Stat260/CS294: Spectral Graph Methods **Lecture 13 - 03/05/2015**

Lecture: Some Practical Considerations (2 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.

13 Basic perturbation theory and basic dimensionality reduction

Today, we will cover two topics: the Davis-Kahan-sin (θ) theorem, which is a basic result from matrix perturbation theory that can be used to understand the robustness of spectral clustering in idealized cases; and basic linear dimensionality reduction methods that, while not spectral graph methods by themselves, have close connections and are often used with spectral graph methods.

13.1 Basic perturbation theory

One way to analyze spectral graph methods—as well as matrix algorithms much more generally—is via matrix perturbation theory. Matrix perturbation theory asks: how do the eigenvalues and eigenvectors of a matrix A change if we add a (small) perturbation E, i.e., if we are working the the matrix $\tilde{A} = A + E$? Depending on the situation, this can be useful in one or more of several ways.

- **Statistically.** There is often noise in the input data, and we might want to make claims about the unobserved processes that generate the observed data. In this case, A might be the hypothesized data, e.g., that has some nice structure; we observe and are working with $\tilde{A} = A + E$, where E might be Gaussian noise, Gaussian plus spiked noise, or whatever; and we want to makeclaims that algorithms we run on $\tilde{A}$ say something about the unobserved A.
- **Algorithmically.** Here, one has the observed matrix A, and one wants to make claims about A, but for algorithmic reasons (or other reasons, but typically algorithmic reasons if randomness is being exploited as a computational resource), one performs random sampling or random projections and computes on the sample/projection. This amounts to constructing a sketch $\tilde{A} = A + E$ of the full input matrix A, where E is whatever is lost in the construction of the sketch, and one wants to provide guarantees about A by computing on $\tilde{A}$.
- **Numerically.** This arises since computers can't represent real numbers exactly, i.e., there is roundoff error, even if it is at the level of machine precision, and thus it is of interest to know the sensitivity of problems and/or algorithms to such roundoff errors. In this case, A is the answer that would have been computed in exact arithmetic, while $\tilde{A}$ is the answer that is computed in the presence of roundoff error. (E.g., inverting a non-invertible matrix is very sensitive, but inverging an orthogonal matrix is not, as quantified by the condition number of the input matrix.)

The usual reference for matrix perturbation theory is the book of Stewart and Sun, which was written primarily with numerical issues in mind.

Most perturbation theorems say that some notion of distance between eigenstuff, e.g., eigenvalues of subspaces defined by eigenvectors of A and $\tilde{A}$, depend on the norm of the error/perturbation E, often times something like a condition number that quantifies the robustness of problems. (E.g., it is easier to estimate extremal eigenvalues than eigenvectors that are buried deep in the spectrum of A, and it is easier if E is smaller.)

For spectral graph methods, certain forms of matrix perturbation theory can provide some intuition and qualitative guidance as to when spectral clustering works. We will cover one such results that is particularly simple to state and think about. When it works, it works well; but since we are only going to describe a particular case of it, when it fails, it might fail ungracefully. In some cases, more sophisticated variants of this result can provide guidance.

When applied to spectral graph methods, matrix perturbation theory is usually used in the following way. Recall that if a graph has k disconnected components, then $0 = \lambda_1 = \lambda_2 = \ldots = \lambda_k =< \lambda_{k+1}$, and the corresponding eigenvectors $v_1, v_2, \ldots, v_k$ can be chosen to be the indicator vectors of the connected components. In this case, the connected components are a reasonable notion of clusters, and the k-means algorithm should trivially find the correct clustering. If we let A be the Adjacency Matrix for this graph, then recall that it splits into k pieces. Let's assume that this is the idealized unobserved case, and the data that we observe, i.e., the graph we are given or the graph that we construct is a noisy version of this, call it $\tilde{A} = A + E$, where E is the noise/error. Among other things E will introduce "cross talk" between the clusters, so they are no longer disconnected. In this case, if E is small, then we might hope that perturbation theory would show that only the top k eigenvectors are small, well-separated from the rest, and that the k eigenvectors of $\tilde{A}$ are perturbed versions of the original indicator vectors.

As stated, this is not true, and the main reason for this is that λ_{k+1} (and others) could be very small. (We saw a version of this before, when we showed that we don't actually need to compute the leading eigenvector, but instead any vector whose Rayleigh quotient was similar would give similar results—where by similar results we mean results on the objective function, as opposed to the actual clustering.) But, if we account for this, then we can get an interesting perturbation bound. (While interesting, in the context of spectral graph methods, this bound is somewhat weak, in the sense that the perturbations are often much larger and the spectral gap are often much larger than the theorem permits.)

This result is known as the Davis-Kahan theorem; and it is used to bound the distance between the eigenspaces of symmetric matrices under symmetric perturbations. (We saw before that symmetric matrices are much "nicer" than general matrices. Fortunately, they are very common in machine learning and data analysis, even if it means considering correlation matrices XX^T or X^TX. Note that if we relaxed this requirement here, then this result would be false, and to get generalizations, we would have to consider all sorts of other messier things like pseudospectra.)

To bound the distance between the eigenspaces, let's define the notion of an angle (a canonical or principal angle) between two subspaces.

Definition 1. *Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be two p-dimensional subspaces of $\mathbb{R}^d$, and let V_1 and V_2 be two orthogonal matrices (i.e., $V_1^TV_1 = I$ and $V_2^TV_2 = I$) spanning $\mathcal{V}_1$ and $\mathcal{V}_2$. Then the* principal angles $\{\theta_i\}_{i=1}^d$ *are s.t.* $\cos(\theta_i)$ *are the singular values of $V_1^TV_2$.*

Several things to note. First, for $d = 1$, this is the usual definition of an angle between two vectors/lines. Second, one can define angles between subspaces of different dimensions, which is of interest if there is a chance that the perturbation introduces rank deficiency, but we won't need that here. Third, this is acutally a full vector of angles, and one could choose the largest to be *the* angle between the subspaces, if one wanted.

Definition 2. *Let* $\sin(\theta(\mathcal{V}_1, \mathcal{V}_2))$ *be the diagonal matrix with the sine of the canonical angles along the diagonal.*

Here is the Davis-Kahan-$\sin(\theta)$ theorem. We won't prove it.

Theorem 1 (Davis-Kahan)**.** *Let* $A, E \in \mathbb{R}^{n \times n}$ *be symmetric matrices, and consider* $\tilde{A} = A + E$*. Let* $S_1 \subset \mathbb{R}$ *be an interval; and deonte by* $\sigma_{S_1}(A)$ *the eigenvalues of* A *in* S_1*, and by* V_1 *the eigenspace corresponding to those eigenvalues. Ditto for* $\sigma_{S_1}\left(\tilde{A}\right)$ *and* $\tilde{V}_1$*. Define the distance between the interval* S_1 *and the spectrum of* A *outside of* S_1 *as*

$$\delta = \min\{\|\lambda - s\| : \lambda \text{ is eigenvalue of } A, \lambda \notin S_1, s \in S_1\}.$$

Then the distance $d\left(V_1, \tilde{V}_1\right) = \|\sin\theta\left(V_1, \tilde{V}_1\right)\|$ *between the two subspaces* V_1 *and* $\tilde{V}_1$ *can be bounded as*

$$d\left(V_1, \tilde{V}_1\right) \leq \frac{\|E\|}{\delta},$$

where $\|\cdot\|$ *denotes the spectral or Frobenius norm.*

What does this result mean? For spectral clustering, let L be the original (symmetric, and SPSD) hypothesized matrix, with k disjoint clusters, and let $\tilde{L}$ be the perturbed observed matrix. In addition, we want to choose the interval such that the first k eigenvalues of both L and $\tilde{L}$ are in it, and so let's choose the interval as follows. Let $S_1 = [0, \lambda_k]$ (where we recall that the first k eigenvalues of the unperturbed matrix equal 0); in this case, $\delta = |\lambda_k - \lambda_{k+1}|$, i.e., δ equals the "spectral gap" between the k^{th} and the $(k+1)^{st}$ eigenvalue.

Thus, the above theorem says that the bound on the distance d between the subspaces defined by the first k eigenvectors of L and $\tilde{L}$ is less if: (1) the norm of the error matrix $\|E\|$ is smaller; and (2) the value of δ, i.e., the spectral gap, is larger. (In particular, note that we need the angle to be less than 90 degrees to get nontrivial results, which is the usual case; otherwise, rank is lost).

This result provides a useful qualitative guide, and there are some more refined versions of it, but note the following.

- It is rarely the case that we see a nontrivial eigenvalue gap in real data.
- It is better to have methods that are robust to slow spectral decay. Such methods exist, but they are more involved in terms of the linear algebra, and so many users of spectral graph methods avoid them. We won't cover them here.
- This issue is analogous to what we saw with Cheeger's Inequality, were we saw that we got similar bounds on the objective function value for any vector whose Rayleigh quotient was close to the value of λ_2, but the actual vector might change a lot (since if there is a very small spectral gap, then permissible vectors might "swing" by 90 degrees).

- BTW, although this invalidates the hypotheses of Theorem 1, the results of spectral algorithms might still be useful, basically since they are used as intermediate steps, i.e., features for some other task.

That being said, knowing this result is useful since it suggests and explains some of the eigenvalue heuristics that people do to make vanilla spectral clustering work.

As an example of this, recall the row-wise reweighting we was last time. As a general rule, eigenvectors of orthogonal matrices are robust, but not otherwise in general. Here, this manifests itself in whether or not the components of an eigenvector on a given component are "bounded away from zero," meaning that there is a nontrivial spectral gap. For L and L_{rw}, the eigenvectors are indicator vectors, so there is no need to worry about this, since they will be as robust as possible to perturbation. But for L_{sym}, the eigenvector is $D^{1/2}\vec{1}_A$, and if there is substantial degree variability then this is a problem, i.e., for low-degree vertices their entries can be very small, and it is difficult to deal with them under perturbation. So, the row-normalization is designed to robustify the algorithms.

This "reweigh to robustify" is an after-the-fact justification. One could alternately note that all the results for degree-homogeneous Cheeger bounds go through to degree-heterogeneous cases, if one puts in factors of d_{max}/d_{min} everywhere. But this leads to much weaker bounds than if one considers conductance and incorporates this into the sweep cut. I.e., from the perspective of optimization objectives, the reason to reweigh is to get tighter Cheeger's Inequality guarantees.

13.2 Linear dimensionality reduction methods

There are a wide range of methods that do the following: construct a graph from the original data; and then perform computations on the graph to do feature identification, clustering, classification, regression, etc. on the original data. (We saw one example of this when we constructed a graph, computed its top k eigenvectors, and then performed k-means on the original data in the space thereby defined.) These methods are sometimes called *non-linear dimensionality reduction methods* since the constructed graphs can be interpreted as so-called kernels and since the resulting methods can be interpreted as kernel-based machine learning methods. Thus, they indirectly boil down to computing the SVD—indirectly in that it is in a feature space that is implicitly defined by the kernel. This general approach is used for many other problems, and so we will describe it in some detail.

To understand this, we will first need to understand a little bit about *linear dimensionality reduction methods* (meaning, basically, those methods that directly boil down to the computing the SVD or truncated SVD of the input data) as well as kernel-based machine learning methods. Both are large topics in its own right, and we will only touch the surface.

13.2.1 PCA (Principal components analysis)

Principal components analysis (PCA) is a common method for linear dimensionality that seeks to find a "maximum variance subspace" to describe the data. In more detail, say we are given $\{x_i\}_{i=1}^n$, with each $x_i \in \mathbb{R}^m$, and let's assume that the data have been centered in that $\sum_i x_i = 0$. Then,

our goal is to find a subspace P, and an embedding $\vec{y}_i = P\vec{x}_i$, where $P^2 = P$, s.t.

$$\text{Var}(\vec{y}) = \frac{1}{n}\sum_i ||Px_i||^2$$

is largest, *i.e.*, maximize the projected variance, or where

$$\text{Err}(\vec{y}) = \frac{1}{n}\sum_i ||x_i - Px_i||_2^2$$

is smallest, *i.e.*, minimize the reconstruction error. Since Euclidean spaces are so structured, the solution to these two problems is identical, and is basically given by computing the SVD or truncated SVD:

- Let $C = \frac{1}{n}\sum_i x_i x_i^T$, *i.e.*, $C \sim XX^T$.
- Define the variance as $\text{Var}(\vec{y}) = \text{Trace}(PCP)$
- Do the eigendecomposition to get $C = \sum_{i=1}^m \lambda_i \hat{e}_i \hat{e}_i^T$, where $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_m \geq 0$.
- Let $P = \sum_{i=1}^d \hat{e}_i \hat{e}_i^T$, and then project onto this subspace spanning the top d eigenfunctions of C.

13.2.2 MDS (Multi-Dimensional Scaling)

A different method (that boils down to taking advantage of the same structural result the SVD) is that of Multi-Dimensional Scaling (MDS), which asks for the subspace that best preserves interpoint distances. In more detail, say we are given $\{x_i\}_{i=1}^n$, with $x_i \in \mathbb{R}^D$, and let's assume that the data are centered in that $\sum_i x_i = 0$. Then, we have $\frac{n(n-1)}{2}$ pairwise distances, denoted Δ_{ij}. The goal is to find vectors $\vec{y}_i$ such that:

$$||\vec{y}_i - \vec{y}_j|| \approx \Delta_{ij}$$

We have the following lemma:

Lemma 1. *If Δ_{ij} denotes the Euclidean distance of zero-mean vectors, then the inner products are*

$$G_{ij} = \frac{1}{2}\left(\sum_k \left(\Delta_{ik}^2 + \Delta_{kj}^2\right) - \Delta_{ij}^2 - \sum_{kl}\Delta_{kl}^2\right)$$

Since the goal is to preserve dot products (which are a proxy for and in some cases related to distances), we will choose $\vec{y}_i$ to minimize

$$\text{Err}(\vec{y}) = \sum_{ij}\left(G_{ij} - \vec{y}_i \cdot \vec{y}_j\right)^2$$

The spectral decomposition of G is given as

$$G = \sum_{i=1}^n \lambda_i \hat{v}_i \hat{v}_i^T$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_n \geq 0$. In this case, the optimal approximation is given by

$$y_{i\xi} = \sqrt{\lambda_\xi} v_{\xi i}$$

for $\xi = 1, 2, \ldots, d$, with $d \leq n$, which are simply scaled truncated eigenvectors. Thus $G \sim X^T X$.

13.2.3 Comparison of PCA and MDS

At one level of granularity, PCA and MDS are "the same," since they both boil down to computing a low-rank approximation to the original data. It is worth looking at them in a little more detail, since they come from different motivations and they generalize to non-linear situations in different ways. In addition, there are a few points worth making as a comparison with some of the graph partitioning results we discussed.

To compare PCA and MDS:

- $C_{ij} = \frac{1}{n}\sum_k x_{ki}x_{kj}$ is a $m \times m$ covariance matrix and takes roughly $O((n+d)m^2)$ time to compute.
- $G_{ij} = \vec{x}_i \cdot \vec{x}_j$ is an $n \times n$ Gram matrix and takes roughly $O((m+d)n^2)$ time to compute.

Here are several things to note:

- PCA computes a low-dimensional representation that most faithfully preserves the covariance structure, in an "averaged" sense. It minimizes the reconstruction error

  $$E_{PCA} = \sum_i ||x_i = \sum_{\xi=1}^{m}(x_i \cdot e_\xi)e_\xi||_2^2,$$

  or equivalently it finds a subspace with minimum variance. The basis for this subsapce is given by the top m eigenvectors of the $d \times d$ covariance matrix $C = \frac{1}{n}\sum_i x_i x_i^T$.

- MDS computes a low-dimensional representation of the high-dimensional data that most faithfully preserve inner products, *i.e.*, that minimizes

  $$E_{MDS} = \sum_{ij} (x_i \cdot x_j - \phi_i \cdot \phi_j)^2$$

  It does so by computing the Gram matrix of inner products $G_{ij} = x_i \cdot x_j$, so $G \approx X^T X$. It the top m eigenvectors of this are $\{v_i\}_{i=1}^m$ and the eigenvalues are $\{\lambda_i\}_{i=1}^m$, then the embedding MDS returns is $\Phi_{i\xi} = \sqrt{\lambda_\xi} v_{\xi i}$.

- Although MDS is designed to preserve inner products, it is often motivated to preserve pairwise distances. To see the connection, let

  $$S_{ij} = ||x_i - x_j||^2$$

  be the matrix of squared interpoint distances. If the points are centered, then a Gram matrix consistent with these squared distances can be derived from the transformation

  $$G = -\frac{1}{2}\left(I - uu^T\right) S \left(I - uu^T\right)$$

  where $u = \frac{1}{\sqrt{n}}(1, \cdots, 1)$.

Here are several additional things to note with respect to PCA and MDS and kernel methods:

- One can "kernelize" PCA by writing everything in terms of dot products. The proof of this is to say that we can "map" the data A to a feature space $\mathcal{F}$ with $\Phi(X)$. Since $C = \frac{1}{n}\sum_{j=1}^{n}\phi(x_j)\phi(x_j)^T$ is a covariance matrix, PCA can be computed from solving the eigenvalue problem: Find a $\lambda > 0$ and a vector $v \neq 0$ s.t.

$$\lambda v = Cv = \frac{1}{n}\sum_{j=1}^{n}(\phi(x_j)\cdot v)\phi(x_j). \tag{1}$$

So, all the eigenvectors v_i with λ_i must be in the span of the mapped data, *i.e.*, $v \in \text{Span}\{\phi(x_1),\ldots,\phi(x_n)\}$, *i.e.*, $v = \sum_{i=1}^{n}\alpha_i\phi(x_i)$ for some set of coefficients $\{\alpha_i\}_{i=1}^{n}$. If we multiply (1) on the left by $\phi(x_k)$, then we get

$$\lambda(\phi(x_k)\cdot v) = (\phi(x_k)\cdot Cv), \qquad k = 1,\ldots,n.$$

If we then define

$$K_{ij} = (\phi(x_i),\phi(x_j)) = k(x_i,x_j) \in \mathbb{R}^{n\times n}, \tag{2}$$

then to compute the eigenvalues we only need

$$\lambda\vec{\alpha} = K\vec{\alpha}, \qquad \alpha = (\alpha_1,\ldots,\alpha_n)^T.$$

Note that we need to normalize (λ_k,α^k), and we can do so by $\hat{K} = K - 1_nK - K1_n - 1_nK1_n$. To extract features of a new pattern $\phi(x)$ onto v^k, we need

$$(v^k\cdot\phi(x)) = \sum_{i=1}^{m}\alpha_i^k\phi(x_i)\cdot\phi(x) = \sum_{i=1}^{m}\alpha_i^k k(x_i,x). \tag{3}$$

So, the nonlinearities enter:

  - The calculation of the matrix elements in (2).
  - The evaluation of the expression (3).

But, we can just compute eigenvalue problems, and there is no need to go explicitly to the high-dimensional space. For more details on this, see "An Introduction to Kernel-Based Learning Algorithms," by by Müller et al. or "Nonlinear component analysis as a kernel eigenvalue problem," by Schölkopf et al.

- Kernel PCA, at least for isotropic kernels K, where $K(x_i,x_j) = f(||x_i - x_j||)$, is a form of MDS and vice versa. For more details on this, see "On a Connection between Kernel PCA and Metric Multidimensional Scaling," by Williams and "Dimensionality Reduction: A Short Tutorial," by Ghodsi. To see this, recall that

  - From the distances-squared, $\{\delta_{ij}\}_{ij}$, where $\delta_{ij} = ||x_i - x_j||_2^2 = (x_i - x_j)^T(x_i - x_j)$, we can construct a matrix A with $A_{ij} = -\frac{1}{2}\delta_{ij}$.
  - Then, we can let $B = HAH$, where H is a "centering" matrix ($H = I - \frac{1}{n}1.1^T$). This can be interpreted as centering, but really it is just a projection matrix (of a form not unlike we we have seen).
  - Note that $B = HX(HX)^T$, (and $b_{ij} = (x_i - \bar{x})^T(x_j - \bar{x})$, with $\bar{x} = \frac{1}{n}\sum_i x_i$), and thus B is SPSD.

– In the feature space, $\tilde{\delta}_{ij}$ is the Euclidean distance:

$$\begin{aligned}\tilde{\delta}_{ij} &= (\phi(x_i) - \phi(x_j))^T(\phi(x_i) - \phi(x_j)) \\ &= ||\phi(x_i) - \phi(x_j)||_2^2 \\ &= 2(1 - r(\delta_{ij})),\end{aligned}$$

where the last line follows since with an isotropic kernel, where $k(x_i, x_j) = r(\delta_{ij})$. (If $K_{ij} = f(||x_i - x_j||)$, then $K_{ij} = r(\delta_{ij})$ $(r(0) = 1)$.) In this case, A is such that $A_{ij} = r(\delta_{ij}) - 1$, $A = K - 1.1^T$, so (fact) $HAH = HKH$. The centering matrix annihilates 11^T, so $HAH = HKH$.

So, $K_{MDS} = -\frac{1}{2}(I - ee^T)A(I - ee^T)$, where A is the matrix of squared distances.

So, the bottom line is that PCA and MDS take the data matrix and use SVD to derive embeddings from eigenvalues. (In the linear case both PCA and MDS rely on SVD and can be constructed in $O(mn^2)$ time $(m > n)$.) They are very similar due to the linear structure and SVD/spectral theory. If we start doing nonlinear learning methods or adding additional constraints, then these methods generalize in somewhat different ways.

13.2.4 An aside on kernels and SPSD matrices

The last few comments were about "kernelizing" PCA and MDS. Here, we discuss this kernel issue somewhat more generally.

Recall that, given a collection $\mathcal{X}$ of data points, which are often but not necessarily elements of $\mathbb{R}^m$, techniques such as linear Support Vector Machines (SVMs), Gaussian Processes (GPs), Principle Component Analysis (PCA), and the related Singular Value Decomposition (SVD), identify and extract structure from $\mathcal{X}$ by computing linear functions, i.e., functions in the form of dot products, of the data. (For example, in PCA the subspace spanned by the first k eigenvectors is used to give a k dimensional model of the data with minimal residual; thus, it provides a low-dimensional representation of the data.)

Said another way, these algorithms can be written in such a way that they only "touch" the data via the correlations between pairs of data points. That is, even if these algorithms are often written in such as way that they access the actual data vectors, they can be written in such a way that they only accesses the correlations between pairs of data vectors. In principle, then, given an "oracle" for a different correlation matrix, one could run the same algorithm by providing correlations from the oracle, rather than the correlations from the original correlation matrix.

This is of interest essentially since it provides much greater flexibility in possible computations; or, said another way, it provides much greater flexibility in statistical modeling, without introducing too much additional computational expense. For example, in some cases, there is some sort of nonlinear structure in the data; or the data, e.g. text, may not support the basic linear operations of addition and scalar multiplication. More commonly, one may simply be interested in working with more flexible statistical models that depend on the data being analyzed, without making assumptions about the underlying geometry of the hypothesized data.

In these cases, a class of statistical learning algorithms known as *kernel-based learning methods* have proved to be quite useful. These methods implicitly map the data into much higher-dimensional

spaces, e.g., even up to certain ∞-dimensional Hilbert spaces, where information about their mutual positions (in the form of inner products) is used for constructing classification, regression, or clustering rules. There are rwo points are important here. First, there is often an efficient method to compute inner products between very complex or even infinite dimensional vectors. Second, while general ∞-dimensional Hilbert spaces are relatively poorly-structured objects, a certain class of ∞-dimensional Hilbert spaces known as Reproducing kernel Hilbert spaces (RKHSs) are sufficiently-heavily regularized that—informally—all of the "nice" behaviors of finite-dimensional Euclidean spaces still hold. Thus, kernel-based algorithms provide a way to deal with nonlinear structure by reducing nonlinear algorithms to algorithms that are linear in some (potentially ∞-dimensional but heavily regularized) feature space $\mathcal{F}$ that is nonlinearly related to the original input space.

The generality of this framework should be emphasized. There are some kernels, e.g., Gaussian rbfs, polynomials, etc., that might be called *a priori kernels*, since they take a general form that doesn't depend (too) heavily on the data; but there are other kernels that might be called *data-dependent kernels* that depend very strongly on the data. In particular, several of the methods to construct graphs from data that we will discuss next time, e.g., Isomap, local linear embedding, Laplacian eigenmap, etc., can be interpreted as providing data-dependent kernels. These methods first induce some sort of local neighborhood structure on the data and then use this local structure to find a global embedding of the data into a lower dimensional space. The manner in which these different algorithms use the local information to construct the global embedding is quite different; but in general they can be interpreted as kernel PCA applied to specially-constructed Gram matrices. Thus, while they are sometimes described in terms of finding non-linear manifold structure, it is often more fruitful to think of them as constructing a data-dependent kernel, in which case they are useful or not depending on issues related to whether kernel methods are useful or whether mis-specified models are useful.