Partial-Hessian Strategies for Fast Learning of Nonlinear Embeddings

Max Vladymyrov and Miguel Á. Carreira-Perpiñán

Electrical Engineering and Computer Science
University of California, Merced
https://eecs.ucmerced.edu

August 30, 2012
Dimensionality reduction

Given a high-dimensional dataset $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N) \subset \mathcal{R}^D$ find a low-dimensional representation $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \subset \mathcal{R}^d$ where $d \ll D$.

Can be used for:

- Data compression.
- Visualization.
- Detect latent manifold structure.
- Fast search.
- ...
Graph-based dimensionality reduction techniques

▷ Input: (sparse) affinity matrix $W$ defined on a set of high-dimensional points $Y$.

▷ Objective function: minimization over the latent points $X$.

▷ Examples:
  • **Spectral methods**: Laplacian Eigenmaps (LE), LLE;
    ✔ closed-form solution;
    ✗ results can be bad.
  • **Nonlinear methods**: SNE, $t$-SNE, elastic embedding (EE);
    ✔ better results;
    ✗ slow to train, limited to small data sets.
**COIL-20 Dataset**

Rotations of 10 objects every $5^\circ$; input is greyscale images of $128 \times 128$.

\[ Y : \quad \begin{array}{cccccccc} 
\vdots 
\end{array} \]
General Embedding Formulation (Carreira-Perpiñán 2010)

For $\mathbf{Y} \in \mathcal{R}^{D \times N}$ matrix of high-d points and $\mathbf{X} \in \mathcal{R}^{d \times N}$ low-d points

$$E(\mathbf{X}, \lambda) = E^+(\mathbf{X}) + \lambda E^-(\mathbf{X}) \quad \lambda \geq 0$$

$E^+(\mathbf{X})$ is the attractive term:

- often quadratic,
- minimal with coincident points;

$E^-(\mathbf{X})$ is the repulsive term:

- often very nonlinear,
- minimal with points separated infinitely.

Optimal embeddings balance both forces.
## General Embedding Formulation: Special Cases

<table>
<thead>
<tr>
<th></th>
<th>( E^+ (\mathbf{X}) )</th>
<th>( E^- (\mathbf{X}) )</th>
</tr>
</thead>
</table>
| **SNE:**       | \[
\sum_{n,m=1}^N p_{nm} \| \mathbf{x}_n - \mathbf{x}_m \|^2
\]                                                                               | \[
\sum_{n=1}^N \log \sum_{m=1}^N e^{-\| \mathbf{x}_n - \mathbf{x}_m \|^2}
\] |
| (Hinton & Roweis, '03) |                                                                                         |                                                                                         |
| **t-SNE:**     | \[
\sum_{n,m=1}^N p_{nm} \log \left( 1 + \| \mathbf{x}_n - \mathbf{x}_m \|^2 \right)
\] | \[
\log \sum_{n,m=1}^N \left( 1 + \| \mathbf{x}_n - \mathbf{x}_m \|^2 \right)^{-1}
\] |
| (van der Maaten & Hinton, '08) |                                                                            |                                                                                         |
| **EE:**        | \[
\sum_{n,m=1}^N w_{nm}^+ \| \mathbf{x}_n - \mathbf{x}_m \|^2
\]                                                                               | \[
\sum_{n,m=1}^N w_{nm}^- e^{-\| \mathbf{x}_n - \mathbf{x}_m \|^2}
\] |
| (Carreira-Perpiñán, '10) |                                                                                         |                                                                                         |
| **LE & LLE:**  | \[
\sum_{n,m=1}^N w_{nm}^+ \| \mathbf{x}_n - \mathbf{x}_m \|^2
\] s.t. constraints                                                             | 0                                                                                       |
| (Belkin & Niyogi, '03) |                                                                                         |                                                                                         |
|                 |                                                                                         |                                                                                         |
|                 | \( w_{nm}^+ \) and \( w_{nm}^- \) are affinity matrices elements                      |                                                                                         |

\( p_{nm} \) and \( w_{nm} \) are probability and weight matrices elements respectively.
Optimization Strategy

For every iteration $k$:

1. Choose positive definite $B_k$.

2. Solve a linear system $B_k p_k = -g_k$ for a search direction $p_k$, where $g_k$ is the gradient.

3. Use line search to find a step size $\alpha$ for the next iteration $X_{k+1} = X_k + \alpha p_k$ (e.g. with backtracking line search).

Convergence is guaranteed! (under mild assumptions)
How to choose good $\mathbf{B}_k$?

Solve linear system $\mathbf{B}_k \mathbf{p}_k = -\mathbf{g}_k$:

$\mathbf{B}_k = \mathbf{I}$ (grad. descent) $\xrightarrow{\text{more Hessian information}}$ faster convergence rate $\rightarrow \mathbf{B}_k = \nabla^2 E$ (Newton’s method)

We want $\mathbf{B}_k$:

- contain as much Hessian information as possible;
- positive definite (pd);
- fast to solve the linear system and scale up to larger $N$. 
The Spectral Direction

The Hessian of the generalized embedding formulation is given by:

\[ \nabla^2 E = 4(L^+ - \lambda L^-) \otimes I_d + 8L^{xx} - 16\lambda \text{vec}(XL^q)\text{vec}(XL^q)^T \]

where \( L^+ \), \( L^- \), \( L^{xx} \), \( L^q \) are graph Laplacians.

\( B = 4L^+ \otimes I_d \) is a convenient Hessian approximation:

- block-diagonal and has \( d \) blocks of \( N \times N \) graph Laplacian \( 4L^+ \);
- always psd \( \Rightarrow \) global convergence under mild assumptions;
- \textbf{constant} for Gaussian kernel. For other kernels we can fix it at some \( X \);
- equal to the Hessian of the spectral methods: \( \nabla^2 E^+(X) \);
- “bends” the gradient of the nonlinear \( E \) using the curvature of the spectral \( E^+ \);
The Spectral Direction (computation)

Solve $Bp_k = g_k$ efficiently for every iteration $k$ (naively $O(N^3d)$):

- Cache Cholesky factor of $L^+$ in first iteration.
- (Further) sparsify the weights of $L^+$ with a $\kappa$-NN graph. Runtime is faster and convergence is still guaranteed.

<table>
<thead>
<tr>
<th></th>
<th>Cost per iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective function</td>
<td>$O(N^2d)$</td>
</tr>
<tr>
<td>Gradient</td>
<td>$O(N^2d)$</td>
</tr>
<tr>
<td><strong>Spectral direction</strong></td>
<td>$O(N\kappa d)$</td>
</tr>
</tbody>
</table>

This strategy adds almost no overhead when compared to the objective function and the gradient computation.
Experimental Evaluation: Methods Compared

Now:

- Gradient descent (GD), $B = I$
  (Hinton & Roweis, '03)
- fixed-point iterations (FP), $B = 4D^+ \otimes I_d$
  (Carreira-Perpiñán, '10)
- Spectral direction (SD), $B = 4L^+ \otimes I_d$
- L-BFGS.

More experiments and methods at the poster:

- Hessian diagonal update;
- nonlinear Conjugate Gradient;
- some other interesting partial-Hessian update.
COIL-20. Convergence analysis, s-SNE

COIL-20 dataset of rotated objects ($N = 720$, $D = 16384$, $d = 2$). Run the algorithms 50 times for 30 seconds each initialized randomly.

![Animation](image.png)
MNIST. \textit{t-SNE}

- $N = 20\,000$ images of handwritten digits (each a $28 \times 28$ pixel grayscale image, $D = 784$).
- One hour of optimization on a modern computer with one CPU.
Conclusions

- We presented a common framework for many well-known dimensionality reduction techniques.
- We presented the **spectral direction**: a new simple, generic and scalable optimization strategy that runs one to two orders of magnitude faster compared to traditional methods.

Ongoing work:

- The evaluation of $E$ and $\nabla E$ remains the bottleneck ($\mathcal{O}(N^2d)$). We can use Fast Multipole Methods to speed up the runtime.
- Avoid line search, use constant, near-optimal step sizes.
MNIST. Embedding after 20 min of EE optimization

Fixed-point iteration

Spectral direction

Animation
COIL-20. Convergence to the same minimum, s-SNE

We initialized $X_0$ close enough to $X_\infty$ so that all methods have the same initial and final points.
COIL-20: Homotopy optimization for EE

Start with small $\lambda$ where $E$ is convex and follow the path of minima to desired $\lambda$ by minimizing over $X$ as $\lambda$ increases. We used 50 log-spaced values from $10^{-4}$ to $10^2$. 

![Graph showing the number of iterations and time as $\lambda$ varies](chart.png)