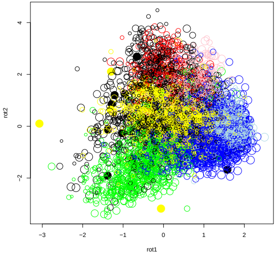


Linear methods for large data

Dean Foster

(U Penn $\rightarrow$ Yahoo Labs)

This is “part two” Martinsson’s talk

A few weeks ago we had Martinsson present:

> **“Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions”**

- It is my current favorite paper.
- Today, I’ll be applying it to a several problems in ML / statistics

Basic method

problem Find a low rank approximation to a $n \times m$ matrix M.

solution Find a $n \times k$ matrix A such that $M \approx AA^\top M$

Basic method

problem Find a low rank approximation to a $n \times m$ matrix M.

solution Find a $n \times k$ matrix A such that $M \approx AA^\top M$

Construction A is constructed by:

1. create a random $m \times k$ matrix Ω (iid normals)
2. compute $M\Omega$
3. Compute thin SVD of result: $UDV^\top = M\Omega$
4. $A = U$

FAST MATRIX REGRESSIONS

Using random methods for regression

Toy problem: $p \ll n$:

Using random methods for regression

Toy problem: $p \ll n$:

- Solving least squares: (a la Mahoney)
  - Generates provably accurate results.
  - Instead of np^2 time, it runs in np time.
  - This is fast! (I.e. as fast as reading the data.)

Using random methods for regression

Toy problem: $p \ll n$:

- Solving least squares: (a la Mahoney)
  - Generates provably accurate results.
  - Instead of np^2 time, it runs in np time.
  - This is fast! (I.e. as fast as reading the data.)

- But we should be unimpressed.

Using random methods for regression

Toy problem: $p \ll n$:

- Solving least squares: (a la Mahoney)
  - Generates provably accurate results.
  - Instead of np^2 time, it runs in np time.
  - This is fast! (I.e. as fast as reading the data.)

- But we should be unimpressed.

- Alternative fast (but stupid) method:
  - Do least squares on a sub-sample of size n/p
  - Runs in time np.
  - Same accuracy as the fast methods.

A better fast regression

- Create "sub-sample" $\hat{X} \equiv A^\top X$
- Estimate
$$\hat{\beta} = (\hat{X}^\top \hat{X})^{-1} X^\top Y$$
- Mahoney also subsampled Y and hence lost accuracy.

New method: Fast and accurate

- As fast as only reading the data (*np* time)
- As accurate as using all the data (NIPS 2013)

New method: Fast and accurate

- As fast as only reading the data (*np* time)
- As accurate as using all the data (NIPS 2013)

What about $p \gg n$?

New method: Fast and accurate

- As fast as only reading the data (*np* time)
- As accurate as using all the data (NIPS 2013)

What about $p \gg n$?

- Sub-sample the other side of the X matrix
- Generates a PCAs regression
- Sub-sample columns almost works
- Fast matrix multiply fixes the “almost” (NIPS 2013)
- Aside: yields fast ridge regression also (JMLR 2013)

Outline:

1. Streaming variable selection.
2. Fast CCAs.
3. Fast HMMs.
4. Fast parsing.
5. Fast clustering.

All are connected to the fast matrix decomposition.

(1) VIF regression

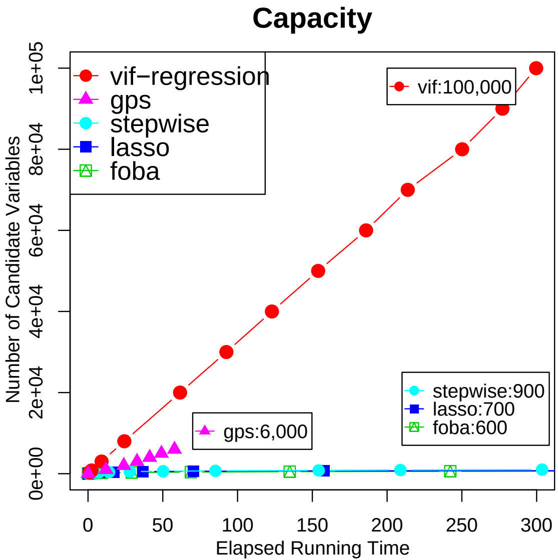
VIF regression

- Basic method: Stream over the features, trying them in order
- Even more gready than stepwise regression (2006)
- provides mFDR protection (2008)
- Instead of orthogonalizing each new X, only approximately orthogonalize it. (2011)
  - Can be done via sampling
  - Can be done use fast matrix methods

VIF regression

- Basic method: Stream over the features, trying them in order
- Even more gready than stepwise regression (2006)
- provides mFDR protection (2008)
- Instead of orthogonalizing each new X, only approximately orthogonalize it. (2011)
  - Can be done via sampling
  - Can be done use fast matrix methods
- For sub-modular problems, this will generate almost as good an estimator as best subsets. (2013)

VIF speed comparison



(2) CCA for Semi-supervised data

CCA: Usual data table for data mining

$$\begin{bmatrix} Y \\ \\ (n \times 1) \\ \\ \end{bmatrix} \begin{bmatrix} \quad X \quad \\ \\ (n \times p) \\ \\ \end{bmatrix}$$

with $p \gg n$

With unlabeled data

m rows of unlabeled data:

$$\begin{bmatrix} Y \\ \\ n \times 1 \\ \\ \end{bmatrix} \qquad \begin{bmatrix} X \\ \\ (n+m) \times p \\ \\ \\ \\ \\ \\ \end{bmatrix}$$

With alternative X's

m rows of unlabeled data, and two sets of equally useful X's:

$$\begin{bmatrix} Y \\ \\ n \times 1 \\ \\ \end{bmatrix} \quad \begin{bmatrix} X \\ \\ (n+m) \times p \\ \\ \\ \\ \\ \\ \end{bmatrix} \quad \begin{bmatrix} Z \\ \\ (n+m) \times p \\ \\ \\ \\ \\ \\ \end{bmatrix}$$

With: $m \gg n$

Examples

- Named entity recognition
  - Y = person / place
  - X = name itself
  - Z = words before target
- Modeling words in a sentence
  - Y = Current word
  - X = previous words
  - Z = future words
- Sitcom speaker identification:
  - Y = which character is speaking
  - X = video
  - Z = sound
- We will call these the multi-view setup

Using a CCA between X and Z to generate features

We can compute a CCA between X and Z to find a good subspace to use to predict Y.

- CCA = canonical correlation analysis
- Finds directions that are most highly correlated

Using a CCA between X and Z to generate features

We can compute a CCA between X and Z to find a good subspace to use to predict Y.

- CCA = canonical correlation analysis
- Finds directions that are most highly correlated
- Can be solved by doing successive regressions
- So, we can use our fast regression algorithms (2014)

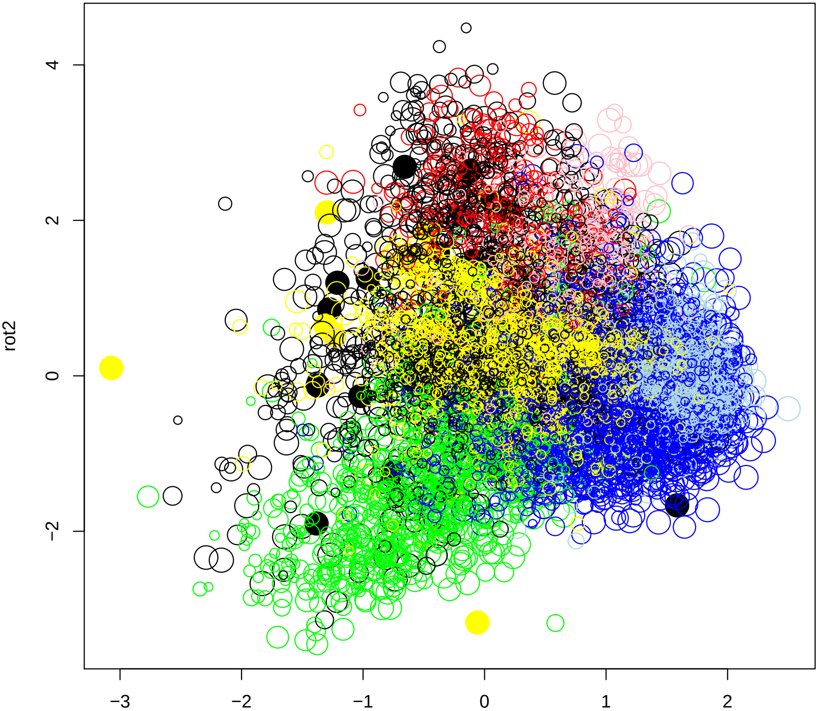
Using a CCA between X and Z to generate features

We can compute a CCA between X and Z to find a good subspace to use to predict Y.

- CCA = canonical correlation analysis
- Finds directions that are most highly correlated
- Can be solved by doing successive regressions
- So, we can use our fast regression algorithms (2014)

Results:

- Theory: Using the top few CCA directions is almost as good as the best linear model. (2006)
- We can use this to generate Eigenwords (ICML 2012)


rot2
4
2
0
−2
−3
−2
−1
0
1
2

Key

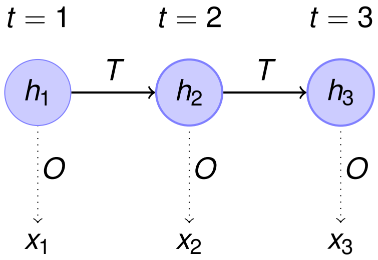
Colors:

- nouns = Blue (dark = NN1, light = NN2)
- verbs = red (dark = VV1, light = VV2)
- adj = green
- unk = yellow
- black = all else

Size = 1/Zipf order, top 50 are solid, rest are open.

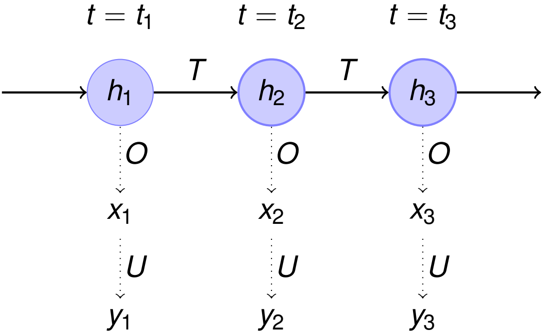
(3) HMMs

Hidden Markov Model



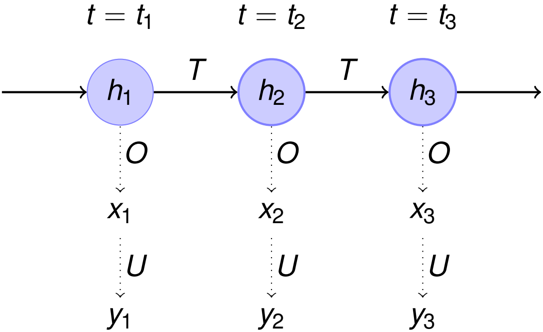
HMM with states h_1, h_2, and h_3 which generate observations x_1, x_2, and x_3.

Hidden Markov Model



The Y's are our eigenwords.

Hidden Markov Model



The Y's are our eigenwords.

$$\Pr(x_t, \ldots, x_1) = 1^T T \mathrm{diag}(OU^\top y_t) \cdots T \mathrm{diag}(OU^\top y_1) \pi$$

Results

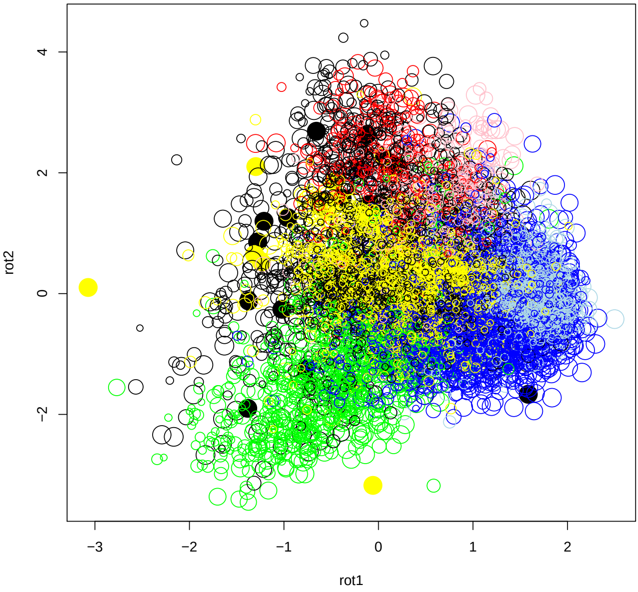
- Sample complexity (2010)
- Empirical results in NLP
  - Named Entity Recognition (CoNLL ’03 shared task)
  - Chunking (CoNLL ’00 shared task)
  - Eigenwords added signal to state of the art systems for both tasks
  - (2011)
- Neural data (2013)

(4) Parsing

Extending to trees

- We can extend the HMM material to dependency parsing
- Same sample complexity (2012)
- Raw MST Parser is 91.8% accurate
- Adding eigenwords: 2.6% error reduction
- eigenwords plus Re-ranking: 7.3% error reduction
- Extended to constituent parsing (2014)

(5) Clustering

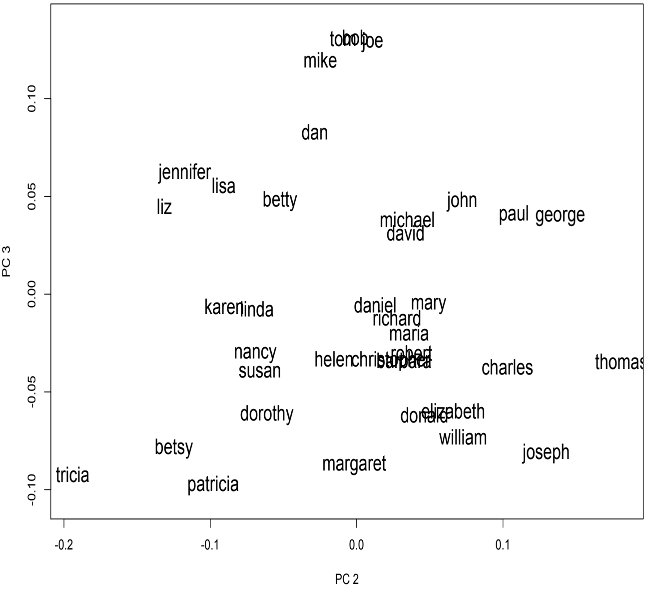


If you rotate this, you will see there are “pointy” directions

Clustering theorem

Theorem (with Hsu, Kakade, Liu, Anima, NIPS 2012)

Maximizing $E(\mu^\top X)^4$ *will find the natural coordinate system for LDA.*


tom
bob
joe
mike
dan
jennifer
lisa
liz
betty
john
paul
george
michael
david
karen
linda
daniel
mary
richard
maria
nancy
helen
robert
barbara
charles
thomas
susan
dorothy
elizabeth
donald
william
joseph
betsy
margaret
tricia
patricia
0.10
0.05
0.00
-0.05
-0.10
-0.2
-0.1
0.0
0.1
PC 2
PC 3

COAUTHORS

Coauthors

		U Penn	Elsewhere
Stat	faculty	Edward George Robert Stine	Tong Zhang (Rutgers) Sham Kakade (MSR)
	students	Dongyu Lin (ATT) Jordan Rodu (CMU) Kory Johnson Yichao Lu	
CS	faculty	Lyle Ungar	Michael Collins (Columbia) Daniel Hsu (Columbia)
	students	Parmaveer Dhillon Jing Zhou (real world)	Shay Cohen (Columbia) Karl Stratos (Columbia)

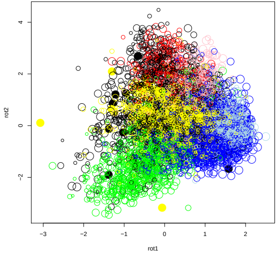
Coauthors

		U Penn	Elsewhere
Stat	faculty	Edward George Robert Stine	Tong Zhang (Rutgers) Sham Kakade (MSR)
	students	Dongyu Lin (ATT) Jordan Rodu (CMU) Kory Johnson Yichao Lu	
CS	faculty	Lyle Ungar	Michael Collins (Columbia) Daniel Hsu (Columbia)
	students	Parmaveer Dhillon Jing Zhou (real world)	Shay Cohen (Columbia) Karl Stratos (Columbia)

’14
’14
’13
’11 ’12 ’13
’11 ’12 ’13
’94 ’00 ’04 ’06 ’08 ’10 ’11 ’12 ’13

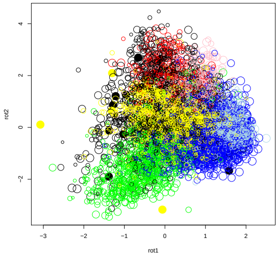
Conclusions

- These new fast matrix methods are easy to program.
- Generate statistically useful results.
- Room for interesting new probability theory.



Conclusions

- These new fast matrix methods are easy to program.
- Generate statistically useful results.
- Room for interesting new probability theory.



Thanks!

Fast Regressions

Theorem (with Yichao Lu, Parmaveer Dhillion, Lyle Ungar)

If $n > p^3$, then the algorithm defined by:

- *Let $m = \sqrt{n}$*
- *Pull out a sub-sample of size m from X's and call it Z.*
- *Let $\hat{\beta} \equiv (Z^\top Z)^{-1} X^\top Y$*

then the CPU time is $O(np)$ and accuracy is as good as the usually estimator.

Fast Principal components regressions

Theorem (with Yichao Lu, Parmaveer Dhillion, Lyle Ungar)

If $p > n$, then using a SRHT on the columns followed by regression will take $O(np\log(p))$ time and lose a constant factor on the statistical accuracy.

PCR is close to ridge regression

Theorem (with Sham Kakade, Parmaveer Dhillion, Lyle Ungar)

*A ridge regression can be quickly approximated by regressing on the top principal components. In particular, for a ridge parameter λ using components with singular values larger than λ will be within a factor of 4 of the ridge estimator on statistical accuracy. (*JMLR *2013)*

mFDR for streaming feature selection

Streaming feature selection was introduced in *JMLR* 2006 (with Zhou, Stine and Ungar).
Let $W(j)$ be the "alpha wealth" at time j. Then for a series of p-values p_j, we can define:

$$W(j) - W(j-1) = \begin{cases} \omega & \text{if } p_j \le \alpha_j \ , \\ -\alpha_j/(1-\alpha_j) & \text{if } p_j > \alpha_j \ . \end{cases} \tag{1}$$

Theorem

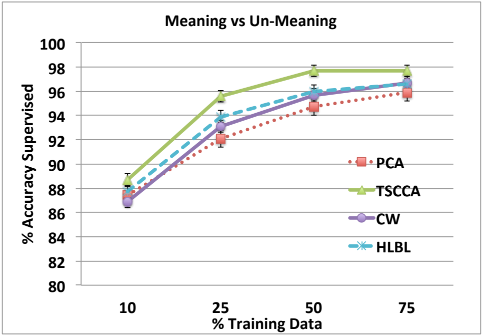
(Foster and Stine, 2008, JRSS-B*) An alpha-investing rule governed by (1) with initial alpha-wealth $W(0) \le \alpha\,\eta$ and pay-out $\omega \le \alpha$ controls mFDR$_\eta$ at level α.*

VIF regression

Theorem

(Foster, Dongyu Lin, 2011) VIF regression approximates a streaming feature selection method with speed $O(np)$.

Eigenwords to estimate PERMA



See paper for the predictions of the other 4:

- *Positive emotion* (aglow, awesome, bliss, . . .),
- *Engagement* (absorbed, attentive, busy, . . .),
- *Relationships* (admiring, agreeable, . . .),
- *Meaning* (aspire, belong, . . .)
- *Achievement* (accomplish, achieve, attain, . . .).

(P. Dhillon, J. Rodu, D. Foster and L. Ungar., *ICML 2012*)

Fast CCA method

(This is work in progress.) Yichao Lu has two current papers on this. The first shows how to use fast PCA and gradient decent to do a fast regression. The second shows how to use this successively to do a fast CCA. Kakade, Hsu and Zhang also have a fast CCA method, but it suffers from getting a less accurate answer than statistically optimal.

Submodular

Theorem

(Foster, Johnson, Stine, 2013) If the R-squared in a regression is submodular (aka subadditive) then a streaming feature selection algorithm will find an estimator whose out risk is within a factor of $e/(e-1)$ *of the optimal risk.*

HMM theorem

This is the first theorem we did for HMMs. We now have many other versions for parsing and extensions to continuous data.

Theorem (with Rodu, Ungar)

Let X_t be generated by an $m \geq 2$ state HMM. Suppose we are given a U which has the property that range$(O) \subset$ range(U) and $|U_{ij}| \leq 1$. Using N independent triples, we have

$$N \geq \frac{128m^2(2t+3)^2}{\epsilon^2 \; \Lambda^2 \sigma_m^4} \log\left(\frac{2m}{\delta}\right) \cdot \overbrace{\frac{\epsilon^2/(2t+3)^2}{\left(\sqrt[2t+3]{1+\epsilon}-1\right)^2}}^{\approx 1}$$

implies that

$$1 - \epsilon \leq \left| \frac{\widehat{\Pr}(x_1, \ldots, x_t)}{\Pr(x_1, \ldots, x_t)} \right| \leq 1 + \epsilon$$

holds with probability at least $1 - \delta$.

CCAs

Theorem

Let $\hat{\beta}$ be the Ridge regression estimator with weights induced by the CCA. Then under the multi-view assumption

$$Risk(\hat{\beta}) \leq \left(5\alpha + \frac{\sum \lambda_i^2}{n}\right)\sigma^2$$

CCAs

Theorem

Let $\hat{\beta}$ be the Ridge regression estimator with weights induced by the CCA. Then under the multi-view assumption

$$Risk(\hat{\beta}) \leq \left(5\alpha + \frac{\sum \lambda_i^2}{n}\right)\sigma^2$$

Estimator is least squares plus a penalty of:

$$\sum_i \frac{1-\lambda_i}{\lambda_i}\beta_i^2$$

Where λ_i's are the correlations

CCAs

Theorem

Let $\hat{\beta}$ be the Ridge regression estimator with weights induced by the CCA. Then under the multi-view assumption

$$Risk(\hat{\beta}) \leq \left(5\alpha + \frac{\sum \lambda_i^2}{n}\right)\sigma^2$$

Multiview property α is the multiview property:

$$\begin{aligned} \sigma_x^2 &\leq \sigma_{x,z}^2(1+\alpha) \\ \sigma_z^2 &\leq \sigma_{x,z}^2(1+\alpha) \end{aligned}$$

- 5α is the bias
- $\frac{\sum \lambda_i^2}{n}$ is variance

Results on ConLL task

- Results on 2 NLP sequence labeling problems: NER (CoNLL '03 shared task) and Chunking (CoNLL '00 shared task).
- Trained on $\sim$ 65 million tokens of unlabeled text in a few hours!

Relative reduction in error over state-of-the-art:

Embedding/Model	NER	Chunking
C&W	15.0%	18.8%
HLBL	19.5%	20.2%
Brown	12.1%	18.7%
Ando+Zhang	5.6%	14.6%

"Multi-View Learning of Word Embeddings via CCA," *NIPS* 2011.

Dependency parsing

In EMNLP 2012 (Rodu, Ungar, Dhillon, Collins) we extended the HMM results to dependency parsing.

Review paper on parsing

We have a review paper: “Spectral Learning of Latent-Variable PCFGs,” with Cohen, Stratos, Collins, and Ungar, submitting to *JMLR*.

Neural data: raw

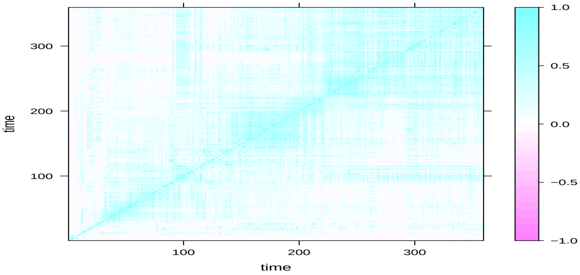


Figure 1: Correlation of raw observations, binned at 10 second bins

Neural data: reduced dimension

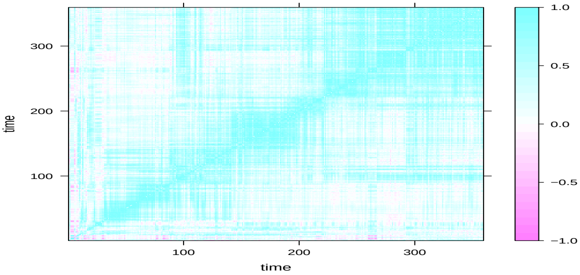


Figure 2: Correlations among reduced dimensional observations k=10

Neural data: state estimate

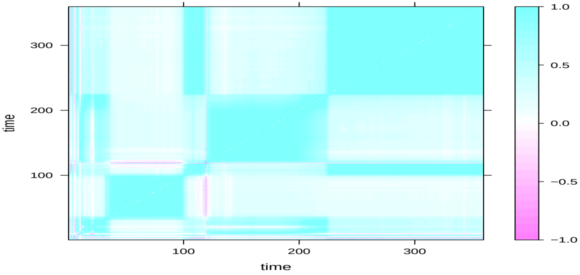


Figure 3: Correlations among the states of the system as time progresses k=10