A Learnable Spectral Memory Graph for Recognition and Segmentation

Timothée Cour [*]   Jianbo Shi [†]

[*]University of Pennsylvania,

[†]University of Pennsylvania, jshi@cis.upenn.edu

Computer & Information Science

Penn
Engineering

A Learnable Spectral Memory Graph for Recognition and Segmentation

by

Timothée Cour and Jianbo Shi

A Learnable Spectral Memory Graph for Recognition and Segmentation

Timothée Cour   **Jianbo Shi**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19103

Abstract

Image segmentation is often treated as an unsupervised task. Segmentation by human, in contrast, relies heavily on memory to produce an object-like clustering, through a mechanism of controlled hallucination. This paper presents a learning algorithm for memory-driven object segmentation and recognition. We propose a general spectral graph learning algorithm based on gradient descent in the space of graph weight matrix using derivatives of eigenvectors. The gradients are efficiently computed using the theory of implicit functions. This algorithm effectively learns a graph network capable of memorizing and retrieving multiple patterns given noisy inputs. We demonstrate the validity of this approach on segmentation and recognition tasks, including geometric shape extraction, and hand-written digit recognition.

Keywords: segmentation, recognition, learning spectral graph, derivative of eigenvectors, normalized cuts

1   Introduction

The act of seeing, the act of measuring photons in an image, is clearly different from the act of perceiving. When we perceive images, the human brain is constantly ignoring sharp shadow edges, hallucinating parts in largely occluded objects and detecting faint but familiar objects. One can conjecture that there is a complex interaction between memory of objects and image pixel analysis.

Image segmentation and object recognition have many aspects in common. Both can be viewed as classification problems where one wants to label pixels indicating image regions or object classes. In both problems, the information we rely on are local image features, regional feature-feature relationships, and global label consistency. In image segmentation, local features are pixel color/texture, and regional relationships are pair-wise pixel similarities. In object recognition, local features are object parts, and feature relationships are spatial compatibility of the object parts. These regional similarity/compatibility measures are integrated to reach a global segmentation/recognition labelling. Such a distributed representation could be a central part of object memory.

This computation process of local feature detection, global decision based on regional spatial feature combination is well captured in the Markov Random Field (MRF) formulation

of both segmentation and recognition problems. MRF has a probability formulation allowing one to apply Bayesian inferencing and learning procedure for tuning graph clique weights. However, both inferencing and learning are computationally difficult.

Spectral graph partitioning, such as Normalized Cut (Ncut)[7], has been developed in the image segmentation domain as a computationally efficient alternative to MRF. However, the graph structure has to be hand designed, and there is little knowledge on how to encode familiar object shapes beyond Gestalt laws. In order to encode high level object memory in spectral graph, an automated learning algorithm is needed.

Learning spectral graph cut turns out to be a difficult task, because eigenvectors are only implicitly defined. Two main attempts have been made in learning spectral graph cuts. Meila-Shi[5] learned the graph weight W_{ij} indirectly through an equivalent random walk matrix P_{ij}, with no explicit constraints on the Ncut eigenvector itself. Bach-Jordan[1] formulated a direct optimization of W with respect to its Ncut eigenvector. However their computation is based on making a differentiable approximation of eigenvector using power method. The resulting computation of derivatives of eigenvector is complex and can be computationally unstable.

We present in this paper a direct method for learning spectral graph cut, based on efficient computation of derivatives of Ncut eigenvectors *in exact analytical form*. This capability allows us to design a set of interesting recognition-segmentation graphs that can encode and detect complex objects. The paper is organized as follows. We describe in Sec. 2 object recognition and segmentation encoding in spectral graph framework. Sec. 3 introduces the error energy to be minimized by learning. Sec. 4 describes the learning algorithm and its convergence properties. We demonstrate our theory with experiments in Sec. 5, and then draw comparison with Recurrent Neural Networks in Sec. 6. We finally summarize our contributions in Sec. 7.

2 Encoding Recognition and Segmentation with Spectral Graph Cuts

A Spectral Memory Graph $G = \langle V, E, W \rangle$ consists of a set V of n nodes (pixels, features, class labels), a set of edges $E \subset V \times V$ indicating which nodes are connected, and a weight function W specifying their pair-wise similarity. Given an input image I, we detect a subset $V(I) \subset V$ of "on" features, inducing the subgraph $G(I) = \langle V(I), E \cap V(I) \times V(I), W(I) \rangle$. Depending on problems, $W(I) \in \mathbb{R}^{|V(I)| \times |V(I)|}$ is obtained by extracting rows and columns of W indexed by $V(I)$, or by a set of parameters Θ and feature outputs $F(I) = \{F_t(I)\}$: $W(I) = f(\Theta, F(I))$. The output for both segmentation and recognition in image I is encoded with Ncut eigenvector X_{ncut}, defined as the second eigenvector of the system $W(I)X = \lambda D_{W(I)}X$, where $D_W = diag(W\mathbf{1})$. The aim is to learn W either directly in the coefficient space (W_{ij}) or through its parameters Θ. Three applications are detailed below to illustrate the concept.

2.1 Case 1: Image Segmentation and Data Clustering

We construct an Image Segmentation Graph using the ideas outlined in [2, 6]. V is the set of pixels or data elements to be clustered. Features such as brightness, color, texture are computed on input image I, $F(I) = \{F_t(I)\}$, giving the following expression of $W(I)$:

$$W(I)_{ij} = \sum_{F_t} \alpha_{F_t} \exp(-\|F_t(I)_i - F_t(I)_j\|/\sigma_{F_t}) \qquad (1)$$

Thresholding $X_{ncut}[W(I)]$ determines the segmentation clusters. Our goal is to learn the parameters $\Theta = \{\alpha_{F_t}, \sigma_{F_t}\}$ using a set of training images with ground truth segmentation.

2.2 Case 2: Geometric Shape Detection

We construct a Shape Graph for detecting and enhancing specific geometrical shapes in random background cluster. $V = \{Edge_{x_i,\theta_j}\}$ is the set of image edge nodes, one for each possible edge position x_i and edge orientation θ_j. We encode invariance by translation and rotation of graph connections with:

$$W(Edge_{x_i,\theta_j}, Edge_{x_{i'},\theta_{j'}}) = f(x_{i'} - x_i, \theta_{j'} - \theta_j) \tag{2}$$

On input image I, edges are detected and quantized into spatial locations and edge orientations. This leads to a subset of nodes $Edge_{on}(I) \subset V$, with the corresponding subgraph weights $W(I) = W(Edge_{on}(I), Edge_{on}(I))$. $X_{ncut}[W(I)]$ is thresholded to determine foreground shape edges vs. background edges. The goal here is to learn function f.

2.3 Case 3: Object Recognition

We construct a Recognition Graph to classify images, given training image/label pairs (see Fig. 1). $V = V_{input} \bigcup V_{output}$ consists of image features (pixels for digit recognition) and classification labels. W has pixel-pixel connections W_{pp}, pixel-label connections $W_{p\ell}$, and label-label connections $W_{\ell\ell}$:

$$W = \begin{bmatrix} W_{pp} & W_{p\ell} \\ W_{p\ell}^T & W_{\ell\ell} \end{bmatrix} \tag{3}$$

To classify an image I, we first extract nodes $V(I) = V_{input}(I) \bigcup V_{output}$, where $V_{input}(I) \subset V_{input}$ indicates pixels with intensity greater than a threshold. This induces a subgraph with weight matrix $W(I)$ obtained by extracting rows and columns of W. $X_{ncut}[W(I)]$ is thresholded to determine which label node is grouped (therefore classified) with the pixel nodes detected. Note that we also obtain a segmentation of the foreground object pixels against the background noise pixels, thereby achieving both segmentation and recognition. One can also extend pixel nodes to object part nodes by using more complex image features. Choosing the right graph weights W can be quite tricky, because of conflicting overlap across images between pixels and labels: in the digit recognition case, we have potentially a huge number of possible input patterns (60,000 for MNIST database) with multiple classification labels (10 digits). The goal here is to learn the entries of W_{pp}, $W_{p\ell}$ and $W_{\ell\ell}$.
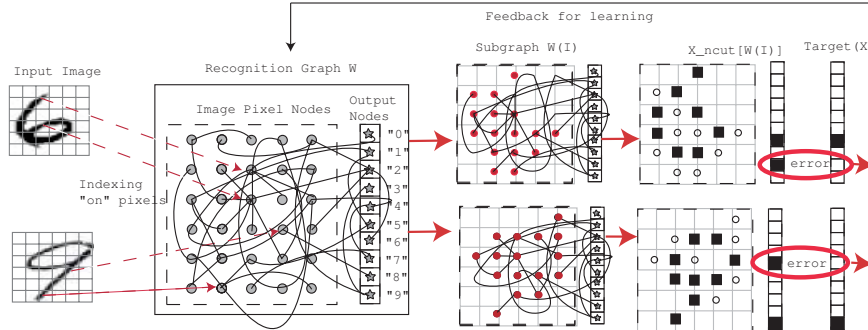

Figure 1: Recognition Graph classifying input patterns into multiple labels. Graph nodes V consist of input pixel nodes V_{input} and output classification nodes V_{output}. Graph weights W, to be learned, include pixel-pixel, pixel-label, and label-label connections. To classify an image I, "on" pixel nodes $V_{input}(I)$ are extracted by image thresholding; they index a corresponding subgraph with weight matrix $W(I)$. Ncut eigenvector $X_{ncut}[W(I)]$ is computed and thresholded, providing labelling of image pixels and classification nodes. Classification errors provide feedback to update W.

3 Error Energy for Learning Spectral Graph Cuts

Our goal is to store the desired input/output patterns using Ncut eigenvectors on graph nodes selected by the input image. Let $X^*(I) \in \mathbb{R}^n$ be a target label vector for segmentation/recognition of image I. In image segmentation and shape detection, the target is a $\{+1/-1\}$ indicator vector encoding two segments. In object recognition, we assign "+1" to the correct output node along with foreground object pixel nodes, and "−1" to all other output nodes along with background pixel nodes.

Definitions $X_p[W]$, λ_p are the pth largest eigenvector, eigenvalue of $WX = \lambda D_W X$ with $\|X_p[W]\| = 1$ and $D_W = diag(W\mathbf{1})$. Define $X_{ncut}[W] = X_2[W]$ for $W \in S_n^{2,X^*(I)}$, a certain subset[1] of S_n, the symmetric matrices in $\mathbb{R}^{n \times n}$. This subset avoids singularities when defining $X_{ncut}[W]$ uniquely. We will use later $W^\dagger$ the pseudo-inverse, $P_{X^\perp}(Z) = Z - (X^T Z)X$ the orthogonal projection along $X^\perp$. Define the one-target energy function:

$$\mathcal{E}(W, I) = \frac{1}{2}\|X_{ncut}[W(I)] - X^*(I)\|^2, \text{ for } W \in S_n^{2,X^*(I)} \tag{4}$$

The multi-target energy function is defined as $\mathcal{E}(W) = \sum_I \mathcal{E}(W, I)$, for $W \in \cap_I S_n^{2,X^*(I)}$. This error energy function has the following property, which will be useful later on when we try to learn the graph network.

Proposition 3.1 ($\mathcal{E}(W, I)$ **has no local minimum**) *The single target energy function has all its local minima in $S_n^{2,X^*} \bigcap \{W : \lambda_2(W) \neq -1\}$ equal to the global minimum, 0.*

The proof, in appendix, shows that at a critical point, the error vector $X_{ncut} - X^*(I)$ is in the kernel of a certain matrix of rank n-1. This shows in fact that $X_{ncut} - X^*(I)$ is proportional to X_{ncut}, which finally leads to $X_{ncut} = X^*(I)$.

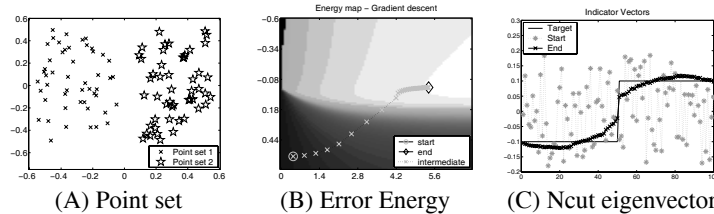(A) Point set   (B) Error Energy   (C) Ncut eigenvector

Figure 2: Learning point set clustering. $W(i, j) = \exp(-\sigma_x|x(i) - x(j)|) + \exp(-\sigma_y|y(i) - y(j)|)$. A) 2D layout of the points. B) Energy landscape of $\mathcal{E}(\sigma_x, \sigma_y)$, and gradient path taken by Eq.7, $(\dot{\sigma}_x, \dot{\sigma}_y) = -(\frac{\partial \mathcal{E}}{\partial \sigma_x}, \frac{\partial \mathcal{E}}{\partial \sigma_y})$. C) Target vector comparing with initial and final learned Ncut vector. The graph nodes are ordered according to their x-axis position.

4 Gradient Descent Learning Rule

Before explaining our solution for learning graph connections, we show why Hebbian learning rule is insufficient for our problem. This rule strengthens weights between nodes that co-occur in an image: $W_{ij} = \sum_I X^*(I)_i X^*(I)_j$ in our notation. Though intuitive,

---

[1] $S_n^{p,Y} = \{W \in S_n : D_W > 0, \lambda_p \text{ single}, \ker(W - \lambda_p D_W) \bigcap Y^\perp = \{0\}\}$. Y defines a polarity on eigenvector X_p via $Y^T X_p > 0$, so as to define X_p uniquely. Note the definition of $S_n^{p,Y}$ ensures that $Y^T X_p \neq 0$.

this rule cannot learn non-linearly separable functions (without hidden units), such as XOR. XOR is defined by the 4 patterns of (x, y, z_{XOR}) : $\{(0, 0, 0), (1, 0, 1), (0, 1, 1), (1, 1, 0)\}$. Hebbian rule gives $W(x, z) = W(y, z) = 0.5, W(x, y) = 0.25$, which is insufficient for coding $(x, y, z_{XOR}) = (1, 1, 0)$.

We minimize the error energy over W by gradient descent[2]: $W := W - \eta \frac{\partial \mathcal{E}}{\partial W}$. We study this learning rule with the continuous-time PDE $\dot{W} = -\frac{\partial \mathcal{E}}{\partial W} = -\frac{\partial \mathcal{E}}{\partial X_{ncut}[W]} \frac{\partial X_{ncut}[W]}{\partial W}$. The difficult part is to show that an exact analytical form of $\frac{\partial X_{ncut}}{\partial W}$ (with $n(n + 1)/2$ partial derivatives) *exists*, and that the resulting PDE *converges*.

Theorem 4.1 (Derivative of Ncut eigenvectors) *The map* $W \rightarrow (X_p, \lambda_p)$ *is* C^∞ *over* $S_n^{p,Y}$, *and we can express the derivatives over any* C^1 *path* $W(t)$ *as:*

$$\frac{dX_p[W(t)]}{dt} = -(W - \lambda_p D_W)^\dagger (W' - \lambda_p D'_W - \frac{d\lambda_p}{dt} D_W)X_p \quad (5)$$

$$\frac{d\lambda_p}{dt} = \frac{X_p^T(W' - \lambda_p D'_W)X_p}{X_p^T D_W X_p} \quad (6)$$

We obtain an analog theorem for the **derivative of standard eigenvectors**, by simply replacing D_W with I. The proof in appendix uses the implicit function theorem to show $X_p[W]$ is C^∞, then differentiates $WX_p = \lambda_p D_W X_p$: $(W - \lambda_p D_W)X'_p + (W' - \lambda_p D'_W - \lambda'_p D_W)X_p = 0$. Computation of Eq.5 is dominated by the pseudo-inverse $(W - \lambda_p D_W)^\dagger$, which requires $O(n^3)$ time if implemented naively as a matrix inversion. We remove this bottleneck by combining the term $(X_{ncut} - X^*(I))$ in $\frac{\partial \mathcal{E}}{\partial X_{ncut}}$ with $(W - \lambda_2 D_W)^\dagger$ in $\frac{\partial X_{ncut}}{\partial W}$. We define $Y = -(W - \lambda_2 D_W)^\dagger (X_{ncut} - X^*(I))$, and solve Y by $\begin{pmatrix} W - \lambda_2 D_W & X_{ncut} \\ X_{ncut}^T & 0 \end{pmatrix} \begin{pmatrix} Y \\ y \end{pmatrix} = -\begin{pmatrix} P_{X_{ncut}^\perp}(X_{ncut} - X^*(I)) \\ 0 \end{pmatrix}$. While the exact solution is also $O(n^3)$, an iterative solver can be used with $O(n^2)$ running time. Putting everything together, we obtain an efficient and exact gradient update rule as:

$$\frac{\partial \mathcal{E}}{\partial W}_{ij} = X_{ncut}^i Y_j + X_{ncut}^j Y_i - \lambda_2(X_{ncut}^i Y_i + X_{ncut}^j Y_j) - \lambda'_{2ij} Y^T D_W X_{ncut} \quad (7)$$

$$\text{with } \lambda'_{2ij} = (2X_{ncut}^i X_{ncut}^j - \lambda_2(X_{ncut}^{i\,2} + X_{ncut}^{j\,2}))/X_{ncut}^T D_W X_{ncut} \quad (8)$$

Empirically, we observe that $\mathcal{E}(W(t))$ converges to 0 exponentially fast when $W(t)$ follows the gradient path, even if the number of training examples grows as $O(n)$. We will prove this fact in the case of a single target. The convergence of $\mathcal{E}(W(t))$ however does not imply that of $W(t)$. Indeed, one can construct functions for which gradient descent leads to limit cycle oscillations. The following proposition shows that this cannot happen here.

Proposition 4.2 (Exponential convergence of the 1-target learning rule to a global minimum) *The PDE* $\dot{W} = -\frac{\partial \mathcal{E}}{\partial W}$ *either converges to a global energy minimum* W_∞, *or it escapes any compact* $K \subset S_n^{2,X^*}$. *In the first case,* $\mathcal{E}(W(t)) \rightarrow 0$ *exponentially.*

Our proof in appendix, shows that $\|\frac{\partial \mathcal{E}}{\partial W}\| \geq b\sqrt{\mathcal{E}}$, leading to the convergence of $W(t)$, and then $\frac{d}{dt}\mathcal{E}(W(t)) \leq -b^2 \mathcal{E}$, which shows the exponential decay of $\mathcal{E}(W(t))$.

The outline of our algorithm is the following:

---

[2]When W is parameterized by Θ, we have instead $\Theta := \Theta - \eta \frac{\partial \mathcal{E}}{\partial \Theta} = \Theta - \eta \frac{\partial \mathcal{E}}{\partial W} \frac{\partial W}{\partial \Theta}$

1. Initialize random matrix W_0 with small variance, but preferably large eigengap
2. Repeat 3-4 until $\sum_I E(W, I) >$ threshold:
3. Compute $\bar{X}_2, \lambda_2$ as second eigenvector, eigenvalue of $D_{W(I)}^{-1/2} W(I) D_{W(I)}^{-1/2}$;
$$X_{ncut} = D_{W(I)}^{-1/2} \bar{X}_2 / \|D_{W(I)}^{-1/2} \bar{X}_2\| \cdot sign(X^*(I)^T D_{W(I)}^{-1/2} \bar{X}_2)$$
4. Compute $W(I) = W(I) - \eta \frac{\partial \mathcal{E}(W,I)}{\partial W(I)}$ with Eq.7, and $\mathcal{E}(W, I) = \frac{1}{2}\|X_{ncut} - X^*(I)\|^2$

5   Experiments

Experiment 1, Fig.2, tested our learning algorithm on point set clustering as described in Sec. 2.1. The weight matrix W is parameterized by $\Theta = (\sigma_x, \sigma_y)$. We calculated the error energy landscape in the whole searching space of (σ_x, σ_y) by brute force. We verified that the gradients computed analytically by Eq.7 are accurate comparing with those directly measured on the energy landscape, as in Fig.2B.

Experiment 2, Fig.3, trained a network to detect rectangular shapes in a cluttered environment, as described in Sec.2.2. With only very few training examples, the network is able to encode and detect the rectangle even with large amount of background clutters.

Experiment 3, Fig.4, verified the algorithm's ability to encode multiple logic functions, AND, OR, XOR, in a single graph network. The graph contains 7 nodes: $\{x, \neg x, y, \neg y, z_{AND}, z_{OR}, z_{XOR}, \}$. A matrix W of size 7×7 was learned. To compute the output for $XOR(x = 1, y = 0)$, for example, a submatrix of W with nodes $(x, \neg y, z_{XOR})$ is extracted, on which Ncut eigenvectors are computed. Results demonstrate that multiple linearly and nonlinearly separable functions can be learned in one graph without hidden units. A network is also trained to encode XOR alone, to ensure that we are not leveraging the AND and OR nodes for learning XOR.

Experiment 4, Fig.5, tested the algorithm on performing recognition and segmentation for multiple object classes. The algorithm described in Sec.2.3 was implemented on MNIST database[4] using binary pixel values as raw features. A graph network W consisting of 410^2 entries was learned on images of 20^2 pixels belonging to 10 digit classes. It achieved around 11% testing error (with similar training error) on 5000 training/testing images. This error rate is high comparing with the state-of-the-art dedicated algorithms. However, our recognition network has an unique capability: it can determine $which$ image features are responsible for producing the classification (via symmetric input-output association).

6   Comparison with Recurrent Neural Networks

At first glance, Recurrent Neural Networks(RNN)[3], governed by $\dot{X}_{rnn} = -X_{rnn} + \sigma(WX_{rnn} + B)$, with σ a non-linear function, are quite similar to our system. Spectral Memory Graph, RNN and related Hopfield nets, Boltzmann machines, all store patterns in distributed network connections. The main difference is in how the patterns are stored. Our system uses input to select a subgraph $W(I)$ of the entire graph W, and encodes patterns using its Ncut eigenvector. By contrast, RNN clamp the value of certain input nodes, and store patterns in local energy minima of the entire graph: the steady-state $X_{rnn} = \sigma(WX_{rnn} + B)$ depends on W and B, the initial input. The non-linearity of σ is essential for RNN, otherwise the system would behave as a linear classifier. The learning rules are also similar but with an important difference. Recurrent back-propagation $W := W - \eta \frac{\partial \mathcal{E}_{rnn}}{\partial W}$, derived from gradient descent of the energy $\mathcal{E}_{rnn}(W) = \frac{1}{2}\|X_{rnn}(W, B) - X^*\|^2$, leads to $\frac{\partial \mathcal{E}_{rnn}}{\partial W} = Y_{rnn} X_{rnn}^T$ with $Y_{rnn} = -(W - diag(\frac{1}{\sigma'(X_{rnn})}))^{-1} (X_{nn} - X^*)$. The energy gradient expressed in Eq.7 is similar, but with essentially 2 more terms: $\frac{\partial \lambda}{\partial W_{ij}}$ shows

the sensibility of the eigenmode w.r.t. W, and $-\lambda(X_iY_i + X_jY_j)$ comes from our choice of ncut vectors instead of eigenvectors. Finally, eigencomputation of Ncut has a more efficient numerical solution than the non-linear systems used in RNN.
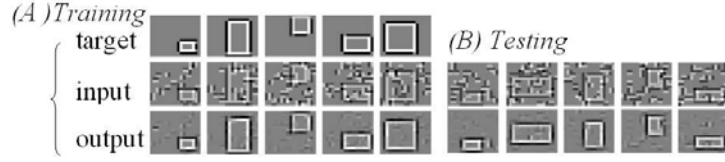

Figure 3: Square shape detection and enhancement. The Shape Graph weights $W(Edge_{x_i,\theta_j}, Edge_{x_{i'},\theta_{j'}}) = f(x_{i'} - x_i, \theta_{j'} - \theta_j)$ are learned with 50 training examples on image size $= 100 \times 100$. f has a total of 4096 free parameters to be learned (edge position displacements $\delta_x = \delta_y = 16$, edge orientation displacements $\delta_\theta = 4$).
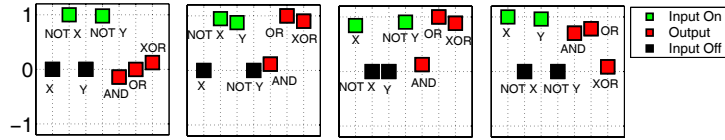

Figure 4: Learning functions AND, OR, XOR. Graph nodes: $\{x, \neg x, y, \neg y, z_{AND}, z_{OR}, z_{XOR}\}$. Ncut on a subgraph extracted from the learned graph W encodes the desired logical operations.
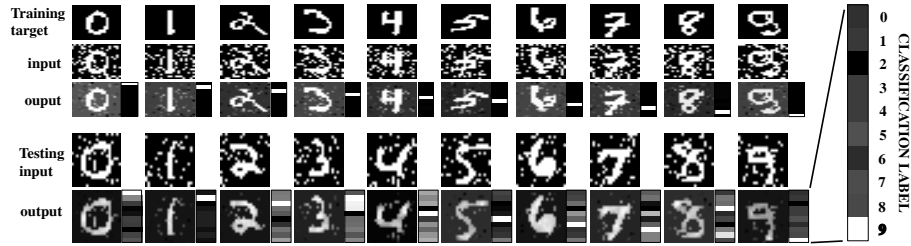

Figure 5: Digit recognition as detailed in Fig.1, on image size of 20×20 pixels with 10 labels. Recognition Graph weight matrix W has 410×410 entries to be learned. $X_{ncut}[W(I)]$ computed from W encodes 1) recognition: digit labelling by picking the output node (vertical bar next to image) with maximum (brightest) value, and 2) segmentation: removing irrelevant image pixels.

7 Conclusion

We make two contributions in this paper. First, we show that a spectral memory graph can memorize and recover multiple object patterns using distributed graph connections. Each input pattern invokes a selected subgraph on which the derived Ncut eigenvector encodes the desired output pattern. Second, we show that such graph connections can be learned directly by minimizing the error on its encoding Ncut eigenvectors. We prove that this learning algorithm converges at an exponential rate in a simple case.

References

[1] Francis R. Bach and Michael I. Jordan. Learning spectral clustering. *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[2] Charless Fowlkes, David Martin, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence(PAMI)*, 26(5):530–549, 2004.

[3] John Hertz, Anders Krogh, and Richard G. Palmer. *introduction to the theory of neural computation*. Addison Wesley Longman, 1991.

[4] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] Marina Meila and Jianbo Shi. Learning segmentation with random walk. *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[6] Andrew Y. Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[7] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence(PAMI)*, 22(8):888–905, 2000.

8   Appendix

Proof of theorem 4.1 Let $\Phi(W, X, \lambda) = \begin{pmatrix} WX - \lambda D_W X \\ X^T X - 1 \end{pmatrix}$ with $W \in S_n^{p,Y}$. Let $W_0 \in S_n^{p,Y}$, $\lambda_0 = \lambda_p(W_0)$, and $X_0 = X_p(W_0)$. $S_n^{p,Y}$ is open, $\Phi(W_0, X_0, \lambda_0) = 0$, and $\begin{pmatrix} \frac{\partial \Phi}{\partial X} & \frac{\partial \Phi}{\partial \lambda} \end{pmatrix}_{W_0, X_0, \lambda_0} = \begin{pmatrix} W_0 - \lambda_0 D_{W_0} & -D_{W_0} X_0 \\ X_0^T & 0 \end{pmatrix} \in GL_{n+1}$ (by expliciting the Kernel, noting λ_0 is single). The function given by the implicit function theorem matches X_p near W_0, so $(X_p(W), \lambda_p(W)) \in C^\infty(S_n^{p,Y})$. $\forall W(t) \in C^1(S_n^{p,Y})$, differentiating $WX = \lambda D_W X$ w.r.t. $W(t)$ yields $(W' - \lambda D')X + (W - \lambda D)X' - \lambda' DX = 0$. Left-multiplying by X^T gives the desired expression for λ'. For X', we notice that $X^T X' = 0$ and that $X' = \arg\min_{Z \in X^\perp} \|(W - \lambda D)Z + (W' - \lambda D' - \lambda' D)X\|$ $\square$

Proof of prop 3.1 $\frac{\partial \mathcal{E}}{\partial W_{ij}} = 0$ at a local minimum (S_n^{2,X^*} is open). If $|\lambda_2| < 1$, ($|\lambda_2| > 1$ is similar) we suppose WLOG that $|x_1| = \max\{|x_i|\}$. By Eq.7, $(\frac{\partial \mathcal{E}}{\partial W_{11}} \cdots \frac{\partial \mathcal{E}}{\partial W_{n1}}) = Y^T \left[\begin{pmatrix} x_1(1-\lambda) & \cdots & x_i - \lambda x_1 \\ & \ddots & 0 \\ 0 & & x_1 - \lambda x_i \end{pmatrix} - \begin{pmatrix} \cdots & \lambda'_{i1} DX & \cdots \end{pmatrix} \right] = Y^T [M_1 - M_2]; M_1 \in GL_n$ and $rk(M_2) = 1$ so $rk(M_1 - M_2) \geq n-1$. $X^T M_1 = X^T M_2$ proves easily, and $Y \in \ker(M1 - M2)^T$ so $Y \propto X$. The pseudo-inverse expression leads to $Y = 0$, then $X = X^*$ $\square$

Proof of prop 4.2 If $W(t)$ stays in a compact $K \subset S_n^{2,X^*}$, it defined $\forall t > 0$ so $\frac{d}{dt}\mathcal{E}(W(t)) \to 0$. $\frac{d}{dt}\mathcal{E}(W(t)) = -\|\frac{\partial \mathcal{E}}{\partial W}\|^2 \Rightarrow \frac{\partial \mathcal{E}}{\partial W}(W_\infty) = 0 \ \forall W_\infty$ in the ω-limit set; and by 3.1, $\mathcal{E}(W_\infty) = 0 \Rightarrow \mathcal{E}(W(t)) \to 0$. We then prove that $W(t) \to W_\infty$. As in 3.1, $\|\frac{\partial \mathcal{E}}{\partial W}\|^2$ writes as $\sum_{ij}\left(Y^T M^{(ij)}(X)\right)^2$, where $Y^T = -(X - X^*)^T(W - \lambda D_W)^\dagger$. Also, $\|Y\| \geq \sqrt{2\mathcal{E}(1-\mathcal{E})}\frac{1}{D_{max}(1-\lambda_n)}$, because $Sp((W - \lambda D_W)^\dagger) = (\frac{1}{1-\lambda_2}, 0, \ldots, \frac{1}{\lambda_n - \lambda_2})$, $Im((W - \lambda D_W)^\dagger) = X^\perp$, and after some calculus, $\|P_{X^\perp}(X - X^*)\|^2 = \|X - X^*\|^2 - \frac{1}{4}\|X - X^*\|^4$. Let $f(W, Z) = \sum_{ij}\left(Z^T M^{(ij)}(X(W))\right)^2$. By 3.1, $f(W, Z) = 0 \Rightarrow Z \propto X$, so $a = \min_{K \times (\mathbf{B(0,1)} \cap X^{*\perp})} f > 0 \Rightarrow \forall t, f(W(t), Y) \geq a \cdot \|Y\|^2$. This yields $\|\frac{\partial \mathcal{E}}{\partial W}\| \geq \frac{\sqrt{2a}}{D_{max}(1-\lambda_n)}\sqrt{\mathcal{E}(1-\mathcal{E})} \geq b\sqrt{\mathcal{E}}$ for some $b > 0$ and $t \geq t_0$. By a variant of Lojasiewicz' inequality (valid because $\int \frac{du}{b\sqrt{u}}$ converges near 0), $W \to W_\infty$. Finally, $\frac{d}{dt}\mathcal{E}(W(t)) \leq -b^2\mathcal{E} \Rightarrow \mathcal{E}(W(t)) \leq \mathcal{E}(W(0))e^{-b^2 t}$ $\square$