Streaming Manifold Learning and DoE Applications

Shinjae Yoo

Computational Science Initiative

Joint work with Hao Huang (GE), Yangang Liu (BNL), Dmitri Zakharov (BNL), Eric Stach (BNL), Hayan Lee(LBL)

Motivation

- Big data stream
  - In-Situ Analysis
    - TEM : 3GB/s image stream
    - Astrophysics data (i.e. LSST)
    - Simulation (i.e. LES-DNS)
  - Drone/Satellite image stream
  - Meta-genomics (1TB reads)
  - Sensor Networks (i.e. SmartGrid)

- Streaming Analytics
  - Viable choice for high velocity and high volume in-situ data analytics
  - There is a clear gap between batch unsupervised learning and streaming unsupervised learning algorithms
  - Manifold learning has proven to outperform on high dimensional and complex data

Manifold Learning

Other Manifold Learning Methods:

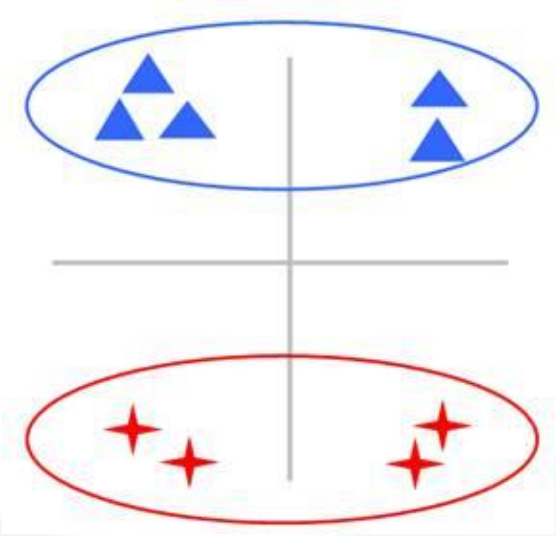
Diffusion Maps (DM), Local Linear Embedding (LLE), Laplacian Eigenmaps, Multi-demensional Scaling (MDS), …

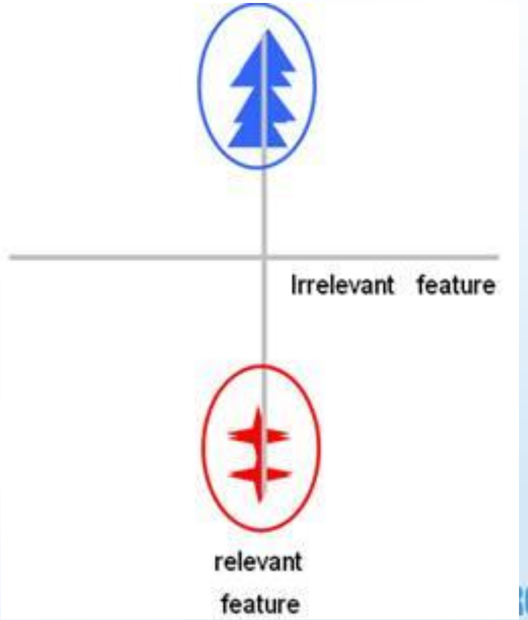
Outline

- Motivation
- Manifold Learning
- Feature Selection
- Spectral Clustering
- Applications

Why Feature Selection?

- Feature Selection is crucial for high volume, velocity, and dimensional data analysis:
- The goal: **minimize information loss** while **removing the noise and redundancy** in the feature space
  - model interpretation,
  - computational efficiency,
  - generalization ability.





Unsupervised Feature Selection

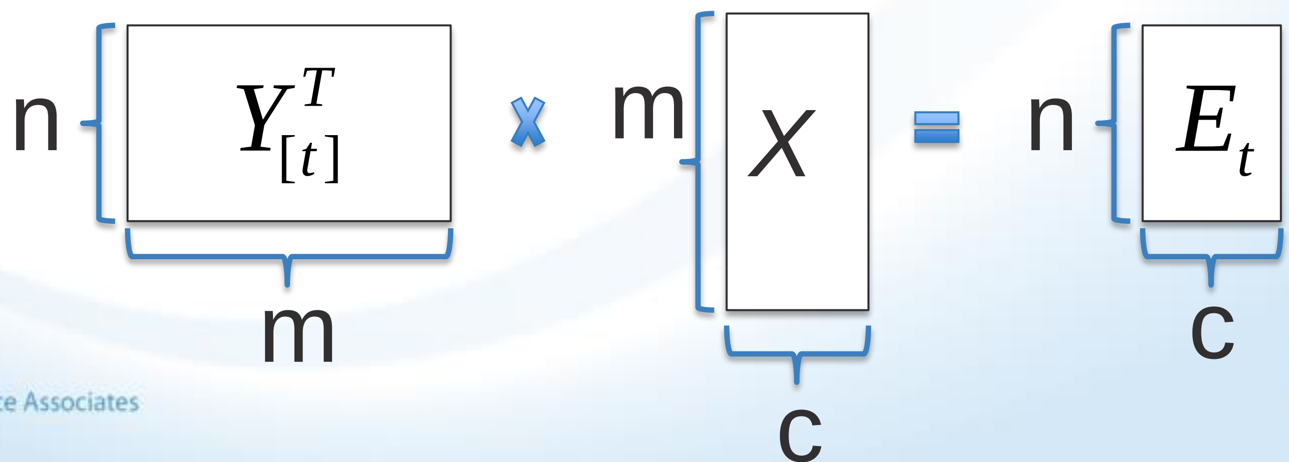
- Multi-Cluster Feature Selection (MCFS [Cai et.al. 2010]) defined the feature importance as:

$$Y_{[t]}^T = [Y_1 \mid Y_2 \mid \cdots \| Y_t]$$

$$EVD(L_{SYM}) = EVD(I - D_{[t]}^{-1/2} Y_{[t]}^T Y_{[t]} D_{[t]}^{-1/2}) = E_t \Lambda E_t^T$$

$$\min_X \left\| Y_{[t]}^T X - E_t \right\|^2 + \alpha \left\| X \right\|_1$$

$$\forall i \in [m], w_{t_i} = \max_{1 \leq h \leq k} |x_{t_{i,h}}|$$



Streaming Challenges

- Batch Algorithms
  - O(n^2) affinity matrix construction
  - Global normalization
  - O(n^3) decompositions

- Streaming Algorithm Requirements
  - Only one time reading
  - Limited processing power
  - Limited memory usage
  - Handling concept drifting or trend changes

General Feature Weighting (batch version)

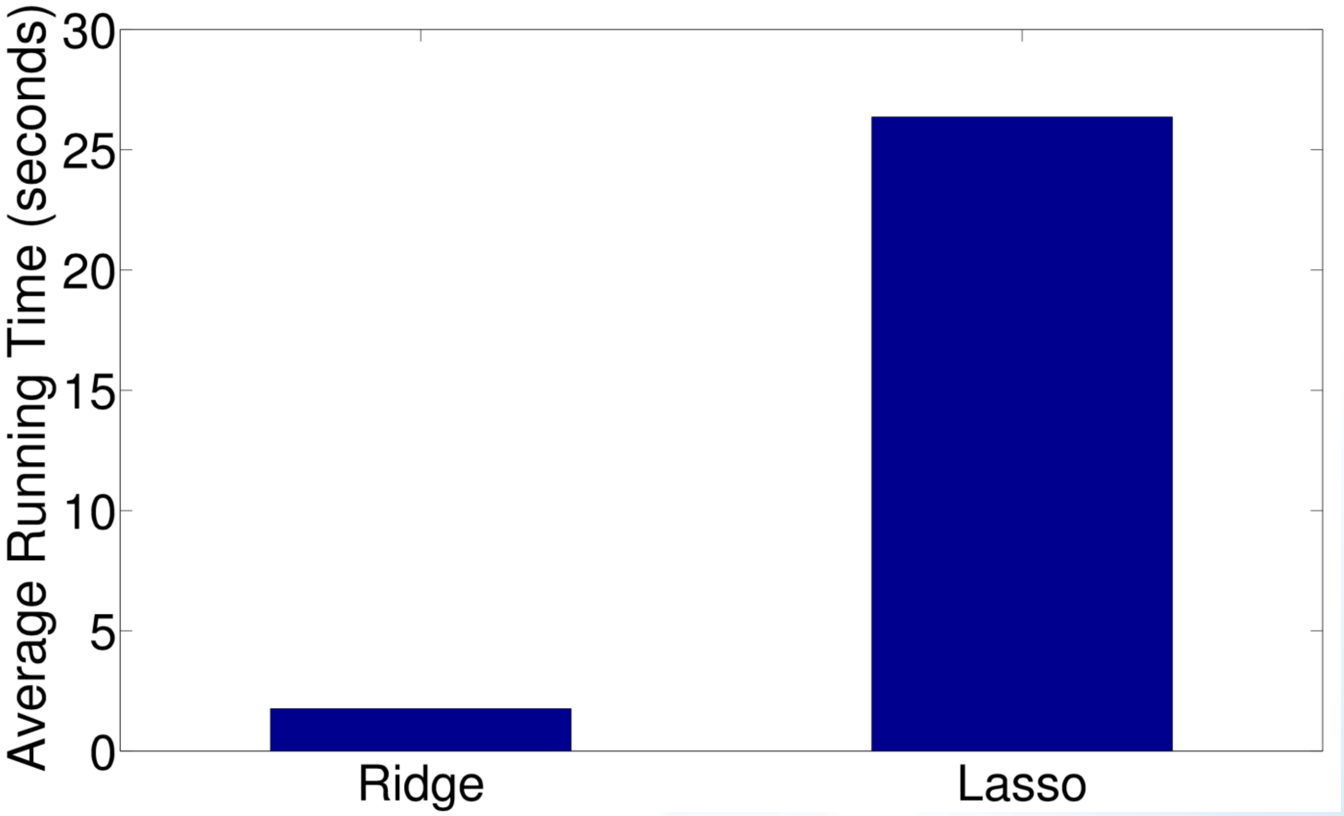
- Simplified normalization and generalize regression type

$$Y_{[t]} = [Y_1 \mid Y_2 \mid \cdots \| Y_t] = U_t \Sigma_t V_t^T \qquad Y_{[t]}^T Y_{[t]} = V_t \Sigma_t^2 V_t^T$$

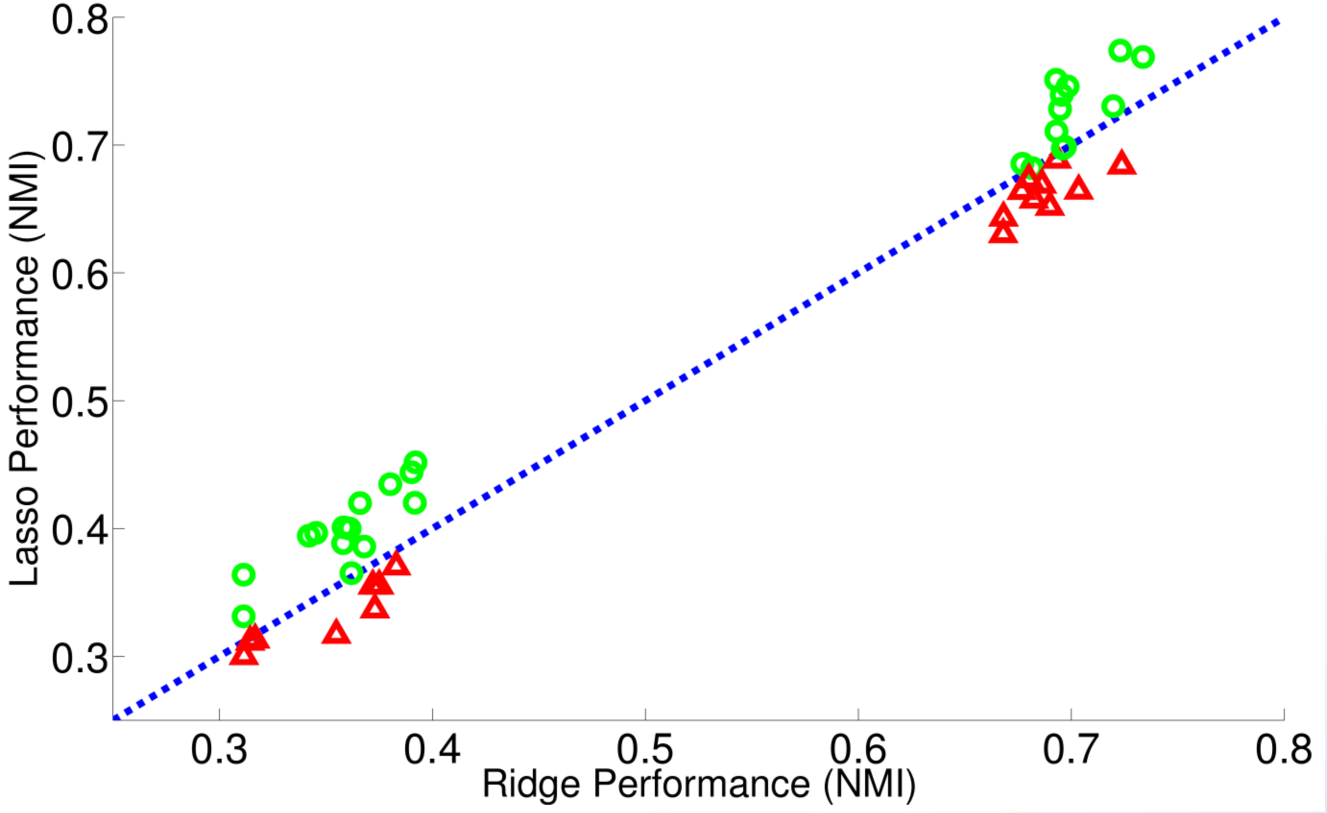
$$\min_{a_{k*}} \left\| Y_{[t]}^T X - V_t \right\|^2 + \alpha \left\| X \right\|_p^p$$

$$n\left\{\underbrace{\boxed{Y_{[t]}^T}}_{m}\right. \times m\left\{\underbrace{\boxed{X}}_{c}\right. = n\left\{\underbrace{\boxed{V_t}}_{c}\right.$$

Lasso vs Ridge



Lasso vs Ridge



Lasso vs Ridge

- Benefit of Ridge and why is it comparable?
  - Computationally cheaper
  - We do not need to solve Ridge solution
    By product of SVD decomposition

LEMMA 3.3. *Consider the ridge regression solution,*

$$X_t = argmin_{X=(x_{i,j})} \|Y_{[t]}^\top X - V_{t_{(k)}}\|_F^2 + \alpha \sum_{i,j} x_{i,j}^2.$$

Then we have the following:

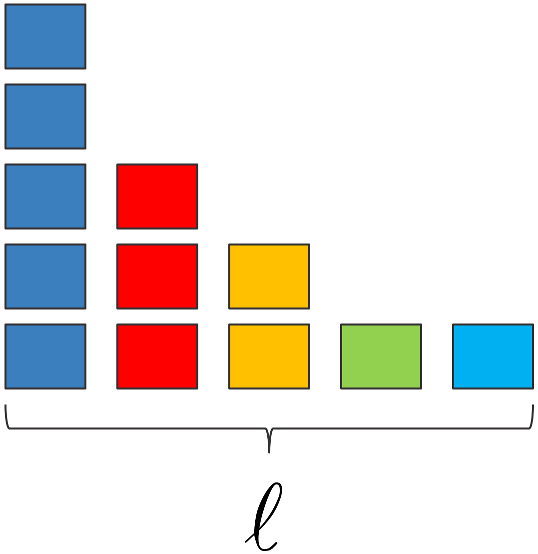
$$X_t = U_{t_{(k)}} \mathrm{diag}(\sigma_{t_1}/(\sigma_{t_1}^2 + \alpha), \ldots, \sigma_{t_k}/(\sigma_{t_k}^2 + \alpha)),$$

where $\sigma_{t_1}, \ldots, \sigma_{t_k}$ *are the top-k singular values in* $Y_{[t]_{(k)}}$.

$$\tilde{X}_{i,j}^L = sign(\hat{X}_{i,j})\{|\hat{X}_{i,j}| - \alpha/2\}_+$$
$$\tilde{X}^R = \hat{X}/(1+\alpha),$$

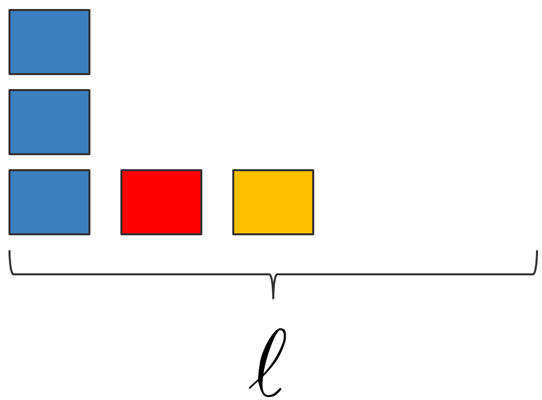
Streaming SVD

- Frequent Items
  - Capture top ℓ frequent items from the item stream



Streaming SVD

- Frequent Items
  - Capture top ℓ frequent items from the item stream

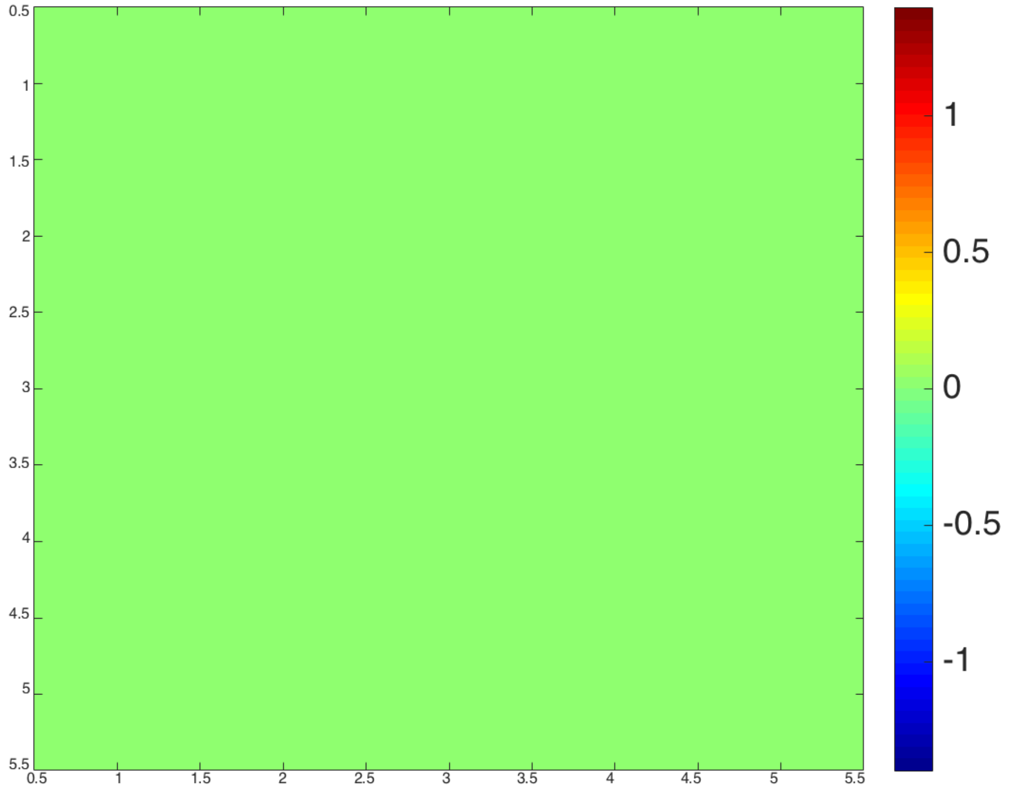


By setting $\ell = \frac{2}{\varepsilon}$

$$|f(i) - f'(i)| \leq \varepsilon n$$

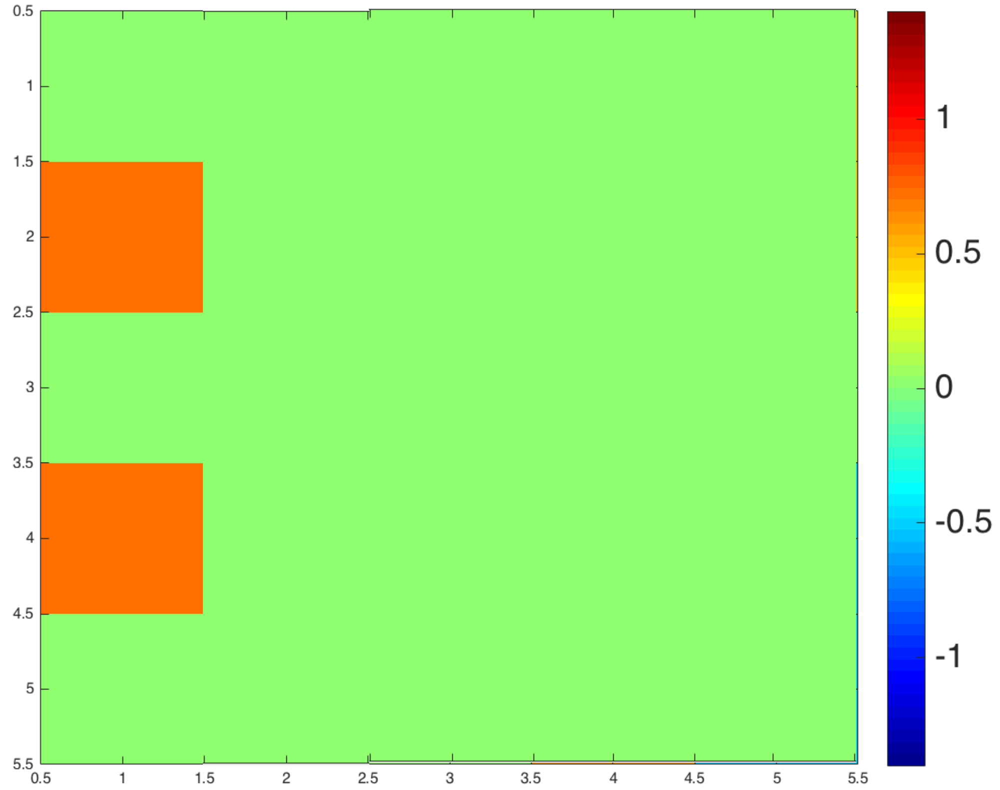
Streaming SVD

- Frequent Directions
  - Capture top ℓ frequent directions from data stream



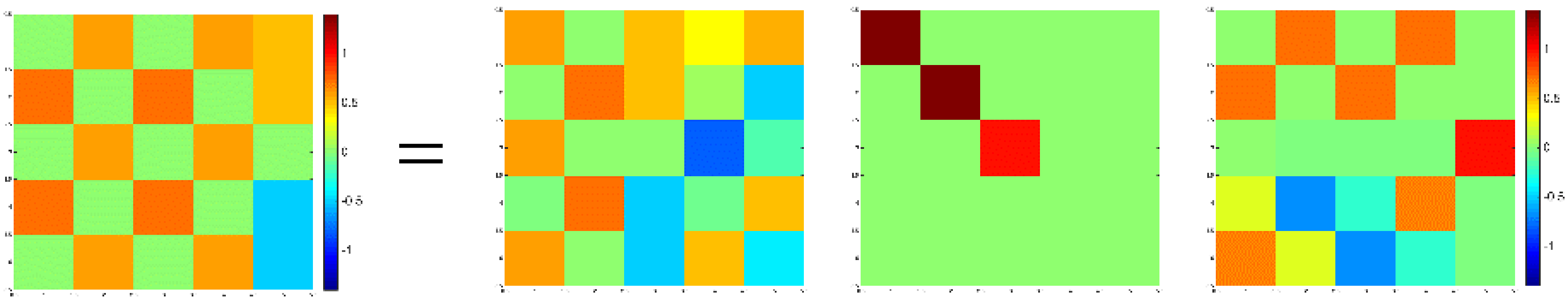
Streaming SVD

- Frequent Directions
  - Capture top k frequent directions from data stream

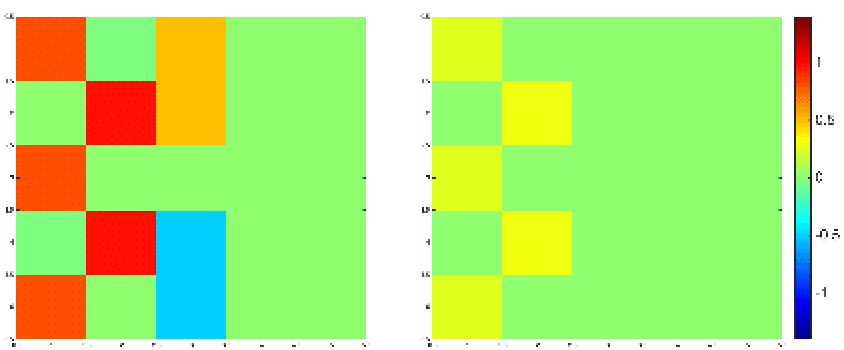


Streaming SVD

- Frequent Directions
  - Capture top k frequent directions from data stream



$$Y = U \quad \Sigma \quad V^T$$



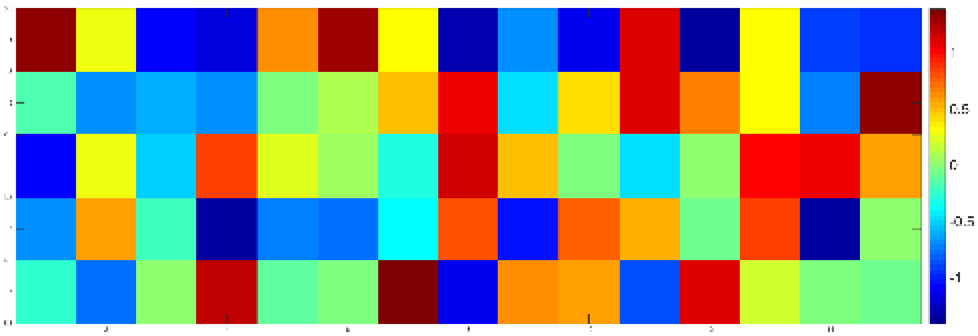
$$B = U\Sigma \qquad B^{new} = U\breve{\Sigma}$$

$$\left\| YY^T - BB^T \right\| \le \varepsilon \left\| Y \right\|_F^2$$

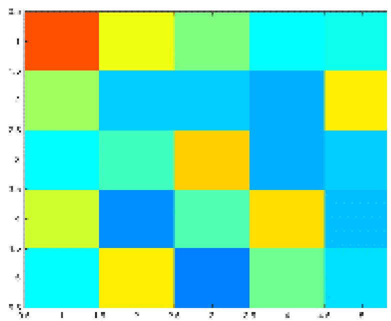
Streaming SVD

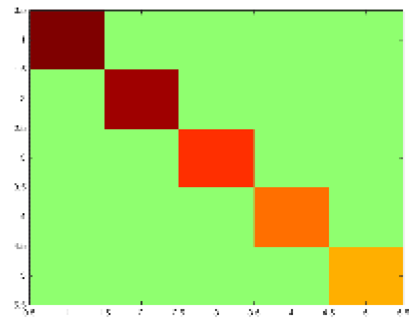
Our extension

- Why one at a time?

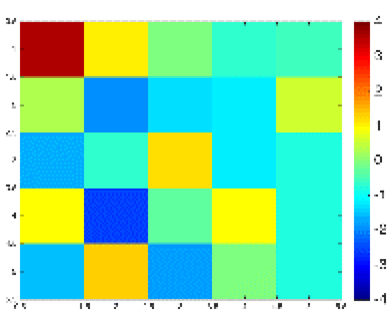


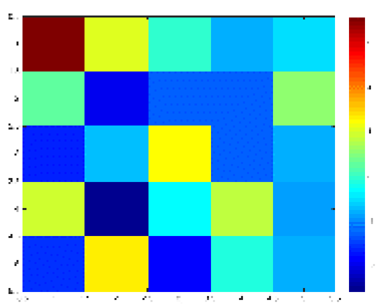
$=$





ℓ $\qquad U_{(l)} \qquad \Sigma_{(l)}$





$B = U_{(l)}\Sigma_{(l)} \qquad B^{new} = U_{(l)}\breve{\Sigma}_{(l)}$

$$\ell = \Omega\left(\frac{\sqrt{m}\|\tilde{Y}_{[t]}\|^2\|\tilde{Y}_{[t]} - \tilde{Y}_{[t]_{(\kappa)}}\|_F^2}{L^2}\right)$$

$$\|V_t - \breve{V}_t\|_F \le \|\tilde{Y}_t\|_F\left(\frac{1}{\sigma_{t_\kappa}}\frac{\sqrt{2}L}{\sqrt{L + 8\|\tilde{Y}_{[t]}\|^2}\sqrt[4]{L^2 + 16\|\tilde{Y}_{[t]}\|^4}} + \frac{\kappa\|\tilde{Y}_{[t]} - \tilde{Y}_{[t]_{(\kappa)}}\|_F}{\breve{\sigma}_{t_\kappa}^2\sqrt{\ell - \kappa}} + \sqrt{\kappa}\max_{i\in[\kappa]}\left|\frac{1}{\tilde{\sigma}_{t_i}} - \frac{1}{\sqrt{\tilde{\sigma}_{t_i}^2 - \tilde{\sigma}_{t_\ell}^2}}\right|\right).$$

Streaming Feature Weighting

Algorithm 2: StreamFeatWeight (streaming update of feature weights at time t)

Input: $Y_t \in \mathbf{R}^{m \times n_t}$, $B_{t-1} \in \mathbf{R}^{m \times \ell}$, and $\alpha \in R$

Output: Feature importance score $\tilde{\mathbf{w}}_t \in \mathbf{R}^m$ and matrix sketch B_t at time t

$C_t \leftarrow [B_{t-1} | Y_t]$

$\tilde{U}_{t_{(\ell)}} \tilde{\Sigma}_{t_{(\ell)}} \tilde{V}^{\top}_{t_{(\ell)}} \leftarrow \textsc{Svd}_\ell(C_t)$
(with $\tilde{\Sigma}_{t_{(\ell)}} = diag(\tilde{\sigma}_{t_1}, \ldots, \tilde{\sigma}_{t_\ell})$)

$\breve{\Sigma}_{t_{(\ell)}} \leftarrow$
$diag\left(\sqrt{\tilde{\sigma}^2_{t_1} - \tilde{\sigma}^2_{t_\ell}}, \sqrt{\tilde{\sigma}^2_{t_2} - \tilde{\sigma}^2_{t_\ell}}, \ldots, \sqrt{\tilde{\sigma}^2_{t_{\ell-1}} - \tilde{\sigma}^2_{t_\ell}}, 0\right)$

$B_t \leftarrow \tilde{U}_{t_{(\ell)}} \breve{\Sigma}_{t_{(\ell)}}$

$\tilde{D}_{t_{(k)}} \leftarrow diag(\breve{\sigma}_{t_1} / (\breve{\sigma}^2_{t_1} + \alpha), \ldots, \breve{\sigma}_{t_k} / (\breve{\sigma}^2_{t_k} + \alpha))$
(where $\breve{\sigma}_{t_i}$ is the ith diagonal element in $\breve{\Sigma}_{t_{(\ell)}}$)

$\tilde{X}_t \leftarrow \tilde{U}_{t_{(k)}} \tilde{D}_{t_{(k)}}$

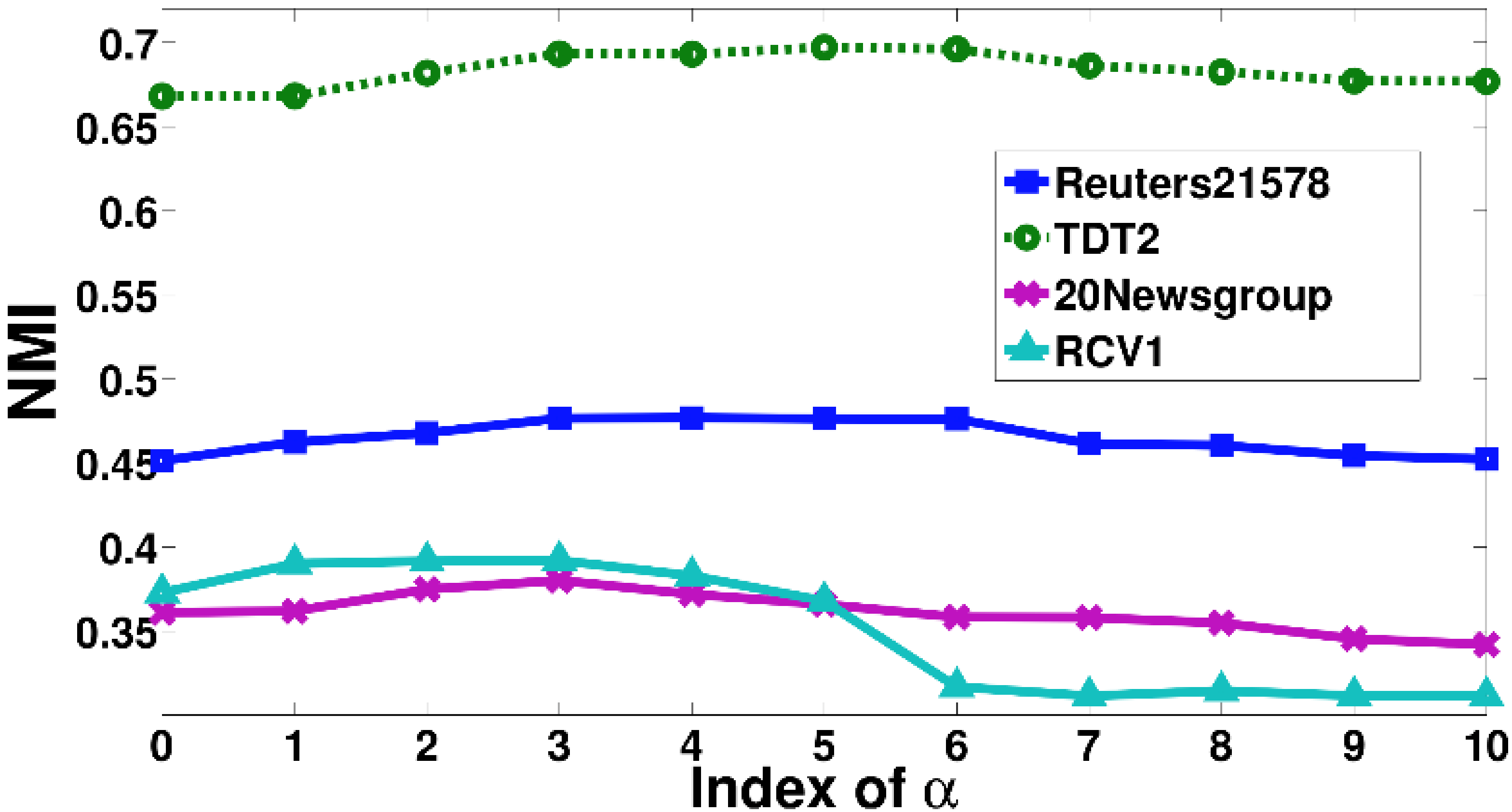
$\forall i \in [m], \tilde{w}_{t_i} \leftarrow \max_{1 \le p \le k} |\tilde{x}_{t_{i,p}}|$, where $\tilde{X}_t = (\tilde{x}_{t_{i,p}})$

Experiments: Setup

- Baselines
  - MCFS (Multi-Cluster Feature Selection)
  - LaplacianScore
  - GFWp2
  - K-Means / Streaming K-Means

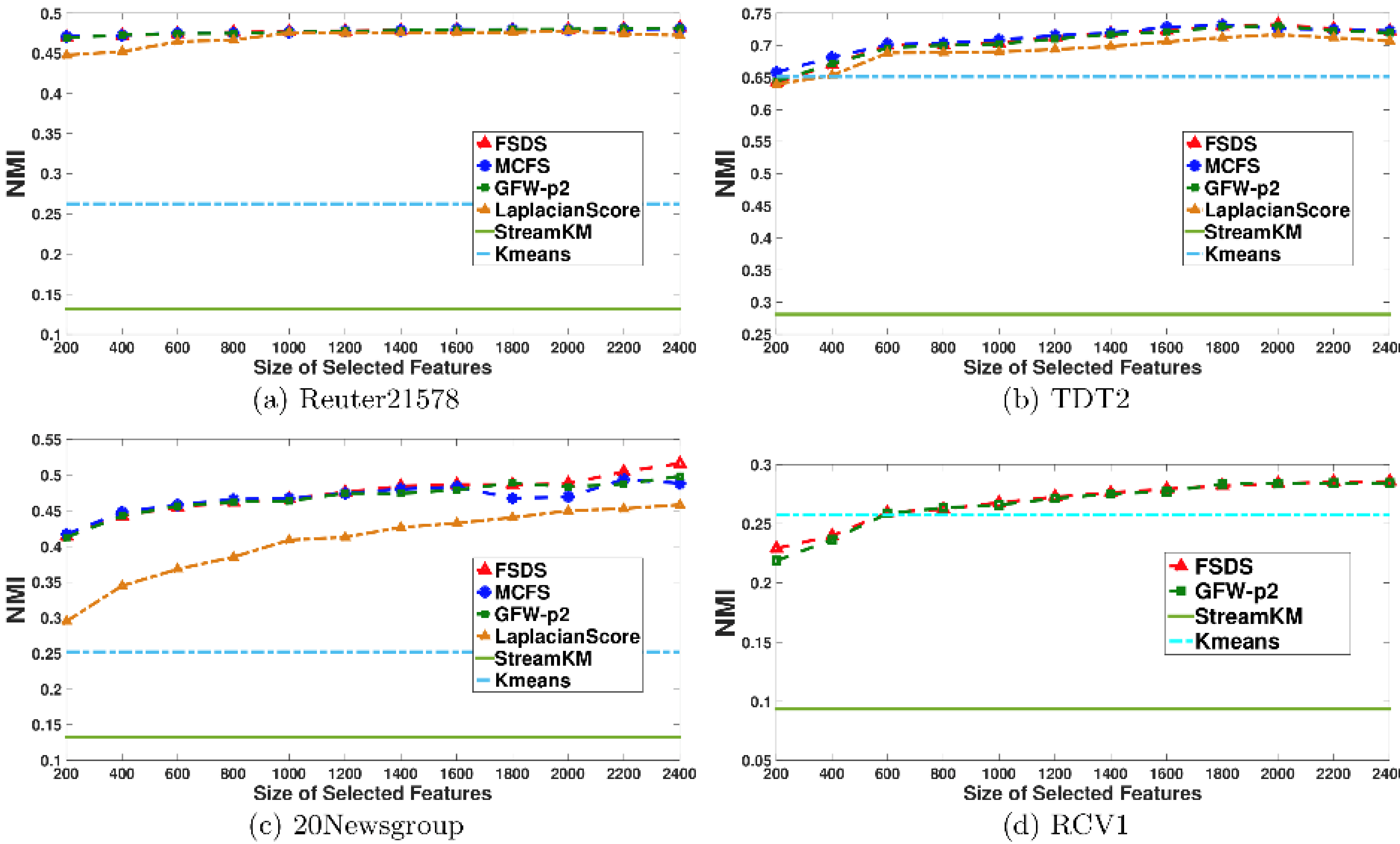
	Dataset	#instances	#features	#clusters
1	Reuter21578	8,293	18,933	65
2	TDT2	9,394	36,771	30
3	20Newsgroup	18,846	26,214	20
4	RCV1	193,844	47,236	103
5	USPS	9,298	256	10
6	MNIST	70,000	784	10
7	Tiny	1,000,000	3,072	75,062

Experiments: Regularization

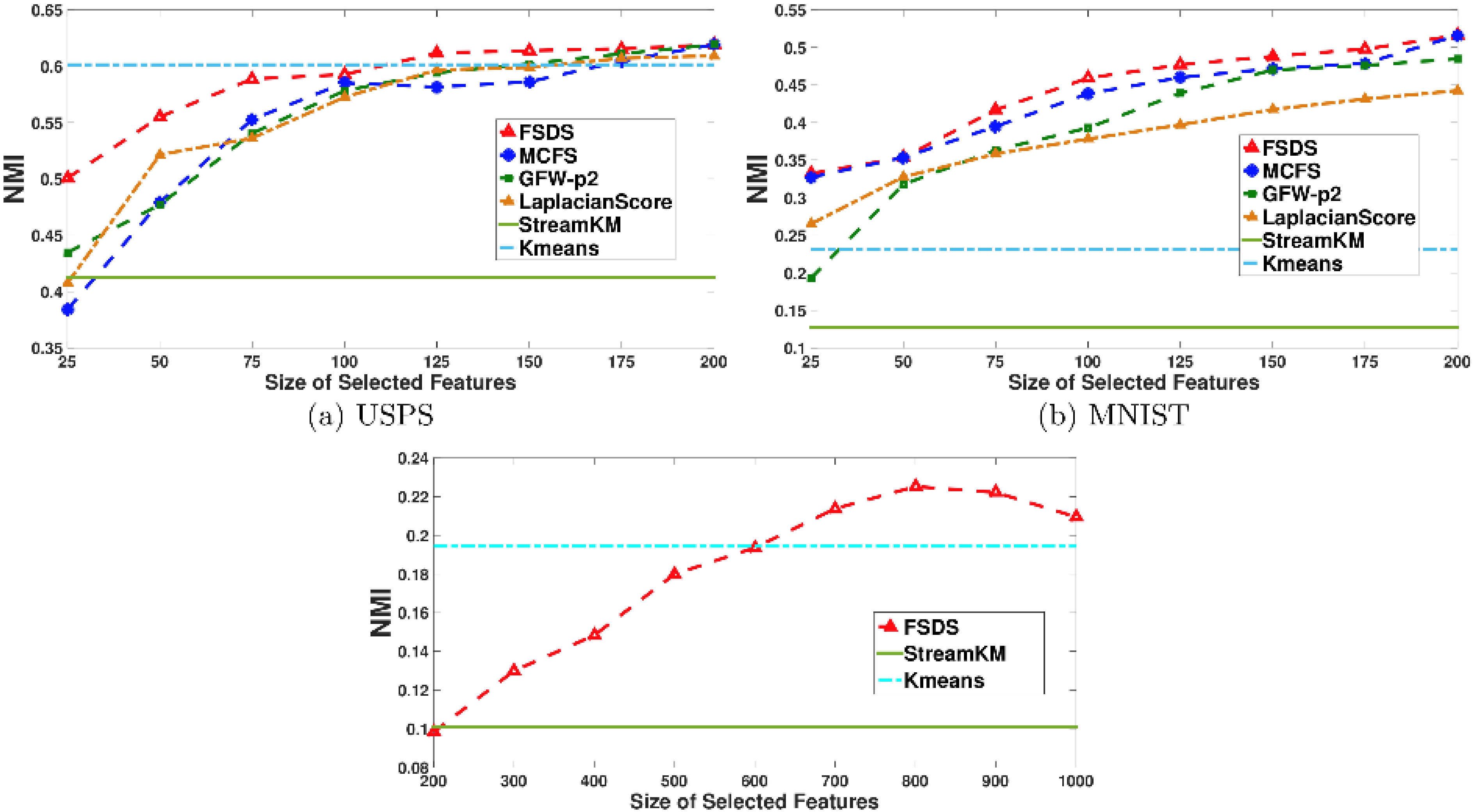


$$\alpha = 2^i \sigma_k$$

Experiments: Effectiveness

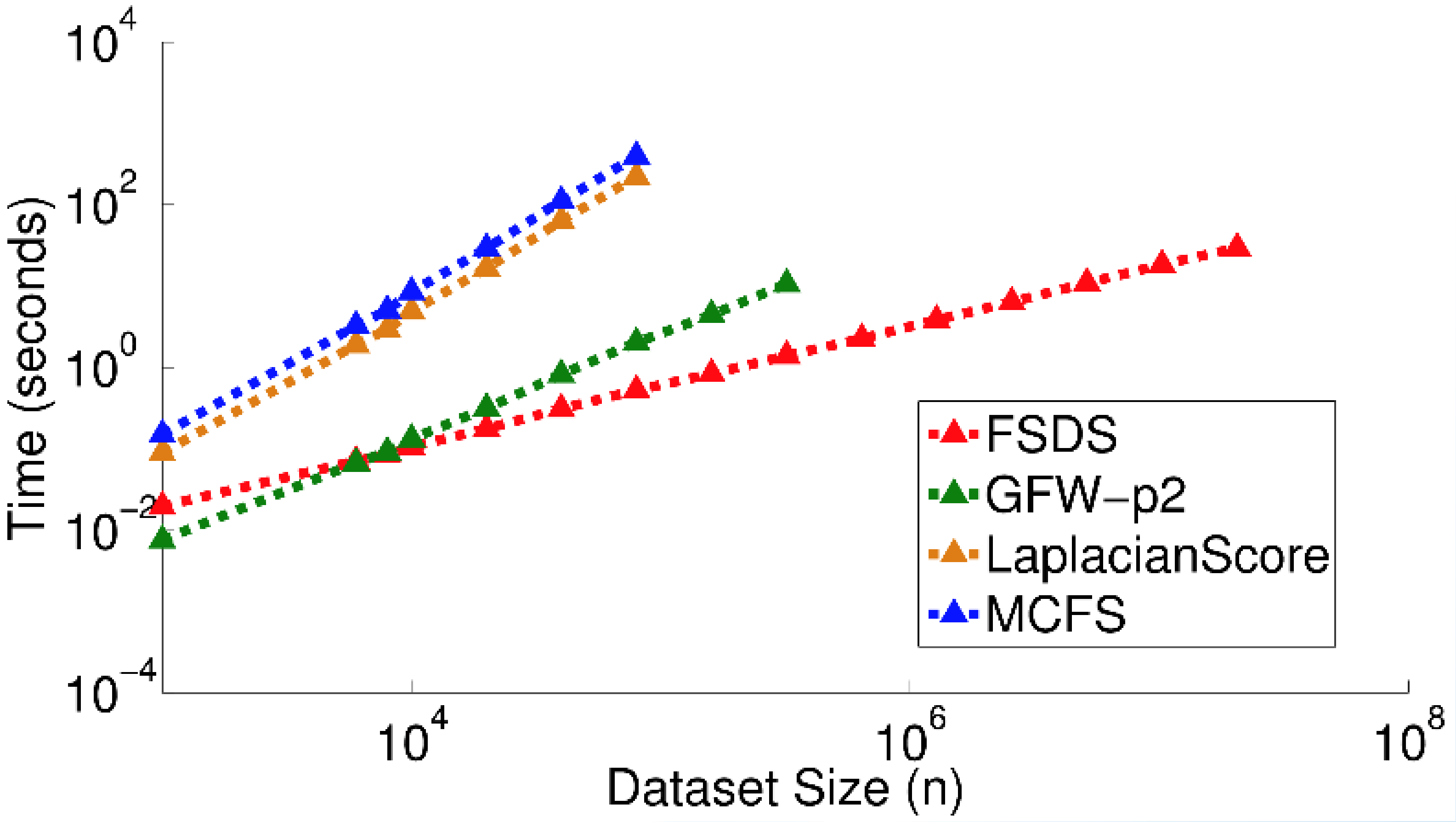


Experiments: Effectiveness

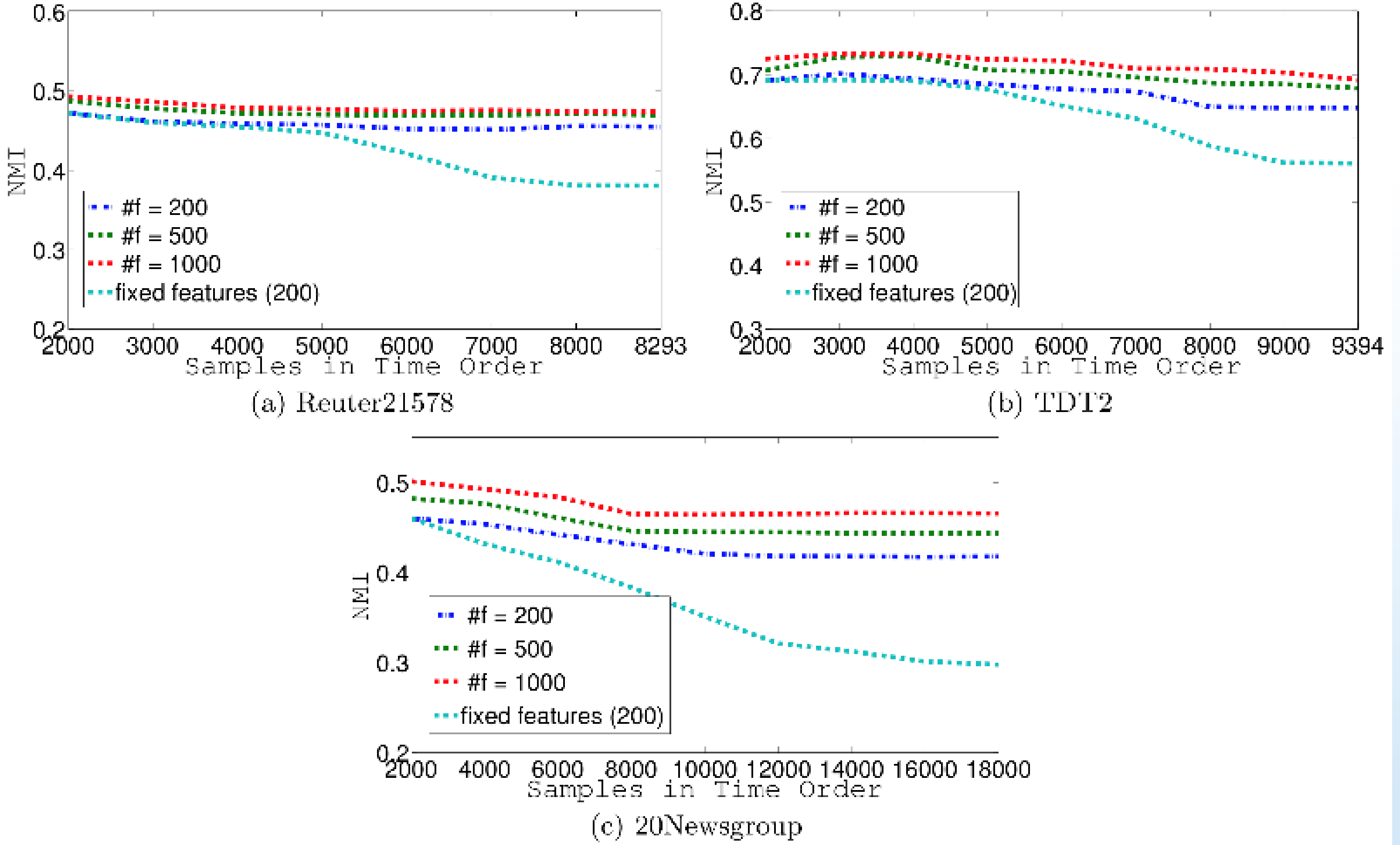


(a) USPS (b) MNIST (c) Tiny

Experiments: Scalability



Experiments: Concept Drift

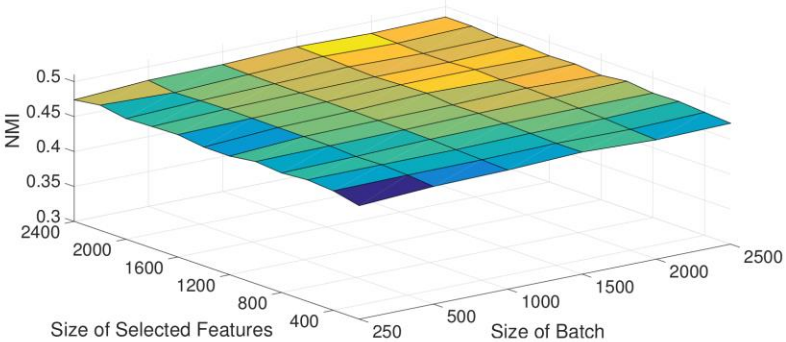


(a) Reuter21578

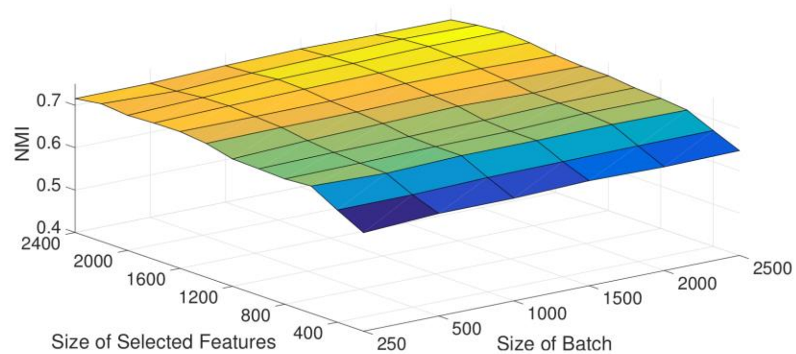
(b) TDT2

(c) 20Newsgroup

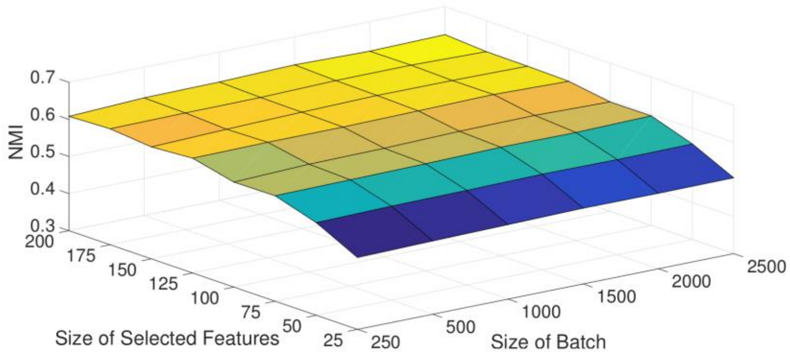
Experiments: Stability



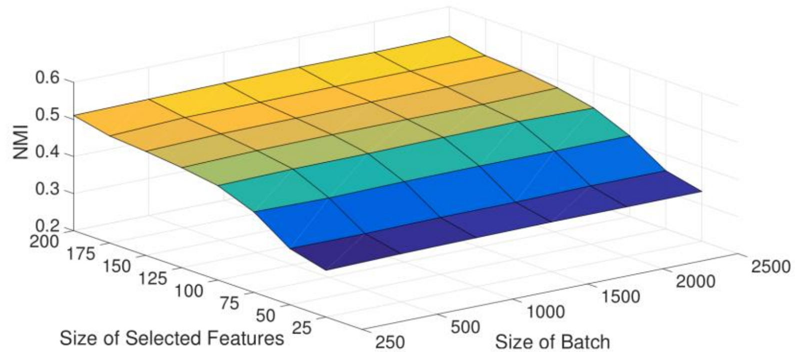
(a) Reuter21578



(b) TDT2

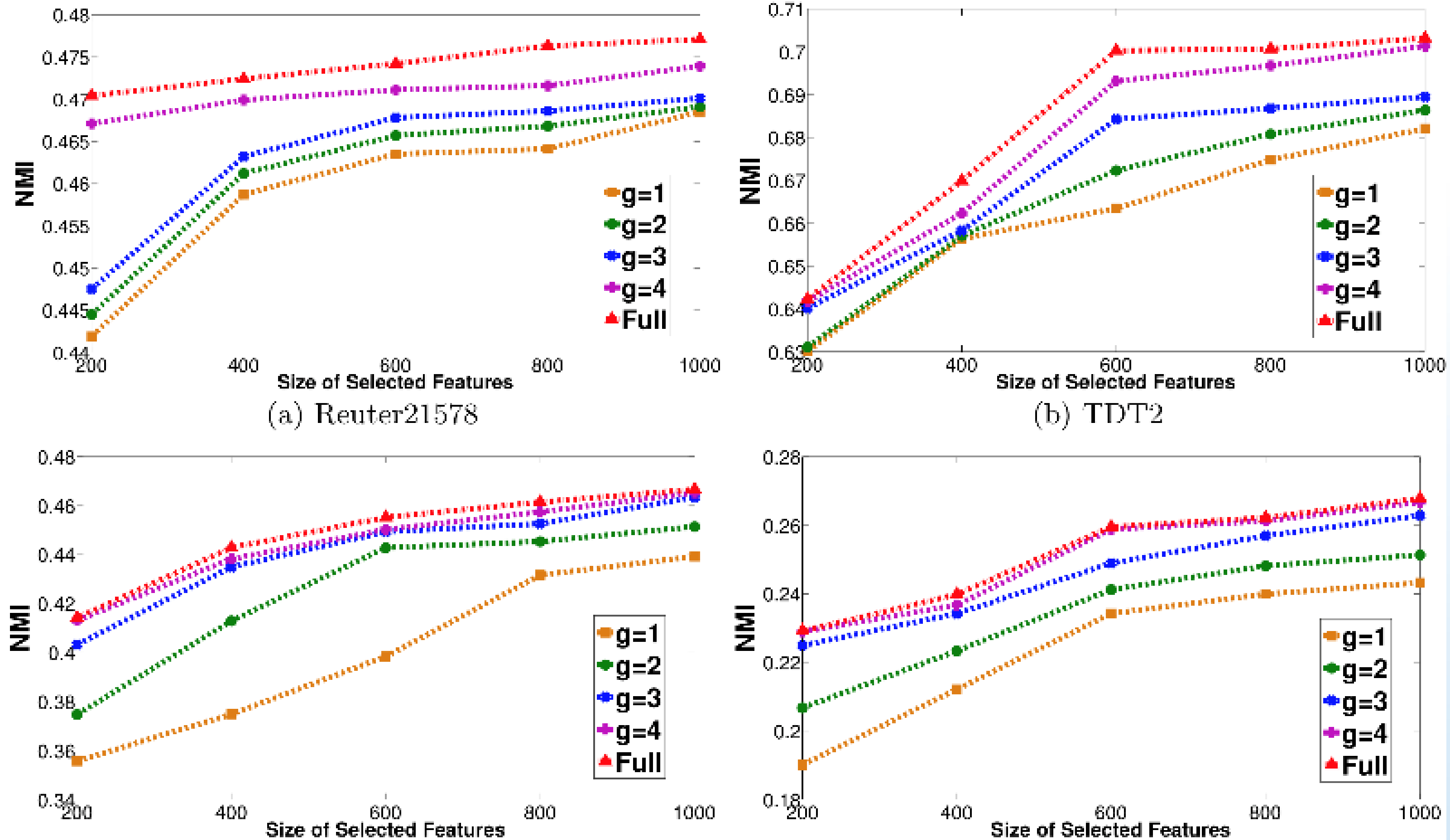


(c) USPS



(d) MNIST

Experiments: Efficient Storage



(a) Reuter21578
(b) TDT2
(c) 20Newsgroup
(d) RCV1

Summary

- Efficient
  - Space: $O(mn_t)$, Time: $O(mn_t\ell)$
- Comparable performance of batch algorithms
  - 97~99% NMI of MCFS on text dataset
- Adapt to concept drift
- Theoretical support

Outline

- Motivation
- Manifold Learning
- Feature Selection
- Spectral Clustering
- Applications

Manifold Learning

Other Manifold Learning Methods:

Diffusion Maps (DM), Local Linear Embedding (LLE), Laplacian Eigenmaps, Multi-demensional Scaling (MDS), …

Spectral Clustering

Algorithm 1: SPECTRALCLUSTERING [6]

Input: Input data $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n] \in R^{m \times n}$ and
$k \in \mathbf{R}$ (number of clusters)

Output: Cluster assignments of n instances

Construct the kernel affinity matrix $W \in R^{n \times n}$, e.g.,

a) $w_{i,j} \leftarrow \exp\left(\frac{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}{2\sigma^2}\right)$ (Gaussian kernel), or

b) $w_{i,j} \leftarrow \frac{\langle \mathbf{y}_i, \mathbf{y}_j \rangle}{\|\mathbf{y}_i\| \|\mathbf{y}_j\|}$ (Cosine kernel)

Degree matrix $D \leftarrow diag(d(\mathbf{y}_1), \ldots, d(\mathbf{y}_n))$
where the degree of a point $\mathbf{y}_i$, $d(\mathbf{y}_i) = \sum_{j=1}^{n} w_{ij}$

Normalized Laplacian $L_{sym} \leftarrow I - D^{-1/2} W D^{-1/2}$

$P \Lambda P^\top \leftarrow \mathrm{EIG}_k(L_{sym})$

$V \leftarrow$ normalized P with unit L_2 row norms

Cluster the rows of V into k clusters (using K-means)

Affinity Matrix Construction

- Cosine kernel

$$SVD(Y_{[t]}) = SVD([Y_1 \,|\, Y_2 \,|\cdots\|\, Y_t]) = U_c \Sigma_c V_c^T$$

$$EIG(W_{COS}) = EIG(Y_{[t]}^T Y_{[t]}) = V_c \Sigma_c^2 V_c^T$$

- Gaussian kernel approximation

$$w_{i,j} \leftarrow \exp\left(\frac{-\|\mathbf{y}_i - \mathbf{y}_j\|^2}{2\sigma^2}\right)$$

1) Draw d i.i.d. samples $\omega(1), \ldots, \omega(d)$ from $p(\omega)$ where $\omega \sim \frac{1}{\sigma^2}\mathcal{N}(0,1))$ and $p(\cdot)$ is the fast Fourier transform;
2) Draw d i.i.d. samples (offsets) $\mathbf{b}_1, \ldots, \mathbf{b}_d$ from uniform distribution on $[0, \pi]$;
3) Compute the projected data where $h(\mathbf{x}) = \cos(\omega^\top \mathbf{x} + \mathbf{b})$;

$$SVD(H_{[t]}) = SVD([h(Y_1) \,|\, h(Y_2) \,|\cdots\|\, h(Y_t)]) = U_g \Sigma_g V_g^T$$

$$EIG(W_{gau}) \approx EIG(H_{[t]}^T H_{[t]}) = V_g \Sigma_g^2 V_g^T$$

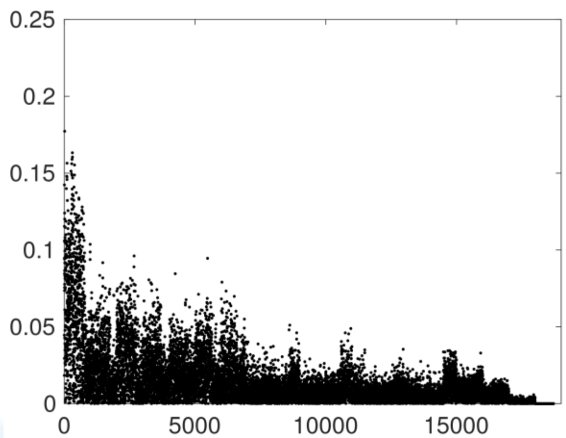
Degree Matrix Approximation

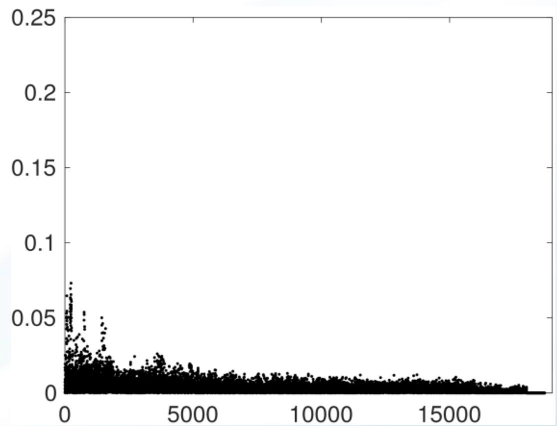
- Degree

$$d_t(y) = \sum_{z \in Y_{[t]}} y^T z = y^T \sum_{z \in Y_{[t]}} z = y^T s_t$$

  - Can construct degree matrix without affinity matrix
  - As *t* increase, degree is also continuously increased

$$\tilde{d}_t(y) = \frac{y^T s_t}{\| s_t \|} = y^T c_t \qquad \tilde{d}_{t+1}(\mathbf{y}) - \tilde{d}_t(\mathbf{y}) \leq \sqrt{n_{t+1}/(n_{[t+1]} - n_{t+1})}$$





Normalized Laplacian

- Symmetric normalized Laplacian

$$L_{SYM} = I - W_{SYM} = I - D^{-0.5} W D^{-0.5}$$

- Lemma 3.2
  - k largest eigenvectors of W_{SYM} is equal to k smallest eigenvectors L_{SYM}

$$\begin{aligned} (L_{sym} - \lambda I)\mathbf{v} &= \mathbf{0} \\ (I - W_{sym} - \lambda I)\mathbf{v} &= \mathbf{0} \\ (-W_{sym} + I - \lambda I)\mathbf{v} &= \mathbf{0} \\ (W_{sym} - I + \lambda I)\mathbf{v} &= \mathbf{0} \\ (W_{sym} - (1-\lambda)I)\mathbf{v} &= \mathbf{0} \\ W_{sym}\mathbf{v} &= (1-\lambda)\mathbf{v} \end{aligned}$$

$$SVD(\tilde{Y}_{[t]}) = SVD(Y_{[t]} D^{-0.5})$$

Manifold Alignment

- Concept drift
  - The low rank basis $\tilde{U}_{t_{(k)}}$ is changing over time due to concept drift
  - Need to update learned centroids according to basis shift

$$R_t = \tilde{U}^T_{t-1_{(k)}} \tilde{U}_{t_{(k)}}$$

Streaming Spectral Clustering

Algorithm 4: StrmSC

Input: Data stream $\mathcal{S}$, $\ell \in \mathbf{R}$, $\kappa\,(\leq \ell) \in \mathbf{R}$, $\rho \in \mathbf{R}$, $k \in \mathbf{R}$, and $\beta \in \mathbf{R}$

Output: Cluster assignments of all instances in $\mathcal{S}$

```
Initialize f = 1/(k(1 + log n)) and an empty set C
s_0 ← all zeros vector ∈ R^m
B_0 ← all zeros matrix ∈ R^{m×ℓ}
Ũ_0 ← Ũ_1 (to skip the first rotation)
while stream S not finished do
    let Y_t ∈ R^{m×n_t} be the batch with timestamp t in stream S
    [B_t, Ũ_{t(κ)}, V̆_t, s_t] ← StrmEmb (Y_t, B_{t−1}, s_{t−1}, κ)
    V̂_t ← normalized V̆_t with unit L_2 row norms
    R_t ← Ũ^⊤_{t−1(ℓ)} Ũ_{t(ℓ)}
    Replace each z ∈ C by R_t z
    for each column x in V̂_t do
        [C, f] ← StrmKMStep (x, C, ρ, β, f)
    end
end
Run batch K-means on weighted points C to form k clusters
```

Streaming Spectral Clustering

Algorithm 5: StrmEmb

Input: $Y_t \in \mathbf{R}^{m \times n_t}$, $B_{t-1} \in \mathbf{R}^{m \times \ell}$, $\mathbf{s}_{t-1} \in \mathbf{R}^m$, and $\kappa \in \mathbf{R}$

Output: B_t, $\tilde{U}_{t_{(\ell)}}$, $\breve{V}_t$, $\mathbf{s}_t$

$\mathbf{s}_t \leftarrow \mathbf{s}_{t-1} + \mathrm{row\_sum}(Y_t)$

$\mathbf{c}_t \leftarrow \mathbf{s}_t / \|\mathbf{s}_t\|$

$\tilde{D}_t \leftarrow diag(\langle \mathbf{y}_1, \mathbf{c}_t \rangle, \ldots, \langle \mathbf{y}_{n_t}, \mathbf{c}_t \rangle)$ where $Y_t = [\mathbf{y}_1, \ldots, \mathbf{y}_{n_t}]$

$\tilde{Y}_t \leftarrow Y_t \tilde{D}_t^{-1/2}$

$C_t \leftarrow [B_{t-1}, \tilde{Y}_t]$

$\tilde{U}_{t_{(\ell)}} \tilde{\Sigma}_{t_{(\ell)}} \tilde{V}_{t_{(\ell)}}^\top \leftarrow \mathrm{SVD}_\ell(C_t)$

$\breve{\Sigma}_t \leftarrow diag\left(\sqrt{\tilde{\sigma}_{t_1}^2 - \tilde{\sigma}_{t_\ell}^2}, \ldots, \sqrt{\tilde{\sigma}_{t_{\ell-1}}^2 - \tilde{\sigma}_{t_\ell}^2}, 0\right)$ where $\tilde{\Sigma}_{t_{(\ell)}} = diag(\tilde{\sigma}_{t_1}, \ldots, \tilde{\sigma}_{t_\ell})$

$B_t \leftarrow \tilde{U}_{t_{(\ell)}} \breve{\Sigma}_{t_{(\ell)}}$

$\tilde{U}_{t_{(\kappa)}} \leftarrow [\mathbf{u}_1, \ldots, \mathbf{u}_\kappa]$ where $\tilde{U}_{t_{(\ell)}} = [\mathbf{u}_1, \ldots, \mathbf{u}_\ell]$

$\tilde{\Sigma}_{t_{(\kappa)}} \leftarrow diag(\tilde{\sigma}_{t_1}, \ldots, \tilde{\sigma}_{t_\kappa})$

$\breve{V}_t^\top \leftarrow \tilde{\Sigma}_{t_{(\kappa)}}^{-1} \tilde{U}_{t_{(\kappa)}} \tilde{Y}_t$

Experiments: Setup

- Baselines
  - K-Means (KM)
  - Symmetric normalized spectral clustering (NJW)
  - Streaming K-Means (SKM)
  - BIRCH, HDDStr
- SSC-1 (streaming manifold only, batch KM)
- SSC-2 (full version, SKM)

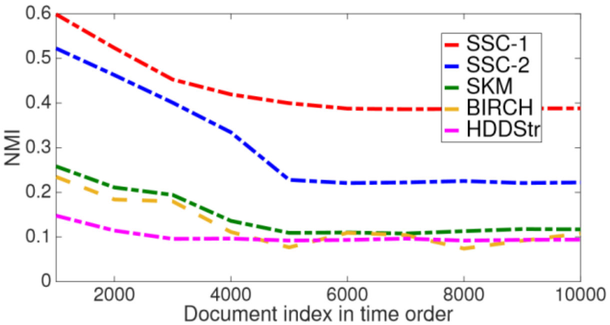
	Dataset	#instances	#features	#clusters
1	sector	9,619	55,197	105
2	ohscal	11,162	11,465	10
3	20NG	18,707	29,476	20
4	RCV1	193,844	47,236	103
5	MNIST	70,000	784	10
6	Tiny	1,000,000	3,072	75,062
7	SenVec (SensIT Vehicle)	98,528	100	3
8	jester	73,421	100	86

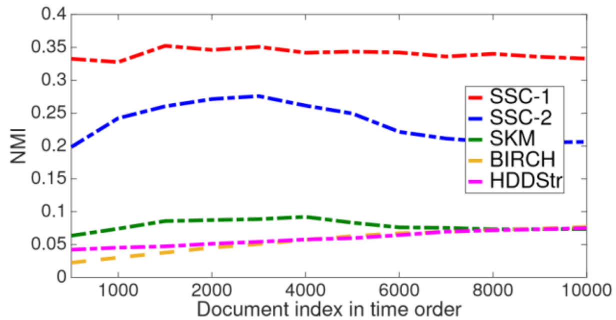
Experiments: NMI/Purity

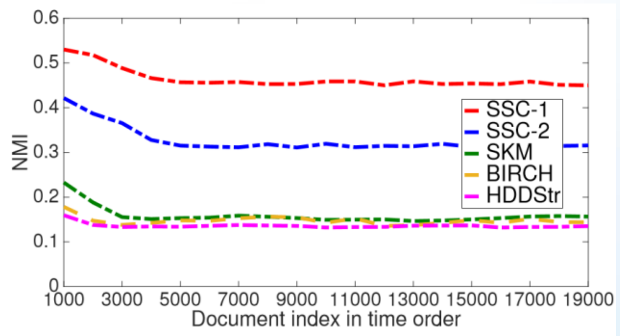
	NJW	SSC-1	KM	BIRCH	SKM	HDDStr	SSC-2
sector	0.4039(0.003) 0.9794(0.001)	0.3961(0.003) 0.9790(0.001)	0.3554(0.010) 0.9576(0.005)	0.0872(0.013) 0.5372(0.006)	0.0921(0.011) 0.6023(0.006)	0.0821(0.037) 0.6117(0.019)	0.2361(0.008) 0.8616(0.002)
ohscal	0.3408(0.018) 0.8897(0.004)	0.3217(0.016) 0.8833(0.005)	0.3041(0.022) 0.8299(0.007)	0.0876(0.048) 0.1902(0.174)	0.1060(0.027) 0.2405(0.058)	0.0781(0.052) 0.3193(0.181)	0.2273(0.018) 0.8584(0.009)
20NG	0.4836(0.008) 0.9643(0.002)	0.4699(0.009) 0.9532(0.004)	0.4686(0.016) 0.8976(0.007)	0.1342(0.037) 0.6723(0.046)	0.1601(0.041) 0.6402(0.048)	0.1232(0.067) 0.7011(0.081)	0.3512(0.010) 0.9412(0.002)
RCV1	[52] 0.2875(0.001) —(—)	0.2733(0.003) 0.9432(0.001)	0.2577(0.009) 0.8721(0.006)	0.1332(0.014) 0.5011(0.009)	0.1222(0.012) 0.5274(0.009)	0.1065(0.037) 0.5567(0.014)	0.1733(0.005) 0.7657(0.003)
MNIST	0.4965(0.011) 0.8868(0.003)	0.3792(0.004) 0.8788(0.001)	0.4873(0.022) 0.8809(0.006)	0.1272(0.053) 0.4292(0.197)	0.1289(0.079) 0.4059(0.206)	0.1431(0.102) 0.4566(0.258)	0.1953(0.003) 0.8587(0.002)
jester	0.5099(0.003) 0.9444(0.001)	0.4723(0.005) 0.9442(0.001)	0.5322(0.002) 0.9446(0.001)	0.1482(0.003) 0.7428(0.009)	0.1501(0.002) 0.7311(0.009)	0.1438(0.006) 0.6857(0.019)	0.2601(0.012) 0.9296(0.007)
SenVec	0.3007(0.001) 0.6769(0.001)	0.2810(0.001) 0.6921(0.001)	0.2831(0.016) 0.6944(0.006)	0.0989(0.071) 0.4639(0.118)	0.0975(0.067) 0.4606(0.102)	0.0864(0.097) 0.5367(0.143)	0.1221(0.021) 0.6261(0.062)
Tiny	—(—) —(—)	0.2001(0.001) 0.9569(0.002)	0.1744(0.001) 0.8793(0.003)	0.0817(0.003) 0.7011(0.003)	0.0721(0.005) 0.7314(0.003)	0.0538(0.012) 0.6593(0.009)	0.1703(0.001) 0.9182(0.001)

Experiments: Concept-drift

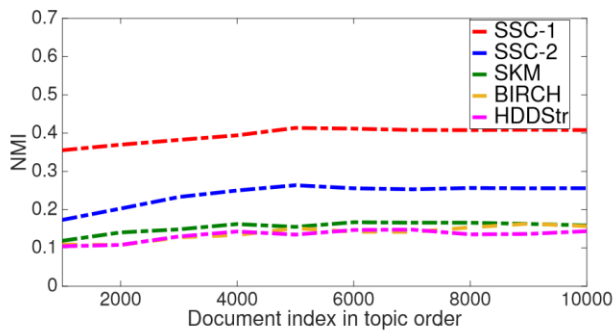
- Ordered by time

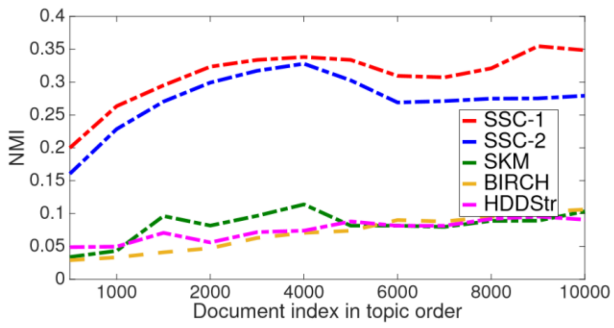


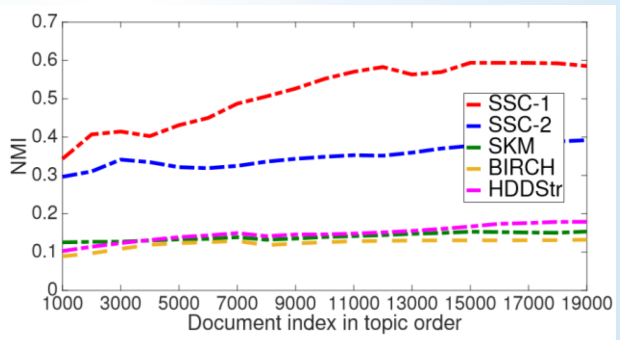




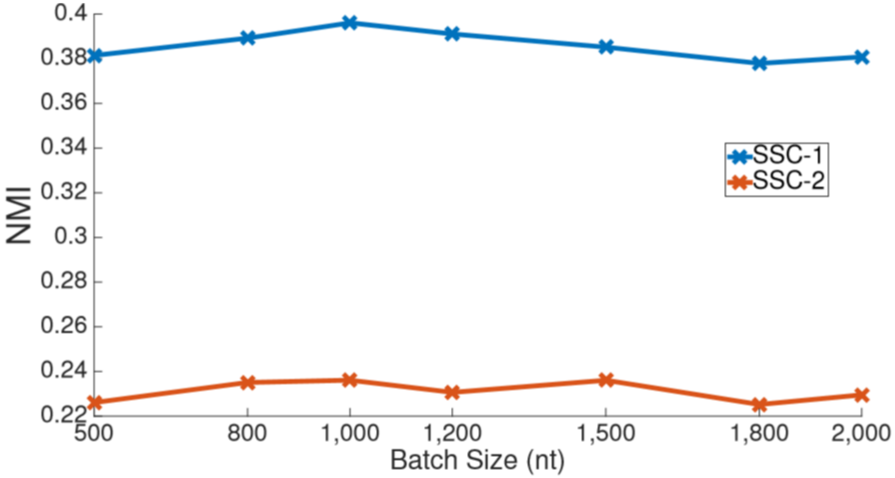
- Ordered by topic



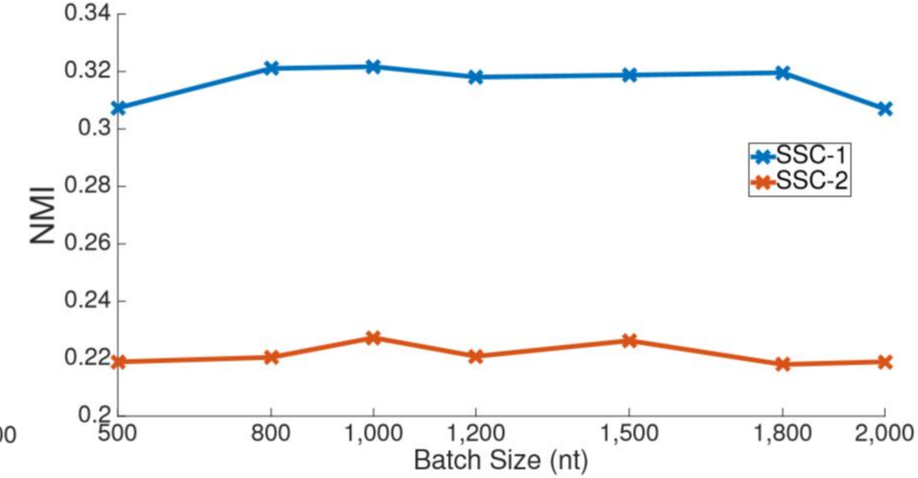




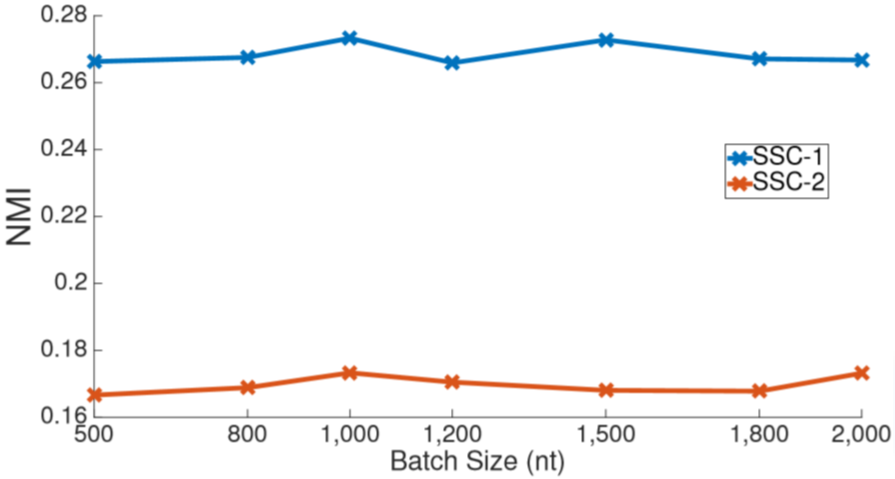
Experiments: Batch Size Stability



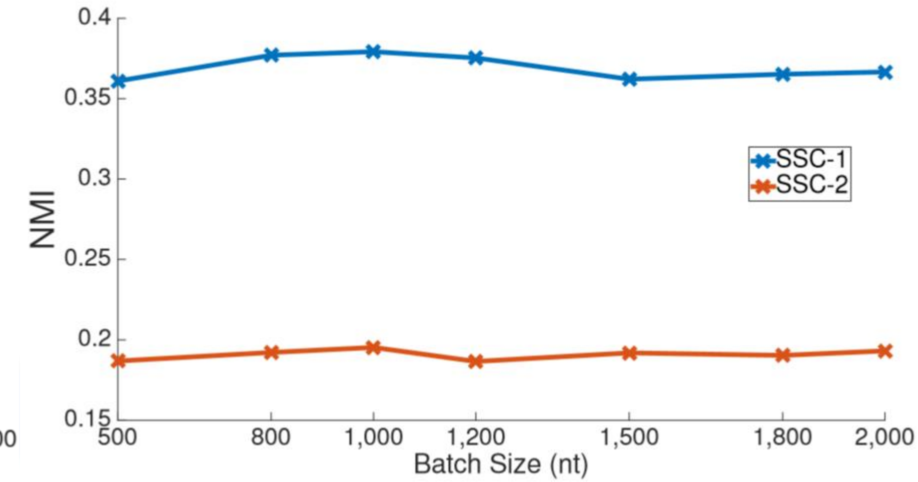
(a) sector



(b) ohscal

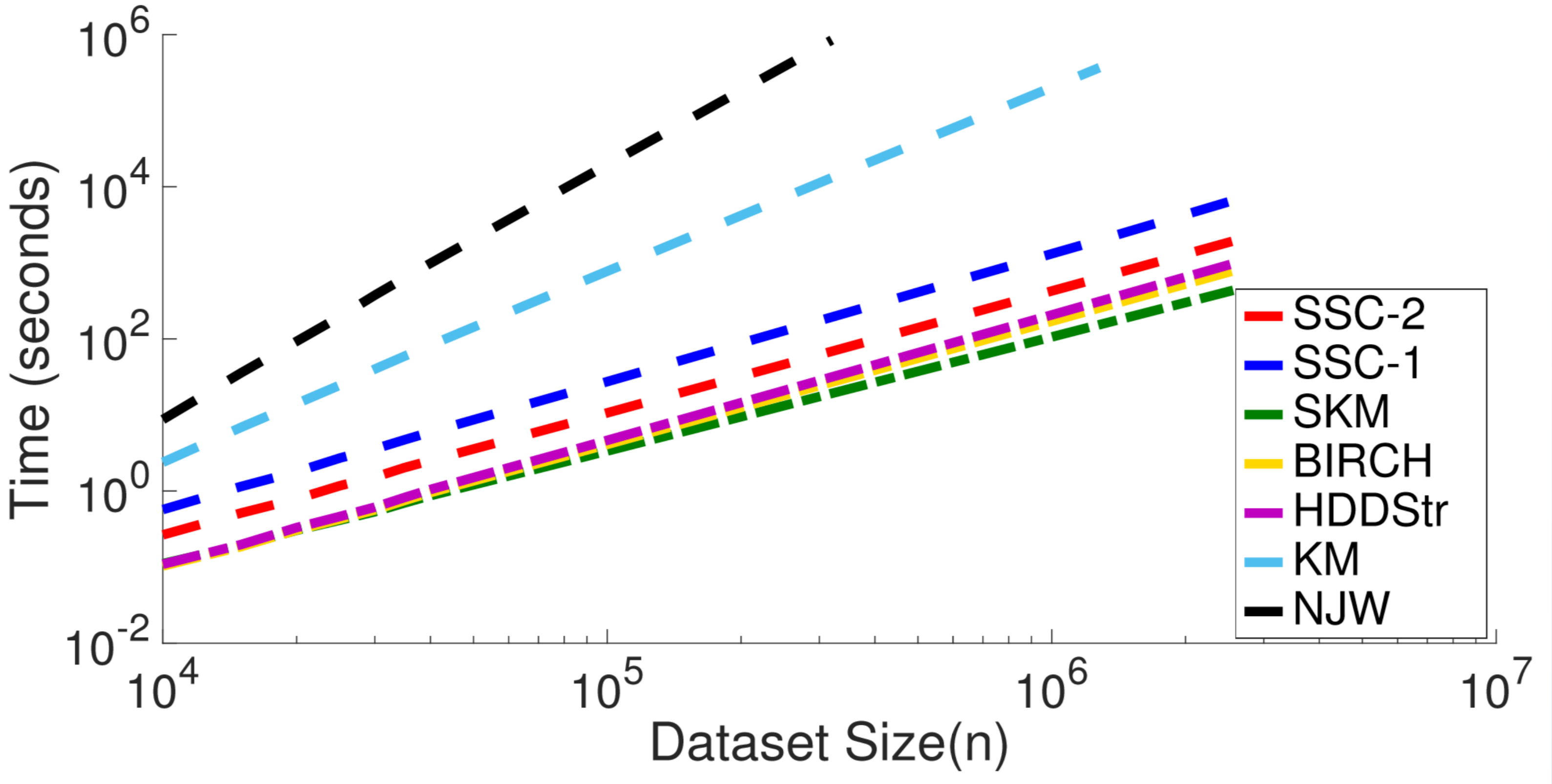


(d) RCV1



(e) MNIST

Experiments: Scalability



Summary

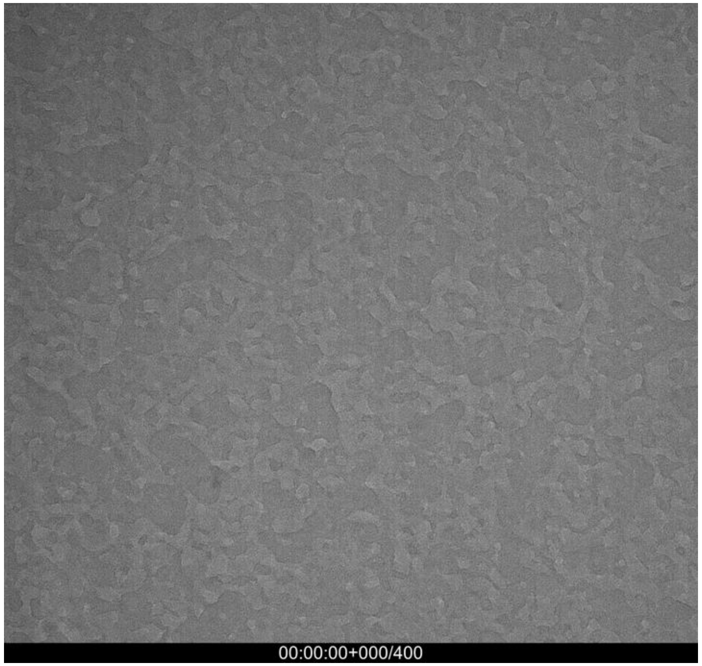
- Efficient
  - Space: $O(mn_t + m\ell)$, Time: $O(\max\{mn_t\ell, m\ell^2\})$
- Comparable performance of batch algorithms
  - SSC-1: 92% NMI of NJW
  - SSC-2: two folds better than streaming algorithms
- Adapt to concept drift
- Theoretical support

Outline

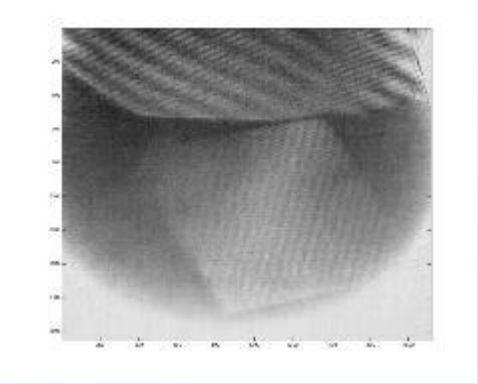
- Motivation
- Manifold Learning
- Feature Selection
- Spectral Clustering
- Applications

Streaming ML on Image Analysis

- Particle growth detection and tracking



- Nanotube Angle calculation



Manifold Learning with Spatio-Temporal Data Analysis

- Spatio-Temporal Analysis
  - Common on scientific simulation / sensor network data
  - Typically 100TB or more simulation output with Peta-scale simulation
    - DNS, LES, or any spatio-temporal simulations
  - Order of magnitude bigger with exascale machines
  - IoT and other application need to scale up
  - Working on simulation parameterization, emulation, outlier detection on spatio-temporal data

Q&As