, turns out to be given by solving a generalized eigenvector problem

$$Ly = \lambda D y,$$

i.e., computing the bottom eigenvectors of that eigenproblem. Actually, we usually solve,

$$\begin{aligned} &\arg\min y^T L y \\ &\text{s.t.} \quad y D \vec{1} = 0 \\ &\quad y^T D y = 1, \end{aligned}$$

the reason being that since $\vec{1} = (1, \ldots, 1)$ is an eigenvector with eigenvalue 0, it is typically removed. As we saw above, the condition $y^T D \vec{1} = 0$ can be interpreted as removing a "translation invariance" in y and so is removed. Alternatively, it is uninteresting if these coordinates are going to be used simple to provide an embedding for other downstream tasks. Also, the condition $y^T D y = 1$ removes an arbitrary scaling factor in the embedding, which is a more benign thing to do if we are using the coordinates as an embedding.

From the previous discussion on Cheeger's Inequality, we know that putting the graph on a line (and, in that setting, sweeping to find a good partition) can reveal when the graph doesn't "look like" a line graph. Among other things, the distances on the line may be distorted a lot, relative to the original distances in the graph (even if they are preserved "on average").

Thus, more generally, the goal is to embed a graph in $\mathbb{R}^n$ s.t. distances are "meaningful," where meaningful might mean the same as or very similar to distances in the graph. Let $G = (V, E, W)$. We know that if W is symmetric, $W = W^T$, and point-wise positive, $W(x, y) \geq 0$, then we can interpret pairwise similarities as probability mass transitions in a Markov chain on a graph. Let $d(x) = \sum_z W(x, z) =$ the degree of node x. Then let P be the $n \times n$ matrix with entries

$$P(x, w) = \frac{W(x, y)}{d(x)}$$

which is the transition probability of going from x to y in one step, which equals the first order neighborhood of the graph. Then, P^t is higher order neighborhoods of the graph; this is sometimes interpreted as $\approx$ the "intrinsic geometry" of an underlying hypothesized manifold.

If the graph is connected, as it usually is due to data preprocessing, then

$$\lim_{t \to \infty} P^t(x, y) = \phi_0(y),$$

where ϕ_0 is the unique stationary distribution

$$\phi_0 = \frac{d(x)}{\sum_z d(z)}$$

of the associated Markov chain. (Note The Markov chain is reversible, as detailed-balance is satisfied: $\phi_0(x)P_1(x,y) = \phi_0(y)P_1(y,x)$.)

Thus, graph G defines a random walk. For some node i the probability going from i to j is $P'_{ij} = \frac{w_{ij}}{d_i}$, where $d_i = \sum_j w_{ij}$. Consider if you are at node i and you are move from i in the following way:

$$\begin{cases} \text{move to a neighbor chosen u.a.r, w.p.} = \frac{1}{d_i} & \text{w.p.} \frac{1}{2} \\ \text{stay at node i} & \text{w.p.} \frac{1}{2} \end{cases}$$

Then the transition matrix $P \in \mathbb{R}^{n \times m} = \frac{1}{2}I + \frac{1}{2}P'$. This is the so-called "lazy" random walk.

Fact: If G is connected, then for any measure initial v on the vertex, $\lim_{t\to\infty} P^t v = \frac{d_i}{\sum_j d_j} = \phi_0(i)$. This ϕ converges to the stationary distribution. P is related to the normalized Laplacian.

If we look at the pre-asymptotic state, for $1 \ll t \ll \infty$, we could define similarity between vertex x and z in terms of the similarity between two density $P_t(x,\cdot)$ and $P_t(z,\cdot)$. That is,

- For $t \in (0,\infty)$, we want a metric between nodes s.t. x and z are close if $P_t(x,\cdot)$ and $P_t(z,\cdot)$ are close.
- There are various notions of closeness: *e.g.*, ℓ_1 norm (see the Szummer and Jaakkola reference); Kullback-Leibler divergences; ℓ_2 distances; and so on.
- The ℓ_2 distance is what we will focus on here (although we might revisit some of the others later).

In this setting, the ℓ_2 distance is defined as

$$\begin{aligned} D_t^2(x,z) &= ||P_t(x,\cdot) - P_t(z,\cdot)||^2_{1/\phi_0} \\ &= \sum_y \frac{(P_t(x,y) - P_t(z,y))^2}{\phi_0(y)}. \end{aligned} \tag{1}$$

(So, in particular, the weights $\frac{1}{\phi_0(y)}$ penalize discrepancies on domains of low density more than high-density domains.)

This notion of distance between points will be small if they are connected by many path. The intuition is that if there are many paths, then there is some sort of geometry, redundancy, and we can do inference like with low-rank approximations. (BTW, it is worth thinking about the difference between minimum spanning trees and random spanning trees, or degree and spectral measures of ranking like eigenvector centrality.)