Deep Learning – Fall 2013

Instructor: Bhiksha Raj

Paper:

T. D. Sanger, "**Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network**", Neural Networks, vol. 2, pp. 459-473, 1989.

Presenter: Khoa Luu (kluu@andrew.cmu.edu)

Table of Contents

Introduction

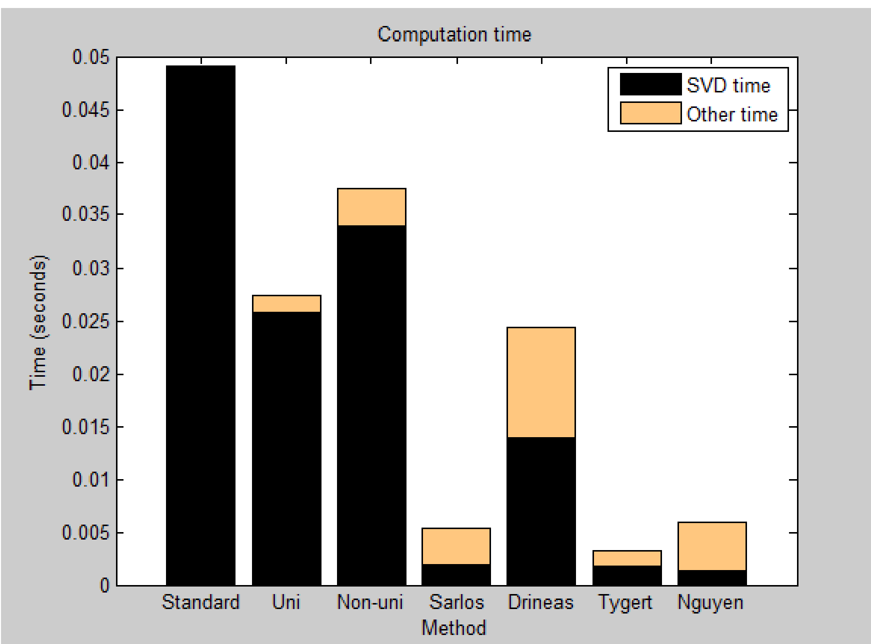
- **What are the aims of this paper?**
  - The optimal solution given by GHA whose weight vectors span the space defined by the *first few eigenvectors* of the correlation matrix of the input.
  - The method *converges* with probability one.

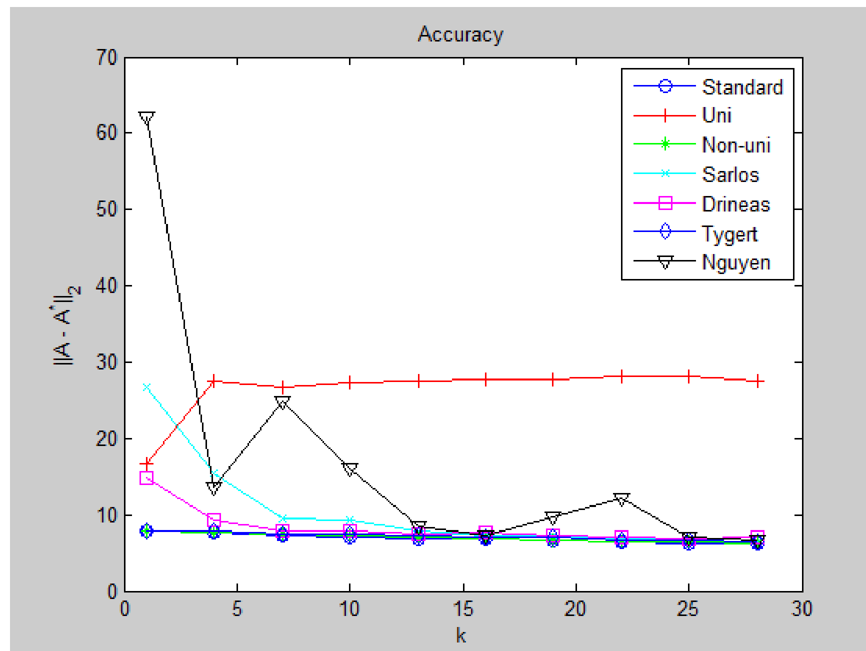
- **Why is Generalized Hebbian Algorithm (GHA) is important?**
  - Guarantee to find the eigenvectors directly from the data without computing correlation matrix that usually takes lots of memory.
  - Example: if network has 4000 inputs $\boldsymbol{x}$, then correlation matrix $\boldsymbol{x}\boldsymbol{x}^T$ has 16 million elements!
    - If the network has 16 outputs, then the computing outer products $\boldsymbol{y}\boldsymbol{x}^T$ take only 64,000 elements and $\boldsymbol{y}\boldsymbol{y}^T$ takes only 256 elements.

- **How does this method work?**
  - Next sections

Additional Comments

- This method is proposed in 1989, there are not many efficient algorithms for Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) as the ones nowadays.
- GHA is significant by then since it is *computational efficiency* and can be *parallelized*.





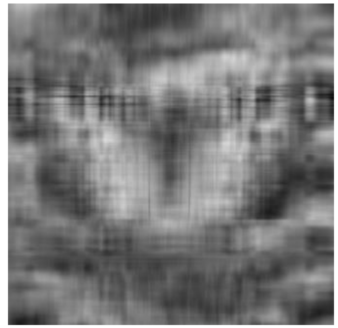
Some SVD algorithms proposed recently

Eigenproblem-related Applications

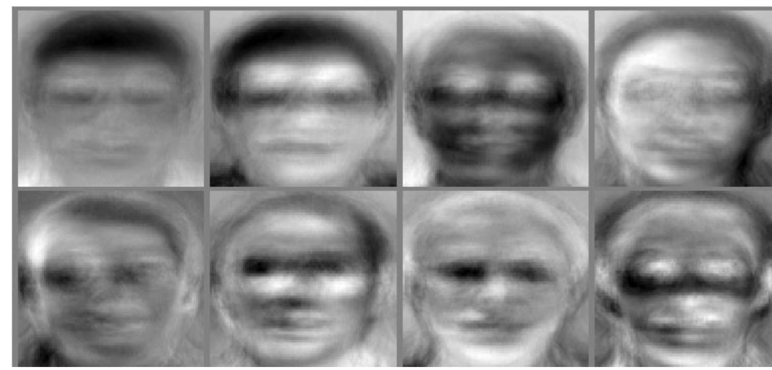
Image Encoding

40 principal components
(6.3:1 compression)

6 principal components
(39.4:1 compression)

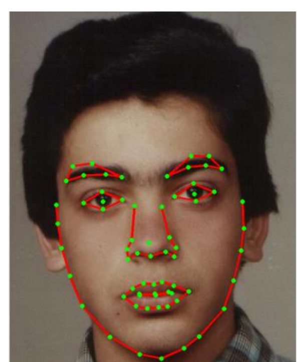


Eigenfaces – Face Recognition



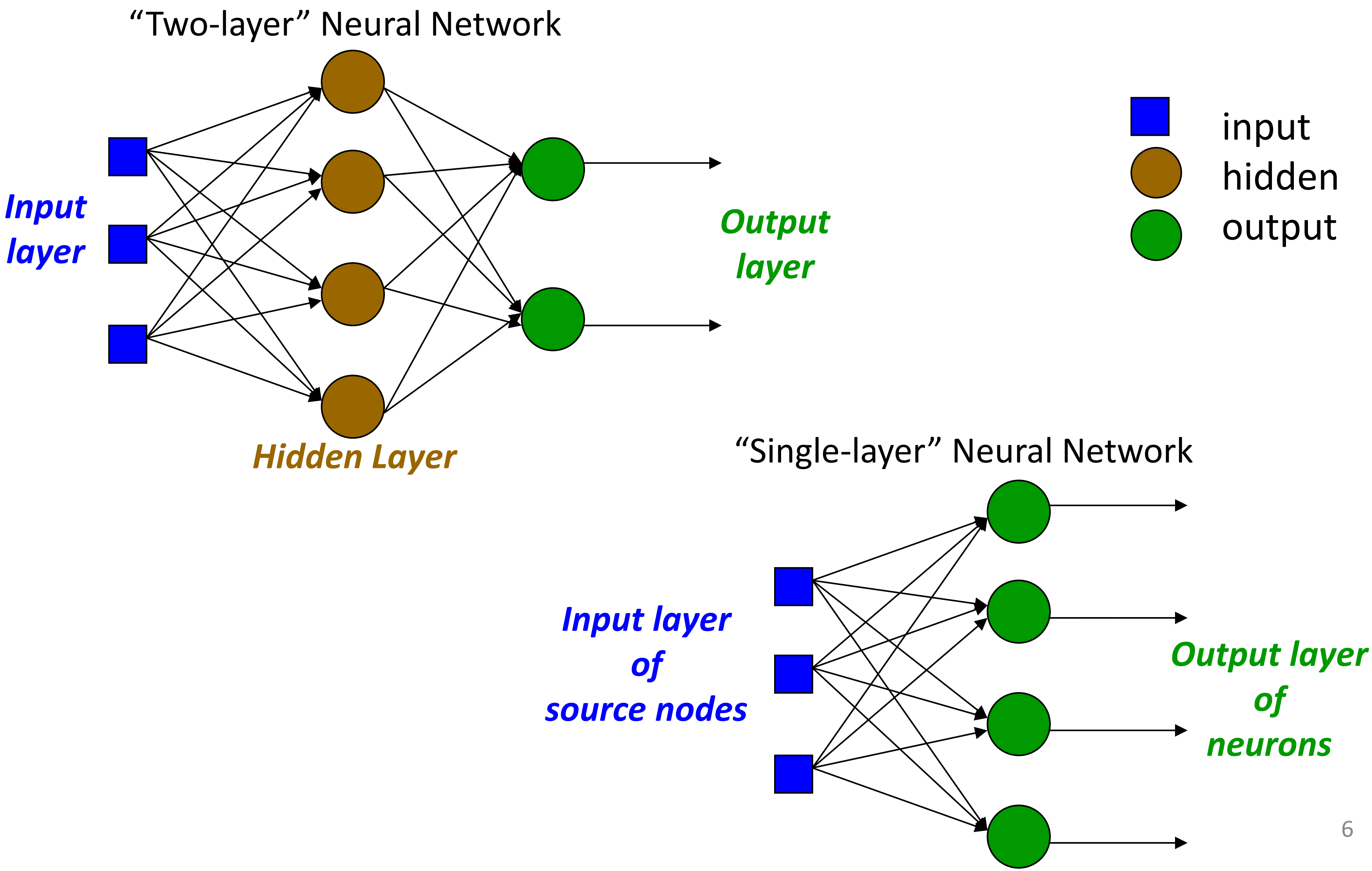
Object Reconstruction

Keypoints Detection



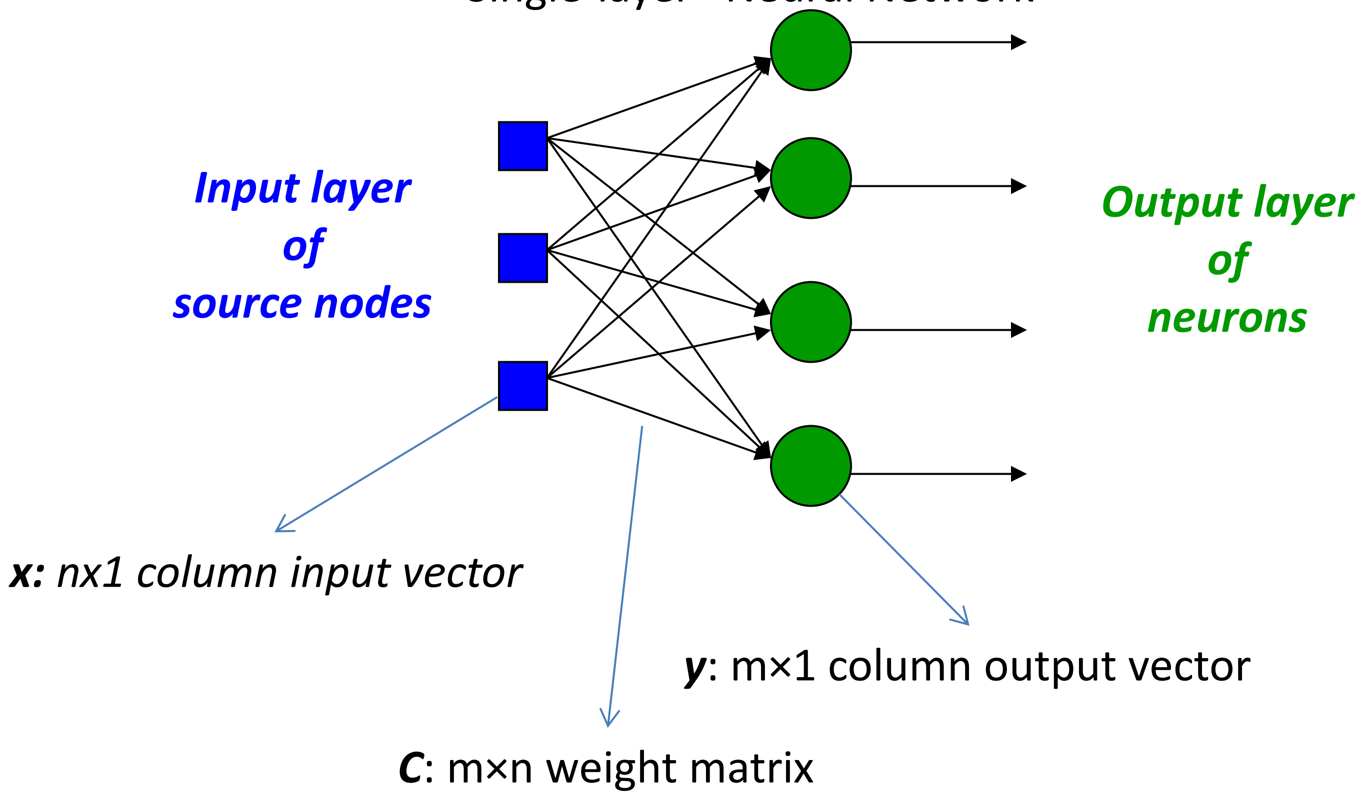
and more …

Unsupervised Feedforward Neural Network



Unsupervised Single-Layer Linear Feedforward Neural Network

"Single-layer" Neural Network



Some assumptions:

- Network is linear
- # of outputs < # of inputs (m < n)

Hebbian Algorithms

Proposed Updating Rule:

$$\boldsymbol{C}(t+1) = \boldsymbol{C}(t) + \gamma(y(t)x^{T}(t))$$

x: *nx1 column input vector*

$\boldsymbol{C}$: m×n weight matrix

$y = \boldsymbol{Cx}$: m×1 column output vector

γ: the rate of change of the weights

If all diagonal elements of $\boldsymbol{CC}^{T}$ equal to 1, then a Hebbian learning rule will cause the rows of $\boldsymbol{C}$ to converge to the principal eigenvector of $\boldsymbol{Q} = E[\boldsymbol{xx}^{T}]$.

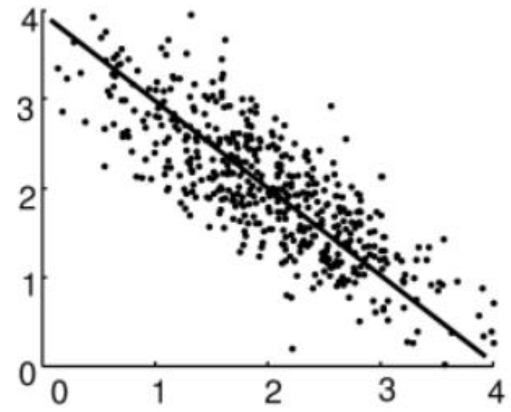
Hebbian Algorithms (cont.)

Proposed Network Learning Algorithm (Oja, 1982):

$$\Delta c_i = \gamma(y_i x - y_i^2 c_i)$$

The approximated differential equation: (why?)

$$c_i(t) = Qc_i(t) - (c_i(t)^T Qc_i(t))c_i(t)$$



Oja (1982) proved that for any choice of initial weights, c_i will converge to the principal component eigenvector e_1 (or its negative).

Hebbian learning rules tend to maximize the variance of the output units:

$$E[y_1] = E[(e_1 x)^2] = e_1^T Q e_1$$

Hebbian Algorithms (cont.)

The approximated differential equation (*where does it come from?*):

$$c_i(t) = Qc_i(t) - (c_i(t)^T Qc_i(t))c_i(t) \qquad (1)$$

An energy function (or the variance of the output y_i) can be defined as:

$$\varepsilon = -1/2\ c_i^T Q c_i$$

Minimizing ε subject to $c_i^T c_i = 1$ (that maximize the variance), we find the solution.

Apply *Gradient Descent* on ε for each element c_{ij} of c_i, we have:

$$\frac{\partial c_{ij}}{\partial t} = -\frac{\partial \varepsilon}{\partial c_i} = [Qc_i]_j$$

Or

$$\frac{\partial c_i}{\partial t} = Qc_i$$

First term of the right hand side of (1)

The second term of the right hand side of (1) is used for constraining $c_i^T c_i \approx 1$.

Generalized Hebbian Algorithm (GHA)

Oja algorithm only finds the first eigenvector e_1. If the network has more than one output, then we need to extend to GHA algorithm.

GHA = Oja algorithm + Gram-Schmidt Orth. process

Again, from Oja's update rule:

$$\Delta c_i = \gamma(y_i x - y_i^2 c_i)$$

Gram-Schmidt process in matrix form:

$$\Delta C(t) = -\mathrm{LT}(C(t)C(t)^T)\, C(t)$$

This equation orthogonalizes C in (m-1) steps if the row norms are kept in 1.

Then GHA is the combination:

$$\dot{C}(t) = C(t)Q - \mathrm{LT}[C(t)QC(t)^T]C(t)$$

LT[***A***]: operator to set all entries above diagonal of ***A*** to zeros.

In GHA, Oja alg. is applied to each row of C independently, thus causes all rows to converge to the principal eigenvector.

GHA Theorem - Convergence

Generalized Hebbian Algorithm

$$C(t+1) = C(t) + \gamma(t)h(\mathrm{C}(t), x(t)) \qquad (2)$$

where: $h(\mathrm{C}(t), x(t)) = y(t)x^{\mathrm{T}}(t) - \mathrm{LT}[y(t)\, y^{\mathrm{T}}(t)]C(t)$

Theorem 1:

If C is assigned random weights at t=0, then with probability 1, Eqn. (2) *will converge, and C will approach the matrix whose rows are the first m eigenvectors* $\{e_k\}$ *of the input correlation matrix* $Q = E[xx^{\mathrm{T}}]$*, ordered by decreasing eigenvalue* $\{\lambda_k\}$*.*

GHA Theorem - Proven

Proven using the ***Theorem 1 of Ljung*** **(*1977*)** *as follows:*

If

1. $\gamma(t)$ is a sequence of positive real numbers such that $\lim_{t\to\infty} \gamma(t) = 0$, $\sum_t \gamma(t)^p < \infty$ for some *p,* and $\sum_{t=0}^{\infty} \gamma(t) = \infty$.
2. $x(t)$ is bounded with prob. 1
3. $H(C, x)$ is continuously differentiable in *C* and *x*, and its derivative is bounded in time.
4. $\overline{h}(C) = \lim_{t\to\infty} E[h(C, x)]$ exists for each *C*.
5. *S* is locally asymptotically stable (Lyapunov sense) set for the differential equation
6. $C(t)$ enters some compact subset $A \subset D\{S\}$ infinitely often w.p. 1;

Then, with the prob. one,

$$\lim_{t\to\infty} C(t) \in S$$

GHA Theorem - Proven

Prove the algorithm:

$$C(t+1) = C(t) + \gamma(t)h(C(t), x(t))$$

Assumption:

$\gamma(t) = 1/t$ (since we need $\lim_{t\to\infty} \gamma(t) = 0$ and $\sum_{t=0}^{\infty} \gamma(t) = \infty$)

$$\overline{h}(C) = \lim_{t\to\infty} E[h(C,x)] = CQ - LT[CQC^T]C$$

We seek the stable points of the differential equation:

$$\dot{C} = CQ - \mathrm{LT}[CQC^T]C \qquad (3)$$

Assume that Q has n distinct strictly positive eigenvalues with corresponding orthonormal eigenvalues. We show the domain of attraction of the solution given by $C = T$ whose rows are the first m eigenvectors in descending eigenvalue order, includes all matrices C with bounded entry magnitudes.

GHA Theorem – Proven (cont.)

Induction method: *if the first (i-1) rows converge to the first (i-1) eigenvectors, then the i-th row will converge to the i-th eigenvector.*

When k = 1, with the first row of Eqn. (3):

$$\dot{c}_1^T = c_1^T Q - (c_1^T Q c_1^T)\, c_1^T$$

Oja, 1982 showed that this equation forces to c_1 to converge with prob. 1 to $\pm e_1$ which is the normalized principal eigenvector of Q (*slide 9*).

GHA Theorem – Proven (cont.)

When $k < i$, at any time t, we have

$$c_k(t) = e_k + \varepsilon_k(t).f_k(t) \qquad (4)$$

where e_k is the k-th ±eigenvector, $f_k(t)$ is a time-varying unit length vector, ε_k is scalar.

We need to show: when $t \to \infty$, then $c_i(t) \to e_i$.

Prove:

Each row in Eqn. (3) can be written as:

$$\dot{c}_i = \mathrm{Q}c_i - \sum_{k \le i} (ci^T Q c_k) c_k \qquad (5)$$

From (4) & (5), we have:

$$\dot{c}_i = \mathrm{Q}c_i - (c_i^T Q c_i) c_i - \sum_{k<i} (ci^T Q c_k) c_k - O(\varepsilon) + O(\varepsilon^2)$$

where $Qe_k = \lambda_k e_k$, ε is the term converging to zero that can be ignored when t is large.

Presenting c_i using entire orthonormal set of eigenvectors:

$$c_i = \sum_{k=0}^{N} \propto_k e_k$$

GHA Theorem – Proven (cont.)

When t is large, the following term can be derived:

$$\dot{c_i} = \sum_{k=0}^{N} \dot{\propto_k}\ e_k$$

where,

$$\dot{\propto_k} = \begin{cases} -\propto_k \sum_{l=0}^{N} \lambda_l \propto_l{}^2 & if\ k < i \\ \propto_k \left(\lambda_k - \sum_{l=0}^{N} \lambda_l \propto_l{}^2\right) & if\ k \geq i \end{cases}$$

We consider three cases: k < i, k > i, and k = i.

When k < i:

$$\dot{\propto_k} = -\eta \propto_k,$$

where η is strictly positive. It equation converges to zero with any initialization of $\propto_k$.

When k > i:

$$\dot{\theta_k} = \theta_k(\lambda_k - \lambda_i)$$

Since λ_i is the largest eigenvalue (decreasing order), therefore $\dot{\theta_k} \rightarrow 0$ for k> i.

GHA Theorem – Proven (cont.)

When k = i:

$$\dot{\propto}_i = \propto_i \left(\lambda_i - \lambda_i \propto_i{}^2 - \sum_{l \neq i}^{N} \lambda_l \propto_l{}^2\right)$$

Since $\propto_k \rightarrow 0$ for k < i, we have

$$\dot{\propto}_i = \propto_i \left(\lambda_i - \lambda_i \propto_i{}^2 - \sum_{l > i}^{N} \lambda_l \theta_l{}^2\right)$$

Since $\theta_k \rightarrow 0$ for k > i, therefore

$$\dot{\propto}_i = \lambda_i (\propto_i - \propto_i{}^3)$$

Apply *Lyapunov* function, we can prove the above equation converges.

Therefore, when t is large, the only significant α is α_i, so c_i will converge to $\pm e_i$.

GHA Theorem – Proven (cont.)

In addition, we also have to prove there exists a compact subset *A* of the set of all matrices such that *C*(*t*) ϵ *A* infinitely often with prob. 1 (the 6th *condition in slide 13*).

Define a norm of *C*:

$$\|C(t)\| = maxi_j(ci_j(t))$$

Set *A* as the compact subset of R^{nm} given by the set of matrices with norm less than or equal to some constant *a*.

Then, when a is sufficiently large, if $\|C(t-1)\| > a$, then $\|C(t)\| < \|C(t-1)\|$ with prob. 1 . Therefore, C will eventually remain within A with prob. 1 as *t* is large.

GHA – Local Implementation

The Generalized Hebbian Algorithm:

$$\Delta c_{ij(t)} = \gamma(t) y_i(t) \left(x_i(t) - \sum_{k \leq i} c_{kj}(t) y_k(t) \right) - \gamma(t) y_i{}^2(t) c_{ij}(t)$$

This Eqn. show how to implement the alg. Using only local operations. That point helps to train the neural networks using parallel programming techniques.

Demo

Thank you!