Differentiable Unsupervised Feature Selection based on a Gated Laplacian

Ofir Lindenbaum, Uri Shaham, Erez Peterfreund, Jonathan Svirsky,
Nicolas Casey, Yuval Kluger

November 2021

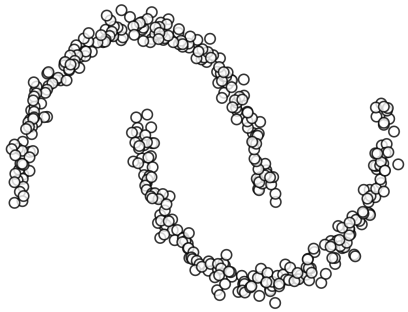
Unsupervised Learning

Find structures in dataset $\tilde{\boldsymbol{X}} \in \mathbb{R}^{N \times D_c}$ (**no label information**)

Unsupervised Learning

Find structures in dataset $\tilde{\boldsymbol{X}} \in \mathbb{R}^{N \times D_c}$ (**no label information**)

- Clustering



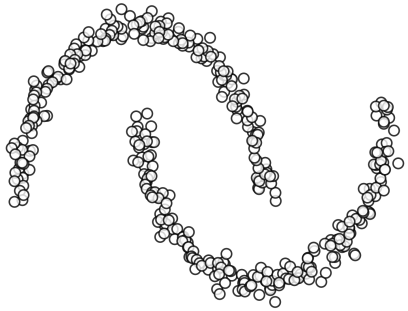
Two-moons

Find subsets $C_1, ..., C_k \subset \tilde{\boldsymbol{X}}$ so that points within C_i are "similar"

Unsupervised Learning

Find structures in dataset $\tilde{\boldsymbol{X}} \in \mathbb{R}^{N \times D_c}$ (**no label information**)

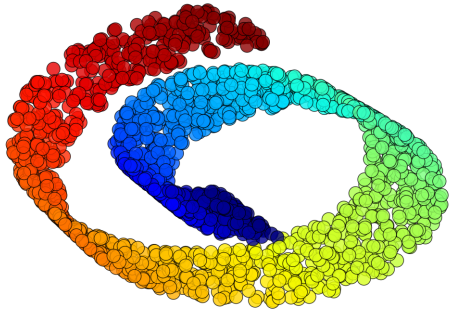
- Clustering



Two-moons

Find subsets $C_1, ..., C_k \subset \tilde{\boldsymbol{X}}$ so that points within C_i are "similar"

- Manifold Learning



Swiss-roll

Embed $\tilde{\boldsymbol{X}}$ into a lower dimension without loosing much information

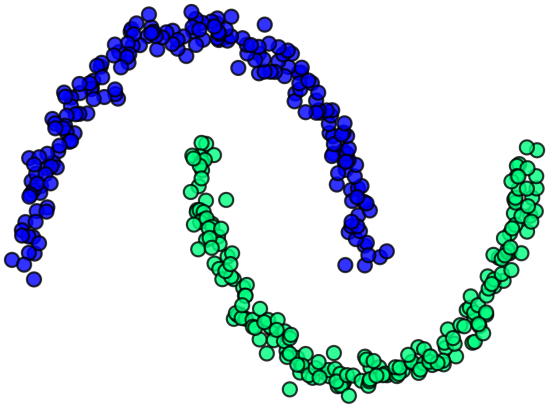
Unsupervised Learning: Nuisance Variables

Consider an **informative** data $\tilde{\boldsymbol{X}}$ concatenated with nuisance variables $\boldsymbol{X} = \left[\tilde{\boldsymbol{X}}|\boldsymbol{\xi}_1|...|\boldsymbol{\xi}_{D_n}\right]_{N\times(D_c+D_n)}$, where $\xi_{i,j} \sim N(0,1)$ and $D = D_c + D_n$

Unsupervised Learning: Nuisance Variables

Consider an **informative** data $\tilde{\boldsymbol{X}}$ concatenated with nuisance variables $\boldsymbol{X} = \left[\tilde{\boldsymbol{X}}|\boldsymbol{\xi}_1|...|\boldsymbol{\xi}_{D_n}\right]_{N\times(D_c+D_n)}$, where $\xi_{i,j} \sim N(0,1)$ and $D = D_c + D_n$

- Spectral Clustering (Ng. et al.)

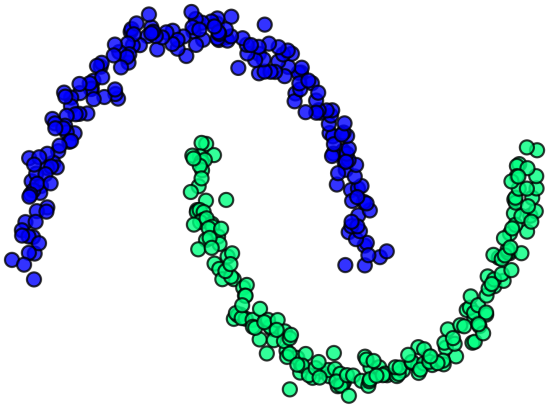


Cluster assignments

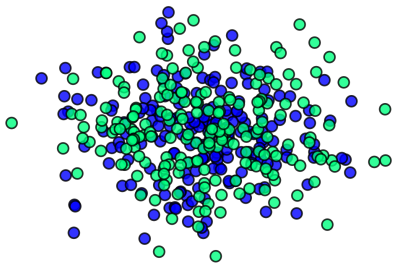
Unsupervised Learning: Nuisance Variables

Consider an **informative** data $\tilde{\boldsymbol{X}}$ concatenated with nuisance variables $\boldsymbol{X} = \left[\tilde{\boldsymbol{X}}|\boldsymbol{\xi}_1|...|\boldsymbol{\xi}_{D_n}\right]_{N\times(D_c+D_n)}$, where $\xi_{i,j} \sim N(0,1)$ and $D = D_c + D_n$

- Spectral Clustering (Ng. et al.)



Cluster assignments

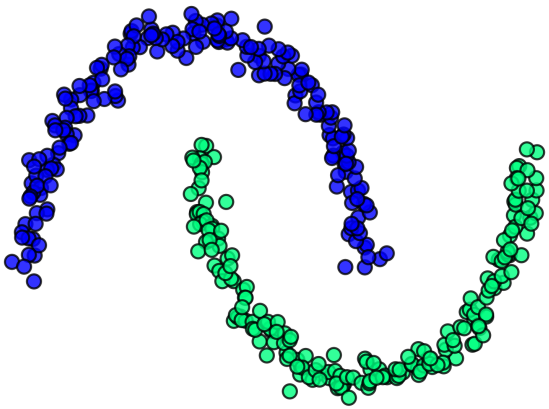


Nuisance variables

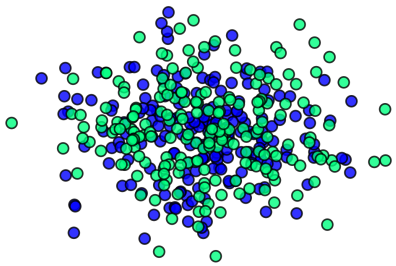
Unsupervised Learning: Nuisance Variables

Consider an **informative** data $\tilde{\boldsymbol{X}}$ concatenated with nuisance variables $\boldsymbol{X} = \left[\tilde{\boldsymbol{X}}|\boldsymbol{\xi}_1|...|\boldsymbol{\xi}_{D_n}\right]_{N\times(D_c+D_n)}$, where $\xi_{i,j} \sim N(0,1)$ and $D = D_c + D_n$

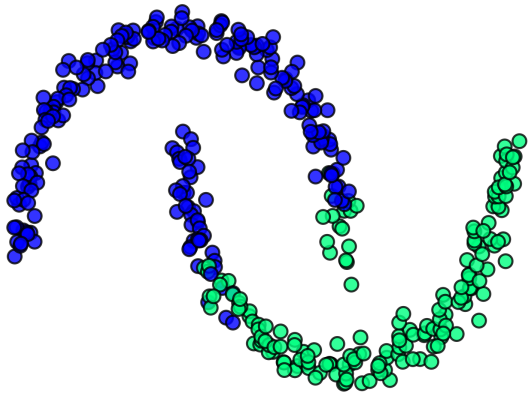
- Spectral Clustering (Ng. et al.)



Cluster assignments



Nuisance variables

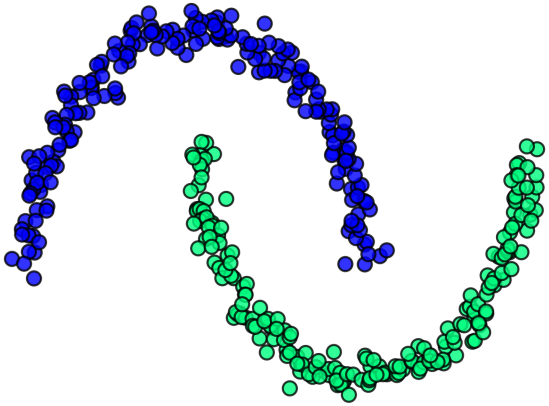


Cluster assignments

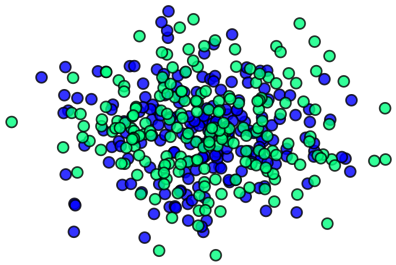
Unsupervised Learning: Nuisance Variables

Consider an **informative** data $\tilde{\boldsymbol{X}}$ concatenated with nuisance variables $\boldsymbol{X} = \left[\tilde{\boldsymbol{X}}|\boldsymbol{\xi}_1|...|\boldsymbol{\xi}_{D_n}\right]_{N\times(D_c+D_n)}$, where $\xi_{i,j} \sim N(0,1)$ and $D = D_c + D_n$

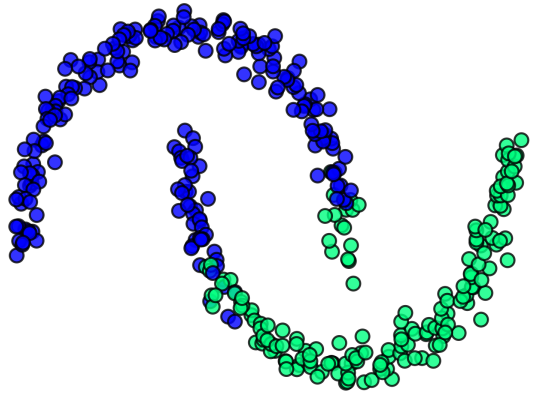
- Spectral Clustering (Ng. et al.)



Cluster assignments

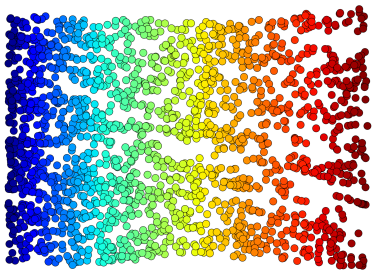


Nuisance variables



Cluster assignments

- ISOMAP (Tenenbaum et al.)

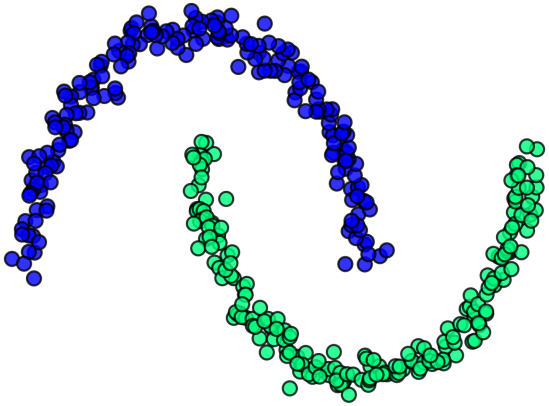


Embedding

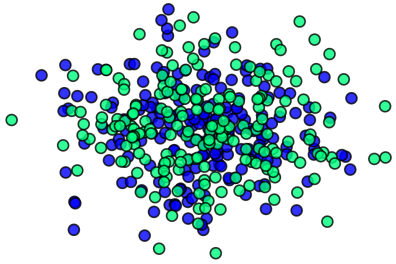
Unsupervised Learning: Nuisance Variables

Consider an **informative** data $\tilde{\boldsymbol{X}}$ concatenated with nuisance variables $\boldsymbol{X} = \left[\tilde{\boldsymbol{X}}|\boldsymbol{\xi}_1|...|\boldsymbol{\xi}_{D_n}\right]_{N\times(D_c+D_n)}$, where $\xi_{i,j} \sim N(0,1)$ and $D = D_c + D_n$

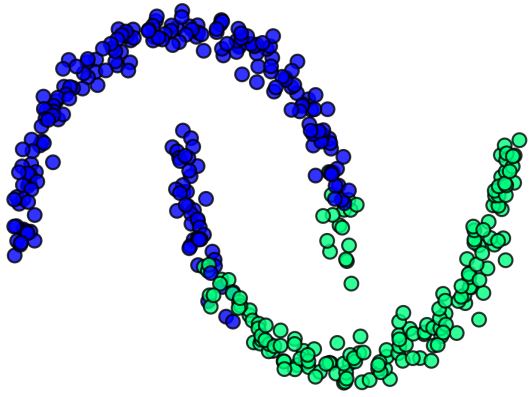
- Spectral Clustering (Ng. et al.)



Cluster assignments

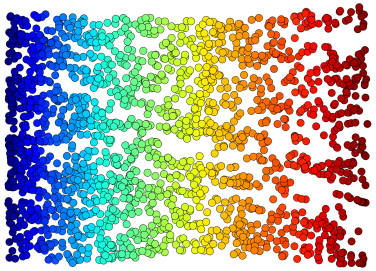


Nuisance variables

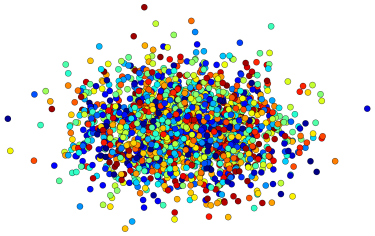


Cluster assignments

- ISOMAP (Tenenbaum et al.)



Embedding

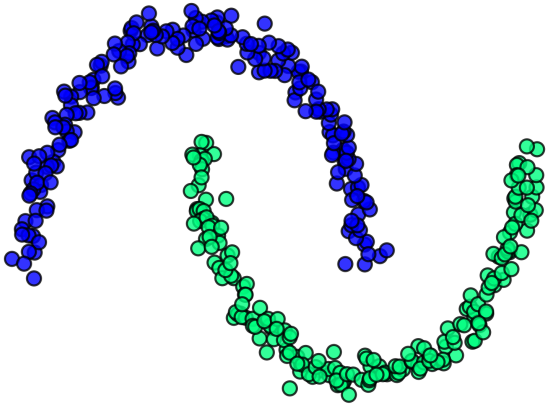


Nuisance variables

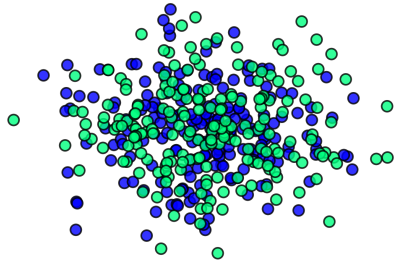
Unsupervised Learning: Nuisance Variables

Consider an **informative** data $\tilde{\boldsymbol{X}}$ concatenated with nuisance variables $\boldsymbol{X} = \left[\tilde{\boldsymbol{X}}|\boldsymbol{\xi}_1|...|\boldsymbol{\xi}_{D_n}\right]_{N\times(D_c+D_n)}$, where $\xi_{i,j} \sim N(0,1)$ and $D = D_c + D_n$

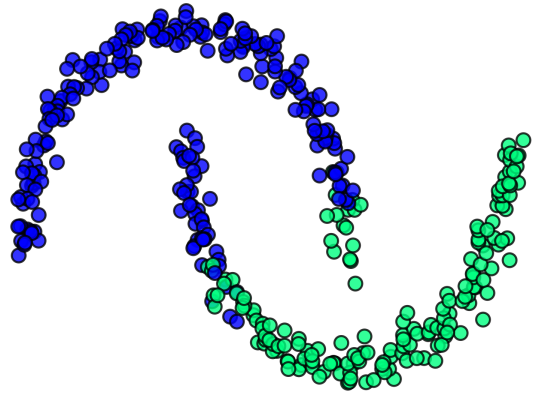
- Spectral Clustering (Ng. et al.)



Cluster assignments

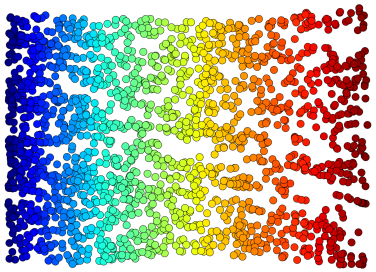


Nuisance variables

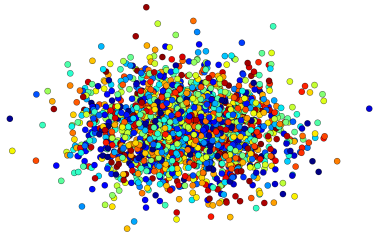


Cluster assignments

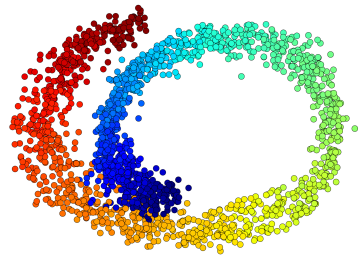
- ISOMAP (Tenenbaum et al.)



Embedding



Nuisance variables



Embedding

Unsupervised Feature Selection: Motivation

Goal (informal)

Find a **subset** of *informative* variables $\mathcal{S} \subset \{1, ..., D\}$ to improve clustering or manifold learning

Unsupervised Feature Selection: Motivation

Goal (informal)

Find a **subset** of *informative* variables $\mathcal{S} \subset \{1, ..., D\}$ to improve clustering or manifold learning

When should we use feature selection?

Unsupervised Feature Selection: Motivation

Goal (informal)

Find a **subset** of *informative* variables $\mathcal{S} \subset \{1, ..., D\}$ to improve clustering or manifold learning

When should we use feature selection?

- # of variables exceeds the # of measurements ($D > N$)

Unsupervised Feature Selection: Motivation

Goal (informal)

Find a **subset** of *informative* variables $\mathcal{S} \subset \{1, ..., D\}$ to improve clustering or manifold learning

When should we use feature selection?

- # of variables exceeds the # of measurements ($D > N$)
- Some of the variables are nuisance (i.e. noisy and information-poor)

Unsupervised Feature Selection: Motivation

Goal (informal)

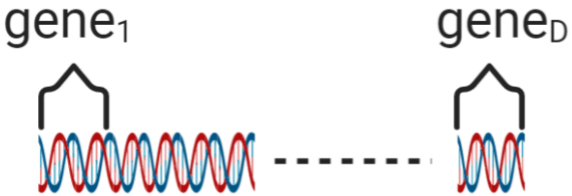
Find a **subset** of *informative* variables $\mathcal{S} \subset \{1, ..., D\}$ to improve clustering or manifold learning

When should we use feature selection?

- # of variables exceeds the # of measurements ($D > N$)
- Some of the variables are nuisance (i.e. noisy and information-poor)

Bio-informatics:

N- individuals, D- genes
Cluster traits/conditions



Unsupervised Feature Selection: Motivation

Goal (informal)

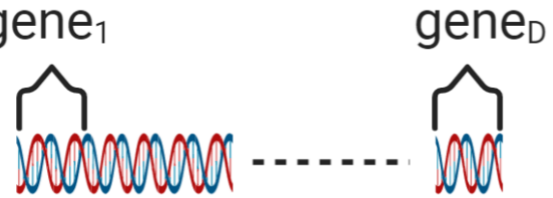
Find a **subset** of *informative* variables $\mathcal{S} \subset \{1, ..., D\}$ to improve clustering or manifold learning

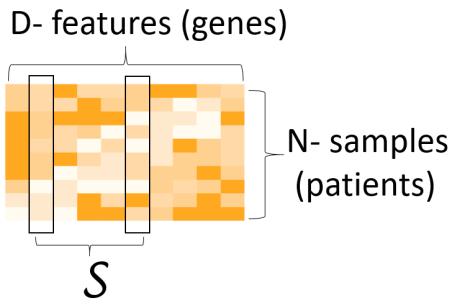
When should we use feature selection?

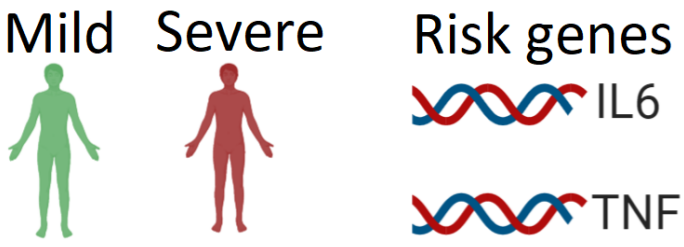
- # of variables exceeds the # of measurements ($D > N$)
- Some of the variables are nuisance (i.e. noisy and information-poor)

Bio-informatics:

N- individuals, D- genes
Cluster traits/conditions







Unsupervised Feature Selection: Motivation

Goal (informal)

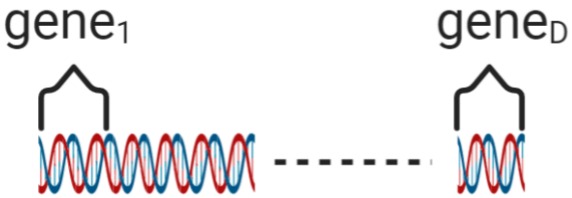
Find a **subset** of *informative* variables $\mathcal{S} \subset \{1, ..., D\}$ to improve clustering or manifold learning

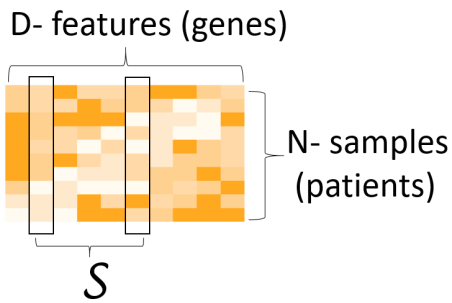
When should we use feature selection?

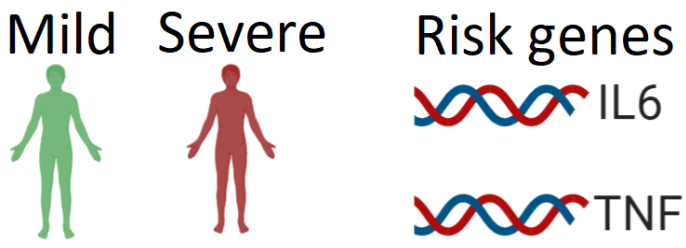
- # of variables exceeds the # of measurements ($D > N$)
- Some of the variables are nuisance (i.e. noisy and information-poor)

Bio-informatics:

N- individuals, D- genes
Cluster traits/conditions







The idea: use "smoothness" to identify *informative* variables

Unsupervised Learning: Kernel Methods

The graph Laplacian[1] is a useful tool for unsupervised learning

[1] Ng et al. (2001), Belkin et al. (2003)

Unsupervised Learning: Kernel Methods

The graph Laplacian[1] is a useful tool for unsupervised learning
Given measurements $\{\boldsymbol{x}_n\}_{n=1}^N$

[1] Ng et al. (2001), Belkin et al. (2003)

Unsupervised Learning: Kernel Methods

The graph Laplacian[1] is a useful tool for unsupervised learning
Given measurements $\{\boldsymbol{x}_n\}_{n=1}^N$

- Compute Gaussian kernel $K_{i,j} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$

[1] Ng et al. (2001), Belkin et al. (2003)

Unsupervised Learning: Kernel Methods

The graph Laplacian[1] is a useful tool for unsupervised learning
Given measurements $\{\boldsymbol{x}_n\}_{n=1}^N$

- Compute Gaussian kernel $K_{i,j} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$
- Compute $\boldsymbol{L} = \boldsymbol{S} - \boldsymbol{K}$, where $S_{i,i} = \sum_j K_{i,j}$

[1] Ng et al. (2001), Belkin et al. (2003)

Unsupervised Learning: Kernel Methods

The graph Laplacian[1] is a useful tool for unsupervised learning
Given measurements $\{\boldsymbol{x}_n\}_{n=1}^N$

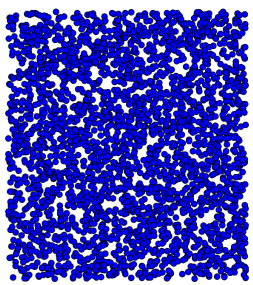
- Compute Gaussian kernel $K_{i,j} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$
- Compute $\boldsymbol{L} = \boldsymbol{S} - \boldsymbol{K}$, where $S_{i,i} = \sum_j K_{i,j}$
- Compute eigen-pairs $0 = \lambda_0 \leq \lambda_1, ..., \lambda_{N-1}$ and $\{\boldsymbol{\psi}_n\}_{n=0}^{N-1}$

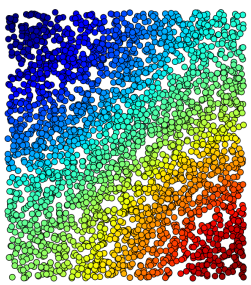
[1] Ng et al. (2001), Belkin et al. (2003)

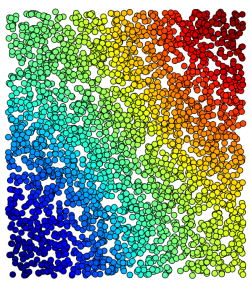
Unsupervised Learning: Kernel Methods

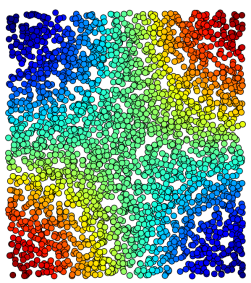
The graph Laplacian[1] is a useful tool for unsupervised learning
Given measurements $\{\boldsymbol{x}_n\}_{n=1}^N$

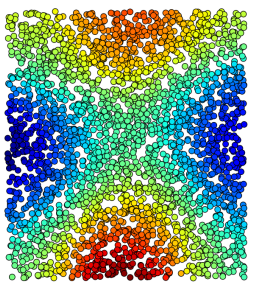
- Compute Gaussian kernel $K_{i,j} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$
- Compute $\boldsymbol{L} = \boldsymbol{S} - \boldsymbol{K}$, where $S_{i,i} = \sum_j K_{i,j}$
- Compute eigen-pairs $0 = \lambda_0 \leq \lambda_1, ..., \lambda_{N-1}$ and $\{\boldsymbol{\psi}_n\}_{n=0}^{N-1}$


$\boldsymbol{\psi}_0$


$\boldsymbol{\psi}_1$


$\boldsymbol{\psi}_2$


$\boldsymbol{\psi}_3$


$\boldsymbol{\psi}_4$

[1] Ng et al. (2001), Belkin et al. (2003)

Unsupervised Feature Selection

"Low-frequency" features could be identified using the **Laplacian-Score**[2]

[2] Niyogi et al. (2006)

Unsupervised Feature Selection

"Low-frequency" features could be identified using the **Laplacian-Score**[2]

- Normalize each feature $\bar{\boldsymbol{f}}_d = \frac{\boldsymbol{f}_d}{\|\boldsymbol{f}_d\|_2}$, where $\boldsymbol{f}_d = (x_1^{(d)}, ..., x_N^{(d)})^\top$

[2] Niyogi et al. (2006)

Unsupervised Feature Selection

"Low-frequency" features could be identified using the **Laplacian-Score**[2]

- Normalize each feature $\bar{\boldsymbol{f}}_d = \frac{\boldsymbol{f}_d}{\|\boldsymbol{f}_d\|_2}$, where $\boldsymbol{f}_d = (x_1^{(d)}, ..., x_N^{(d)})^\top$
- Compute the Rayleigh quotient

$$\text{Laplacian-Score}(d) \stackrel{\triangle}{=} \bar{\boldsymbol{f}}_d^\top \boldsymbol{L} \bar{\boldsymbol{f}}_d$$

[2] Niyogi et al. (2006)

Unsupervised Feature Selection

"Low-frequency" features could be identified using the **Laplacian-Score**[2]

- Normalize each feature $\bar{\boldsymbol{f}}_d = \frac{\boldsymbol{f}_d}{\|\boldsymbol{f}_d\|_2}$, where $\boldsymbol{f}_d = (x_1^{(d)}, ..., x_N^{(d)})^\top$
- Compute the Rayleigh quotient

$$\text{Laplacian-Score}(d) \stackrel{\triangle}{=} \bar{\boldsymbol{f}}_d^\top \boldsymbol{L} \bar{\boldsymbol{f}}_d = \sum_{n=0}^{N-1} \lambda_n \langle \boldsymbol{\psi}_n, \bar{\boldsymbol{f}}_d \rangle^2,$$

where $\boldsymbol{L} = \sum_{n=0}^{N-1} \lambda_n \boldsymbol{\psi}_n \boldsymbol{\psi}_n^\top$ is the eigen-decomposition of $\boldsymbol{L}$

[2] Niyogi et al. (2006)

Unsupervised Feature Selection

"Low-frequency" features could be identified using the **Laplacian-Score**[2]

- Normalize each feature $\bar{\boldsymbol{f}}_d = \frac{\boldsymbol{f}_d}{\|\boldsymbol{f}_d\|_2}$, where $\boldsymbol{f}_d = (x_1^{(d)}, ..., x_N^{(d)})^\top$
- Compute the Rayleigh quotient

$$\text{Laplacian-Score}(d) \stackrel{\triangle}{=} \bar{\boldsymbol{f}}_d^\top \boldsymbol{L} \bar{\boldsymbol{f}}_d = \sum_{n=0}^{N-1} \lambda_n \langle \boldsymbol{\psi}_n, \bar{\boldsymbol{f}}_d \rangle^2,$$

where $\boldsymbol{L} = \sum_{n=0}^{N-1} \lambda_n \boldsymbol{\psi}_n \boldsymbol{\psi}_n^\top$ is the eigen-decomposition of $\boldsymbol{L}$

Feature is correlated with low frequency eigenvectors

[2] Niyogi et al. (2006)

Unsupervised Feature Selection

"Low-frequency" features could be identified using the **Laplacian-Score**[2]

- Normalize each feature $\bar{\boldsymbol{f}}_d = \frac{\boldsymbol{f}_d}{\|\boldsymbol{f}_d\|_2}$, where $\boldsymbol{f}_d = (x_1^{(d)}, ..., x_N^{(d)})^\top$
- Compute the Rayleigh quotient

$$\text{Laplacian-Score}(d) \stackrel{\triangle}{=} \bar{\boldsymbol{f}}_d^\top \boldsymbol{L} \bar{\boldsymbol{f}}_d = \sum_{n=0}^{N-1} \lambda_n \langle \boldsymbol{\psi}_n, \bar{\boldsymbol{f}}_d \rangle^2,$$

where $\boldsymbol{L} = \sum_{n=0}^{N-1} \lambda_n \boldsymbol{\psi}_n \boldsymbol{\psi}_n^\top$ is the eigen-decomposition of $\boldsymbol{L}$

Feature is correlated with low frequency eigenvectors $\Longrightarrow$ Weighted by small eigenvalues

[2] Niyogi et al. (2006)

Unsupervised Feature Selection

"Low-frequency" features could be identified using the **Laplacian-Score**[2]

- Normalize each feature $\bar{\boldsymbol{f}}_d = \frac{\boldsymbol{f}_d}{\|\boldsymbol{f}_d\|_2}$, where $\boldsymbol{f}_d = (x_1^{(d)}, ..., x_N^{(d)})^\top$
- Compute the Rayleigh quotient

$$\text{Laplacian-Score}(d) \stackrel{\triangle}{=} \bar{\boldsymbol{f}}_d^\top \boldsymbol{L} \bar{\boldsymbol{f}}_d = \sum_{n=0}^{N-1} \lambda_n \langle \boldsymbol{\psi}_n, \bar{\boldsymbol{f}}_d \rangle^2,$$

where $\boldsymbol{L} = \sum_{n=0}^{N-1} \lambda_n \boldsymbol{\psi}_n \boldsymbol{\psi}_n^\top$ is the eigen-decomposition of $\boldsymbol{L}$

Feature is correlated with low frequency eigenvectors $\Longrightarrow$ Weighted by small eigenvalues $\Longrightarrow$ Small Laplacian-Score

[2] Niyogi et al. (2006)

Unsupervised Feature Selection

"Low-frequency" features could be identified using the **Laplacian-Score**[2]

- Normalize each feature $\bar{\boldsymbol{f}}_d = \frac{\boldsymbol{f}_d}{\|\boldsymbol{f}_d\|_2}$, where $\boldsymbol{f}_d = (x_1^{(d)}, ..., x_N^{(d)})^\top$
- Compute the Rayleigh quotient

$$\text{Laplacian-Score}(d) \stackrel{\triangle}{=} \bar{\boldsymbol{f}}_d^\top \boldsymbol{L} \bar{\boldsymbol{f}}_d = \sum_{n=0}^{N-1} \lambda_n \langle \boldsymbol{\psi}_n, \bar{\boldsymbol{f}}_d \rangle^2,$$

where $\boldsymbol{L} = \sum_{n=0}^{N-1} \lambda_n \boldsymbol{\psi}_n \boldsymbol{\psi}_n^\top$ is the eigen-decomposition of $\boldsymbol{L}$

Feature is correlated with low frequency eigenvectors $\Longrightarrow$ Weighted by small eigenvalues $\Longrightarrow$ Small Laplacian-Score

Problem

When many nuisance variable are added they dominate the Laplacian

[2] Niyogi et al. (2006)

Unsupervised Feature Selection

"Low-frequency" features could be identified using the **Laplacian-Score**[2]

- Normalize each feature $\bar{\boldsymbol{f}}_d = \frac{\boldsymbol{f}_d}{\|\boldsymbol{f}_d\|_2}$, where $\boldsymbol{f}_d = (x_1^{(d)}, ..., x_N^{(d)})^\top$
- Compute the Rayleigh quotient

$$\text{Laplacian-Score}(d) \stackrel{\triangle}{=} \bar{\boldsymbol{f}}_d^\top \boldsymbol{L} \bar{\boldsymbol{f}}_d = \sum_{n=0}^{N-1} \lambda_n \langle \boldsymbol{\psi}_n, \bar{\boldsymbol{f}}_d \rangle^2,$$

where $\boldsymbol{L} = \sum_{n=0}^{N-1} \lambda_n \boldsymbol{\psi}_n \boldsymbol{\psi}_n^\top$ is the eigen-decomposition of $\boldsymbol{L}$

Feature is correlated with low frequency eigenvectors $\Longrightarrow$ Weighted by small eigenvalues $\Longrightarrow$ Small Laplacian-Score

Problem

When many nuisance variable are added they dominate the Laplacian

e.g. (# nuisance variables)$\sim$ (cluster separation)^4 $\rightarrow$ Laplacian "breaks"

[2] Niyogi et al. (2006)

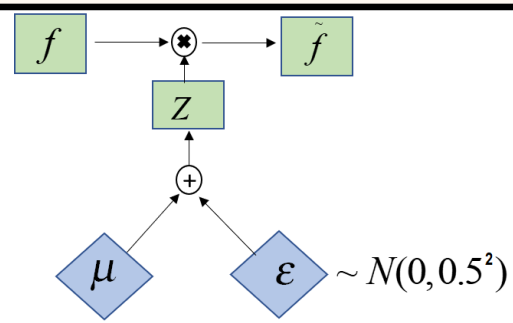
Differentiable Unsupervised Feature Selection (DUFS)

The idea: "clean" the Laplacian by gating nuisance features

Differentiable Unsupervised Feature Selection (DUFS)

The idea: "clean" the Laplacian by gating nuisance features

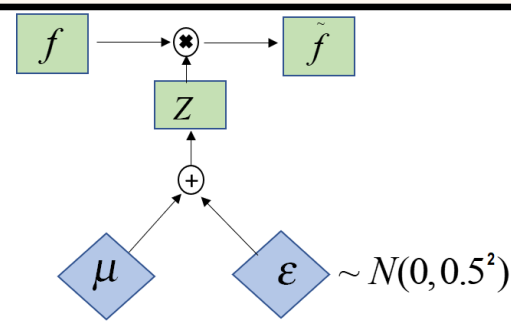
1. Filter features using stochastic gates



Differentiable Unsupervised Feature Selection (DUFS)

The idea: "clean" the Laplacian by gating nuisance features

1. Filter features using stochastic gates



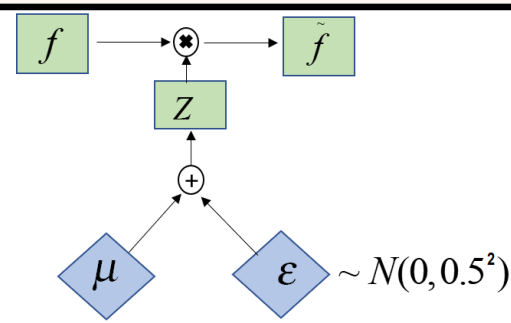
2. Define the **gated** measurement matrix

$$\bar{\boldsymbol{X}} = \left[\bar{\boldsymbol{f}}_1 \cdot \mathrm{z}_1 \middle| \quad \bar{\boldsymbol{f}}_2 \cdot \mathrm{z}_2 \middle| \quad \ldots \quad \middle| \bar{\boldsymbol{f}}_D \cdot \mathrm{z}_D\right]$$

Differentiable Unsupervised Feature Selection (DUFS)

The idea: "clean" the Laplacian by gating nuisance features

1. Filter features using stochastic gates

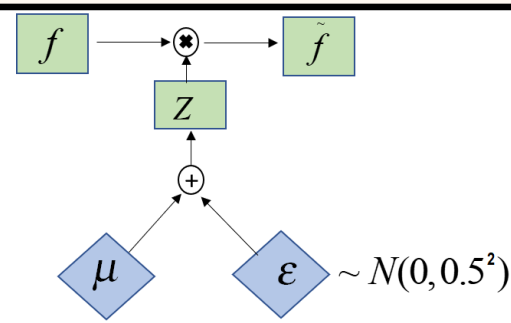


2. Define the **gated** measurement matrix
$\bar{\boldsymbol{X}} = \left[\bar{\boldsymbol{f}}_1 \cdot \mathrm{z}_1 \middle| \quad \bar{\boldsymbol{f}}_2 \cdot \mathrm{z}_2 \middle| \quad \ldots \quad \middle| \bar{\boldsymbol{f}}_D \cdot \mathrm{z}_D\right]$
3. Compute the **gated** diffusion operator $\boldsymbol{P}_{\bar{x}} = \boldsymbol{S}_{\bar{x}}^{-1} \boldsymbol{K}_{\bar{x}}$
$\boldsymbol{K}_{\bar{x}}$ and $\boldsymbol{S}_{\bar{x}}$ are the Kernel and degree matrices

Differentiable Unsupervised Feature Selection (DUFS)

The idea: "clean" the Laplacian by gating nuisance features

1. Filter features using stochastic gates



2. Define the **gated** measurement matrix
$\bar{\boldsymbol{X}} = \left[\bar{\boldsymbol{f}}_1 \cdot \mathsf{z}_1 \middle| \quad \bar{\boldsymbol{f}}_2 \cdot \mathsf{z}_2 \middle| \quad \ldots \quad \middle| \bar{\boldsymbol{f}}_D \cdot \mathsf{z}_D\right]$
3. Compute the **gated** diffusion operator $\boldsymbol{P}_{\bar{x}} = \boldsymbol{S}_{\bar{x}}^{-1}\boldsymbol{K}_{\bar{x}}$
$\boldsymbol{K}_{\bar{x}}$ and $\boldsymbol{S}_{\bar{x}}$ are the Kernel and degree matrices
4. Identify **smooth** features by minimizing

$$L(\boldsymbol{\mu}) := \underbrace{-\mathrm{Tr}\left[\bar{\boldsymbol{X}}^T \boldsymbol{P}_{\bar{x}} \bar{\boldsymbol{X}}\right]}_{\text{Smoothness}} + \lambda \underbrace{\mathbb{E}_{\mathsf{z}} \|\mathbf{z}\|_0}_{\text{Regularization}}$$

Gaussian **ST**ochastic **G**ates (**STG**)

The idea: use a truncated Gaussian to relax the Bernoulli distribution

Gaussian **ST**ochastic **G**ates (**STG**)

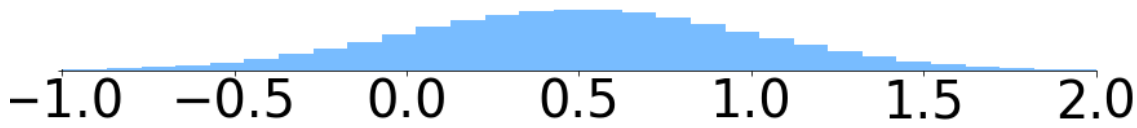
The idea: use a truncated Gaussian to relax the Bernoulli distribution

- Draw from a Gaussian $\epsilon \sim N(0, 0.5)$, shift by $\mu = 0.5$

Gaussian **ST**ochastic **G**ates (**STG**)

The idea: use a truncated Gaussian to relax the Bernoulli distribution

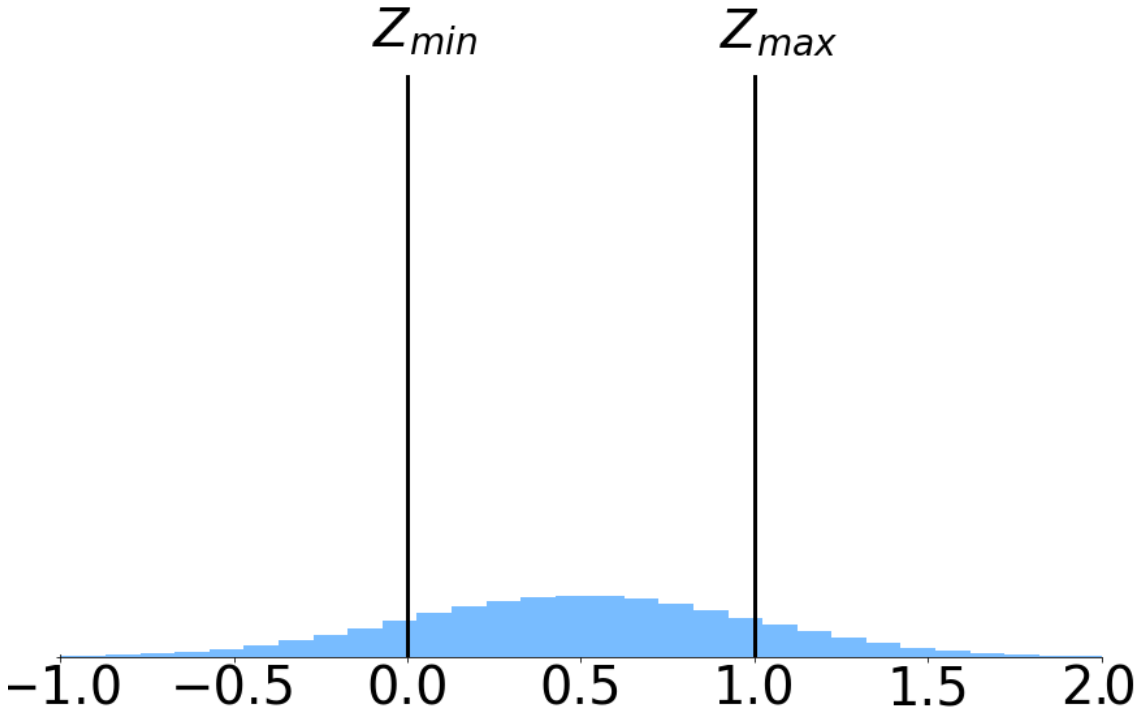
- Draw from a Gaussian $\epsilon \sim N(0, 0.5)$, shift by $\mu = 0.5$



Gaussian **ST**ochastic **G**ates (**STG**)

The idea: use a truncated Gaussian to relax the Bernoulli distribution

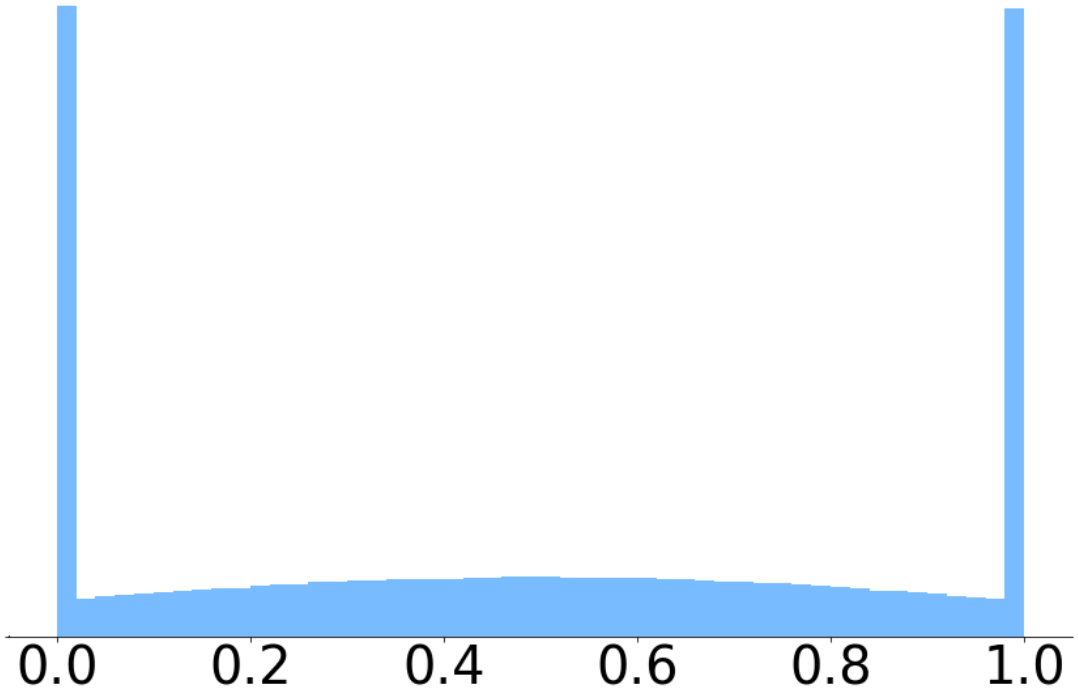
- Truncate into $[0, 1]$



Gaussian **ST**ochastic **G**ates (**STG**)

The idea: use a truncated Gaussian to relax the Bernoulli distribution

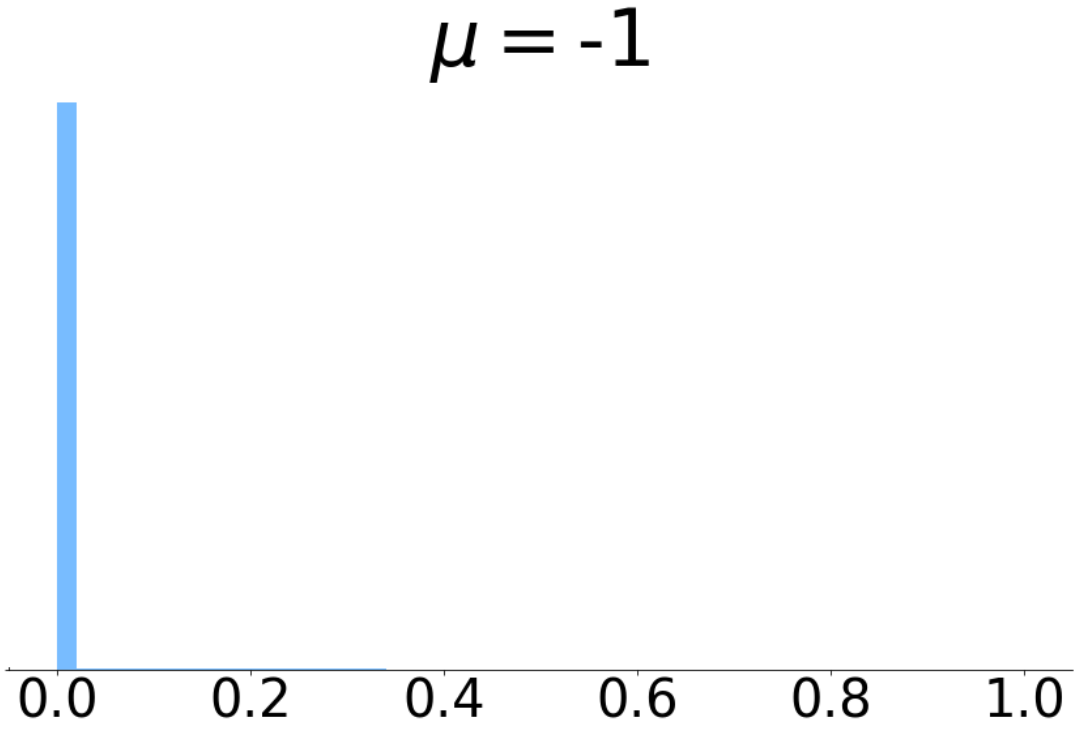
- Define the STochastic Gate (STG), denoted by z



Gaussian **ST**ochastic **G**ates (**STG**)

The idea: use a truncated Gaussian to relax the Bernoulli distribution

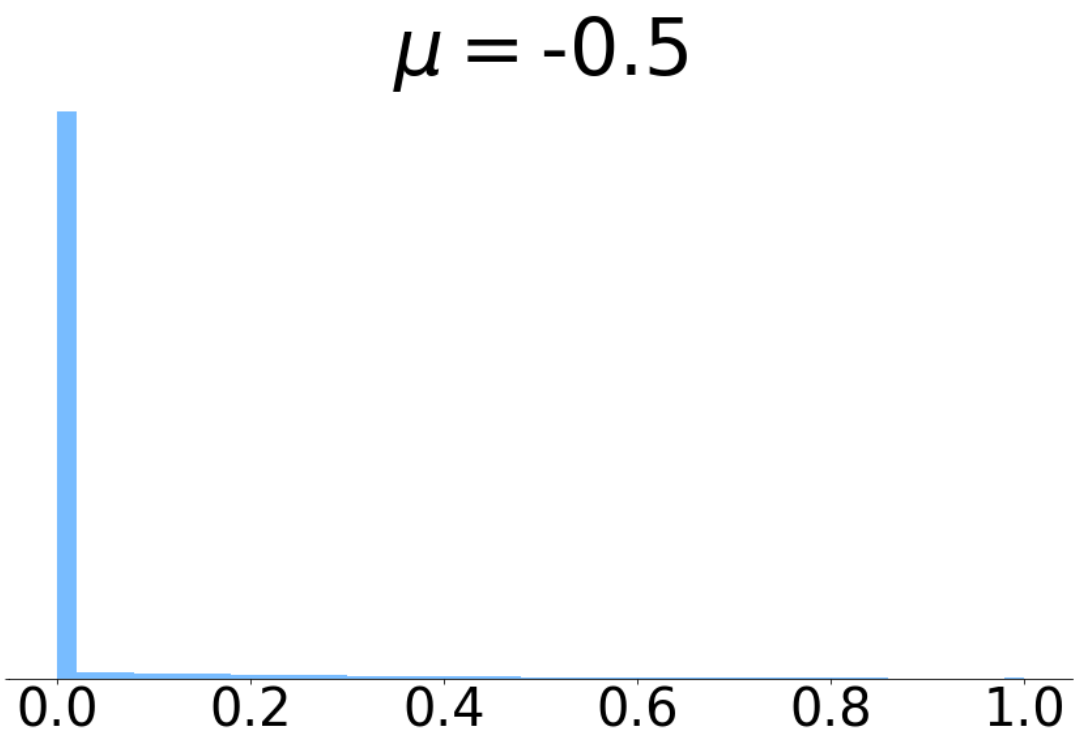
- Each gate is controlled by a trainable parameter μ



Gaussian **ST**ochastic **G**ates (**STG**)

The idea: use a truncated Gaussian to relax the Bernoulli distribution

- Each gate is controlled by a trainable parameter μ



- Learn model and gate parameters by minimizing

$$\widehat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}\in\mathbb{R}^D}{\arg\min} -\mathrm{Tr}\left[\bar{\boldsymbol{X}}^T \boldsymbol{P}_{\bar{x}} \bar{\boldsymbol{X}}\right] + \lambda \mathbb{E}_{\mathbf{z}} \|\mathbf{z}\|_0$$

$$\Downarrow$$

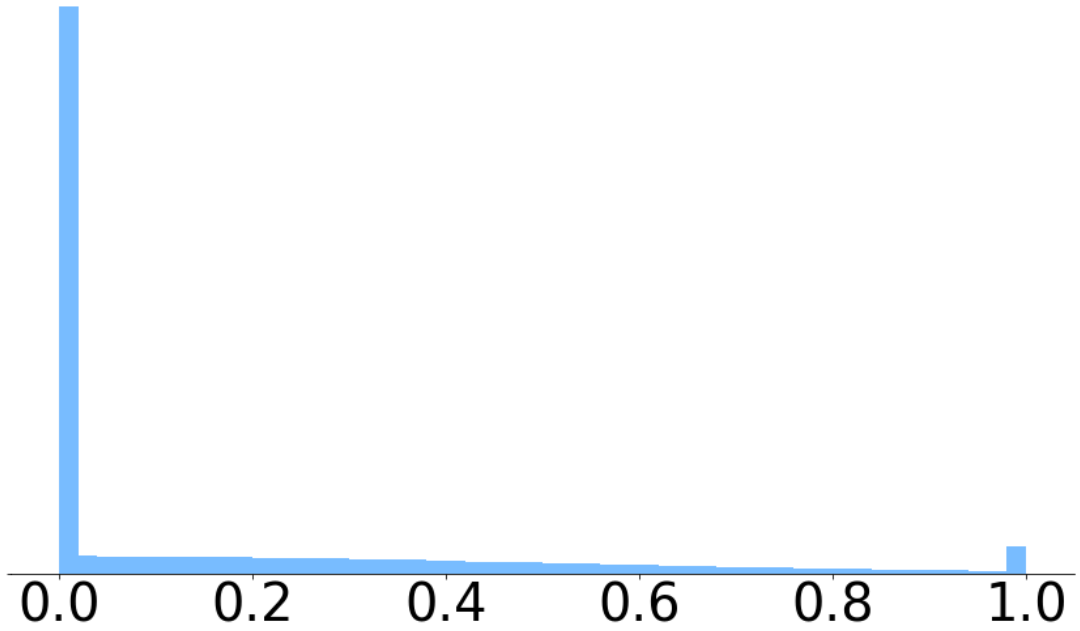
$$\sum_{d=1}^{D} \Phi\left(\frac{\mu_d}{0.5}\right)$$

Gaussian **ST**ochastic **G**ates (**STG**)

The idea: use a truncated Gaussian to relax the Bernoulli distribution

- Each gate is controlled by a trainable parameter μ

$$\mu = 0$$



- Learn model and gate parameters by minimizing

$$\widehat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}\in\mathbb{R}^D}{\arg\min} -\mathrm{Tr}\left[\bar{\boldsymbol{X}}^T \boldsymbol{P}_{\bar{x}} \bar{\boldsymbol{X}}\right] + \lambda \mathbb{E}_{\mathbf{z}} \|\mathbf{z}\|_0$$

$$\Downarrow$$

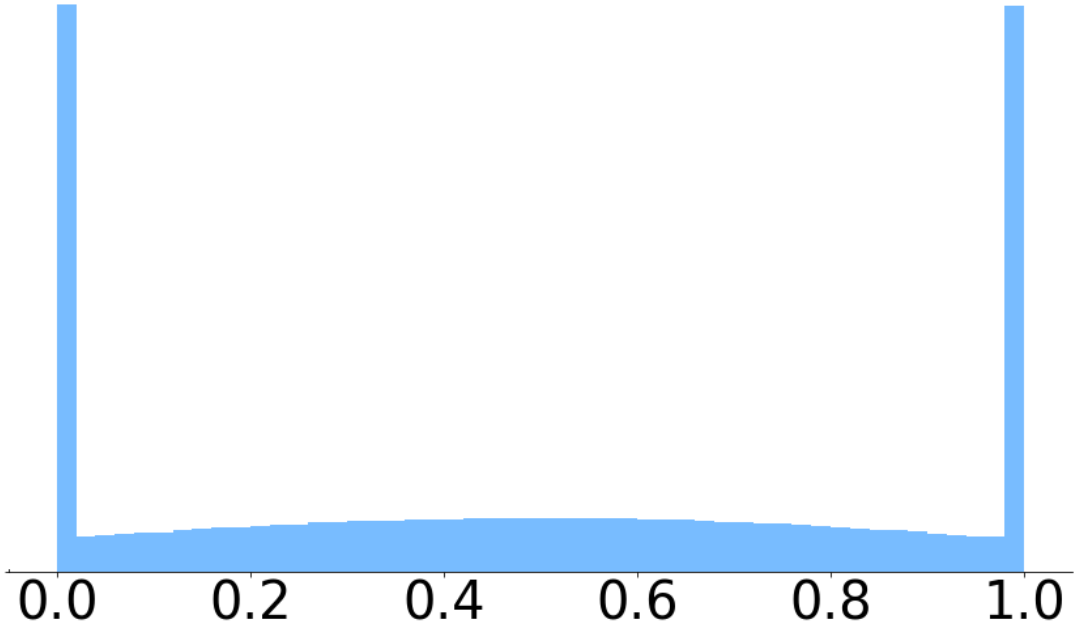
$$\sum_{d=1}^{D} \Phi\left(\frac{\mu_d}{0.5}\right)$$

Gaussian **ST**ochastic **G**ates (**STG**)

The idea: use a truncated Gaussian to relax the Bernoulli distribution

- Each gate is controlled by a trainable parameter μ

$$\mu = 0.5$$



- Learn model and gate parameters by minimizing

$$\widehat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}\in\mathbb{R}^D}{\arg\min} -\mathrm{Tr}\left[\bar{\boldsymbol{X}}^T \boldsymbol{P}_{\bar{x}} \bar{\boldsymbol{X}}\right] + \lambda \mathbb{E}_{\mathbf{z}} \|\mathbf{z}\|_0$$

$$\Downarrow$$

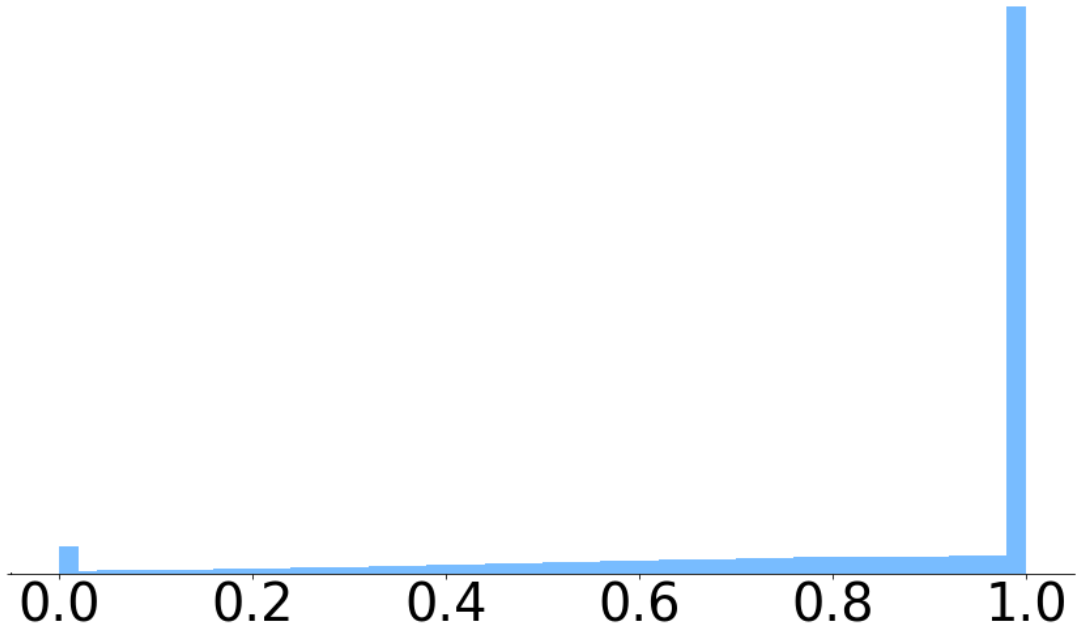
$$\sum_{d=1}^{D} \Phi\left(\frac{\mu_d}{0.5}\right)$$

Gaussian **ST**ochastic **G**ates (**STG**)

The idea: use a truncated Gaussian to relax the Bernoulli distribution

- Each gate is controlled by a trainable parameter μ

$$\mu = 1$$



- Learn model and gate parameters by minimizing

$$\widehat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}\in\mathbb{R}^D}{\arg\min} -\mathsf{Tr}\left[\bar{\boldsymbol{X}}^T \boldsymbol{P}_{\bar{x}} \bar{\boldsymbol{X}}\right] + \lambda \mathbb{E}_{\mathbf{z}} \|\mathbf{z}\|_0$$

$$\Downarrow$$

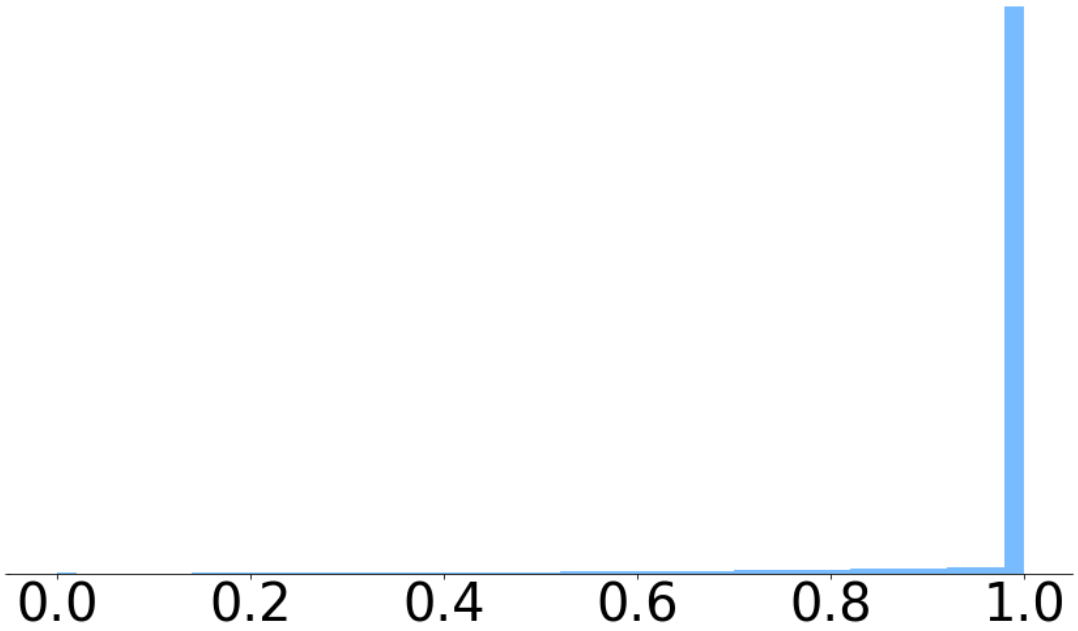
$$\sum_{d=1}^{D} \Phi\left(\frac{\mu_d}{0.5}\right)$$

Gaussian **ST**ochastic **G**ates (**STG**)

The idea: use a truncated Gaussian to relax the Bernoulli distribution

- Each gate is controlled by a trainable parameter μ

$$\mu = 1.5$$



- Learn model and gate parameters by minimizing

$$\widehat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}\in\mathbb{R}^D}{\arg\min} -\mathsf{Tr}\left[\bar{\boldsymbol{X}}^T \boldsymbol{P}_{\bar{x}} \bar{\boldsymbol{X}}\right] + \lambda \mathbb{E}_{\mathbf{z}} \|\mathbf{z}\|_0$$

$$\Downarrow$$

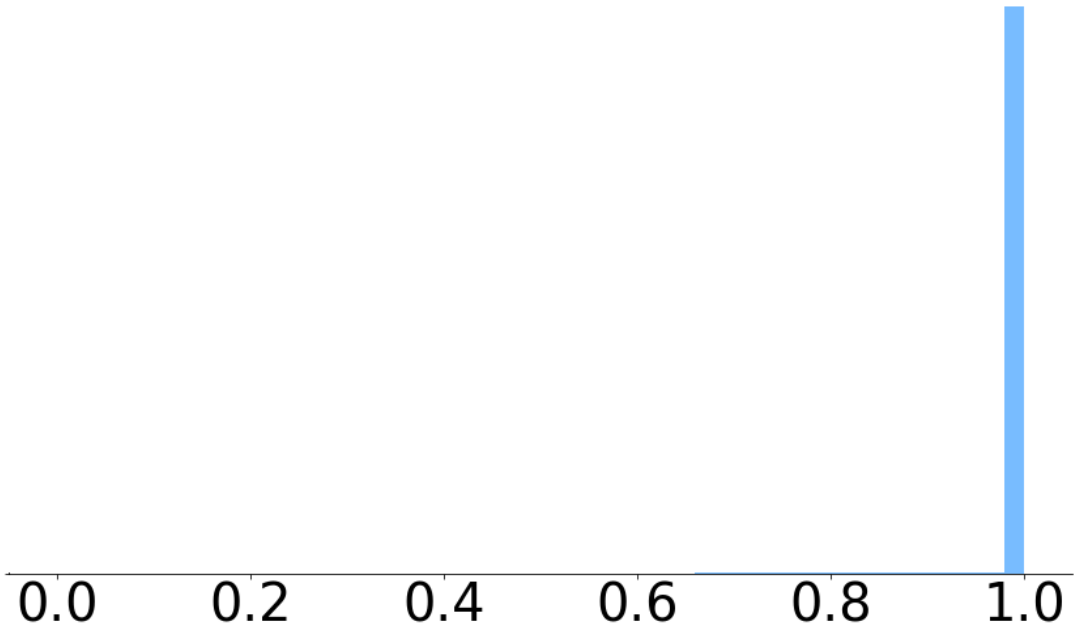
$$\sum_{d=1}^{D} \Phi\left(\frac{\mu_d}{0.5}\right)$$

Gaussian **ST**ochastic **G**ates (**STG**)

The idea: use a truncated Gaussian to relax the Bernoulli distribution

- Each gate is controlled by a trainable parameter μ

$$\mu = 2$$



- Learn model and gate parameters by minimizing

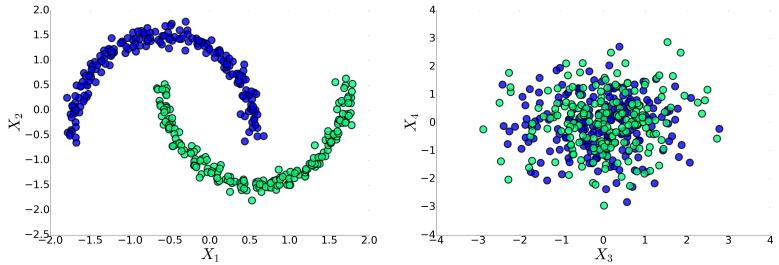
$$\widehat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}\in\mathbb{R}^D}{\arg\min} -\mathrm{Tr}\left[\bar{\boldsymbol{X}}^T \boldsymbol{P}_{\bar{x}} \bar{\boldsymbol{X}}\right] + \lambda \mathbb{E}_{\mathbf{z}} \|\mathbf{z}\|_0$$

$$\Downarrow$$

$$\sum_{d=1}^{D} \Phi\left(\frac{\mu_d}{0.5}\right)$$

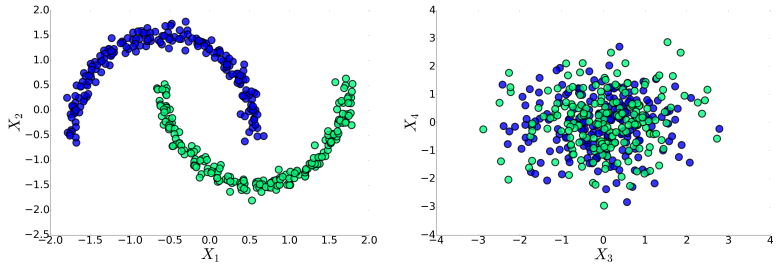
Results: Noisy Two-moons

Consider an **informative** data $\tilde{\boldsymbol{X}}$ concatenated with nuisance variables $\boldsymbol{X} = \left[\tilde{\boldsymbol{X}}|\boldsymbol{\xi}_1|...|\boldsymbol{\xi}_{D_n}\right]_{N\times(D)}$, where $D = D_c + D_n$, and $\xi_{i,j} \sim N(0,1)$

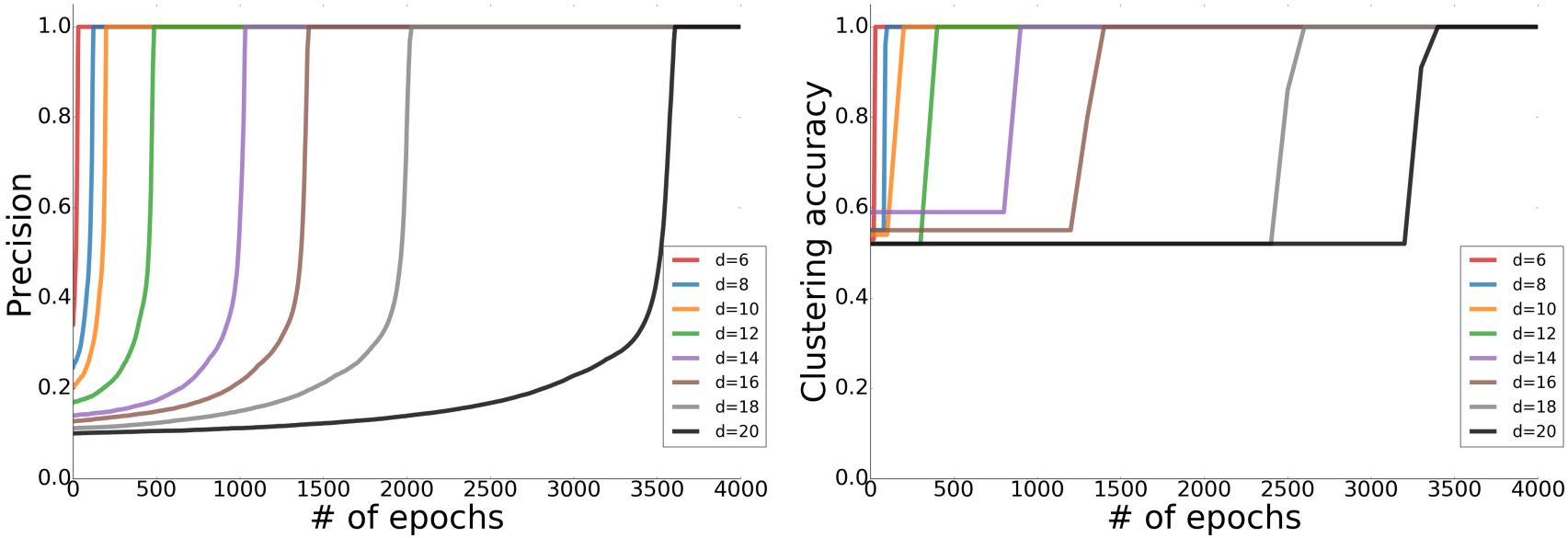


Results: Noisy Two-moons

Consider an **informative** data $\tilde{\boldsymbol{X}}$ concatenated with nuisance variables $\boldsymbol{X} = \left[\tilde{\boldsymbol{X}}|\boldsymbol{\xi}_1|...|\boldsymbol{\xi}_{D_n}\right]_{N\times(D)}$, where $D = D_c + D_n$, and $\xi_{i,j} \sim N(0,1)$



DUFS performance for different number of nuisance variables (d)



Results: MNIST

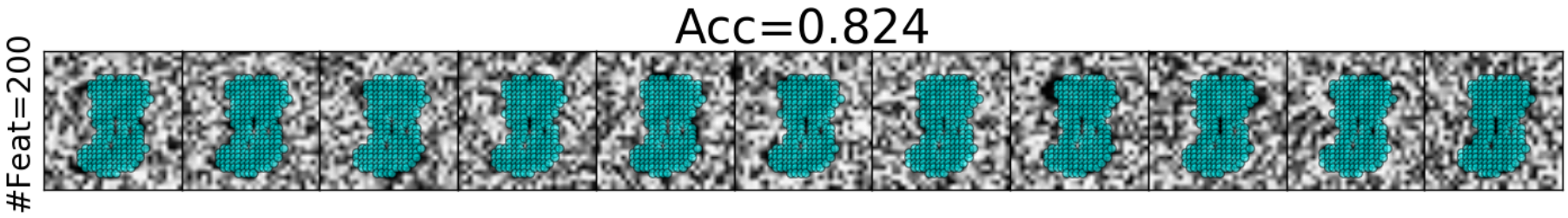
Use digits of 3s and 8s from noisy MNIST

Results: MNIST

Use digits of 3s and 8s from noisy MNIST

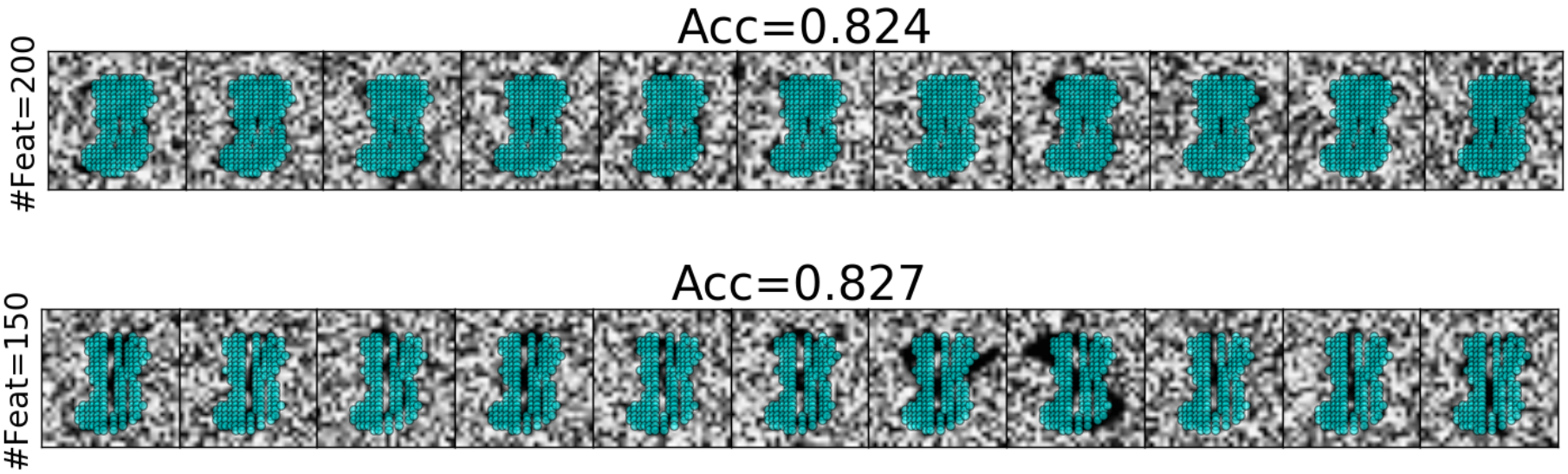
Results: MNIST

Selected features and clustering accuaracy



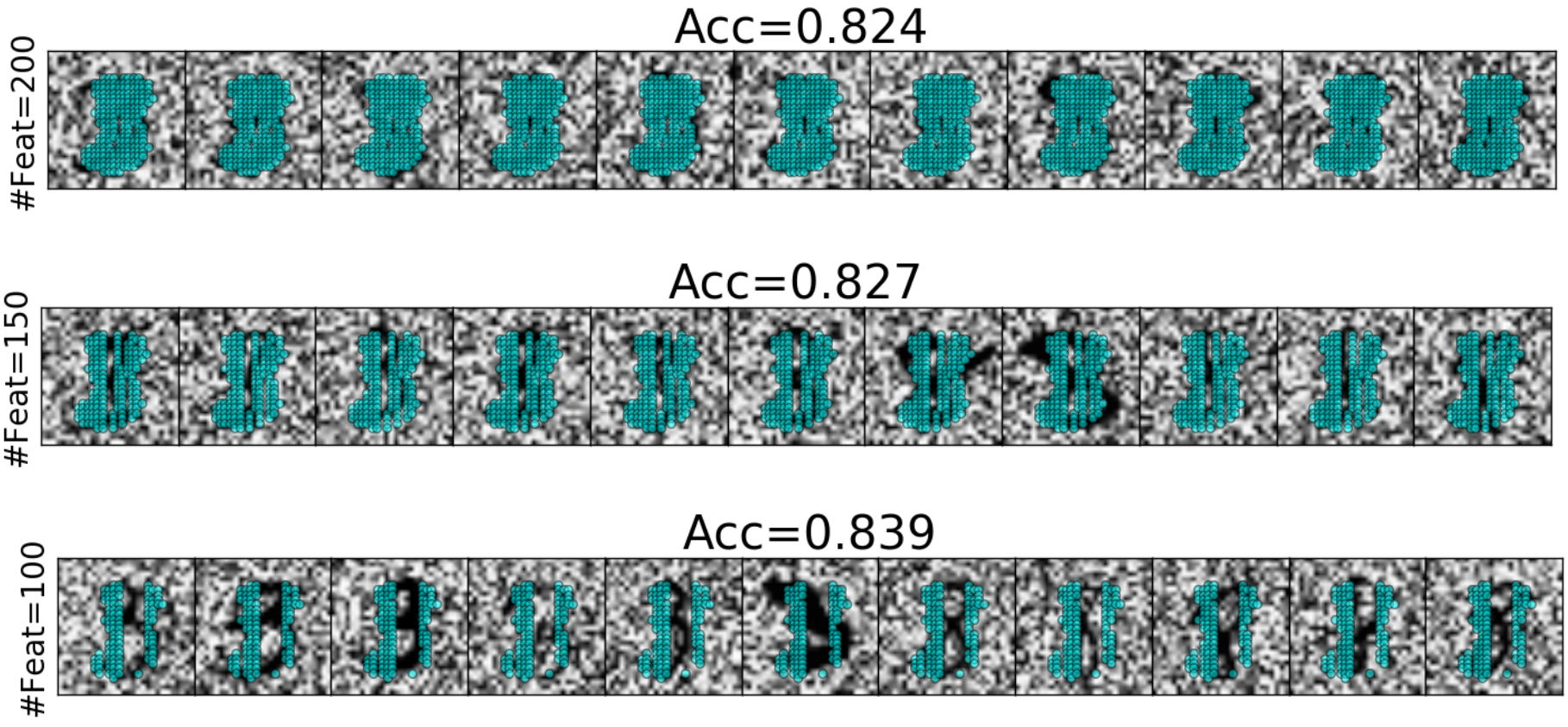
Results: MNIST

Selected features and clustering accuaracy



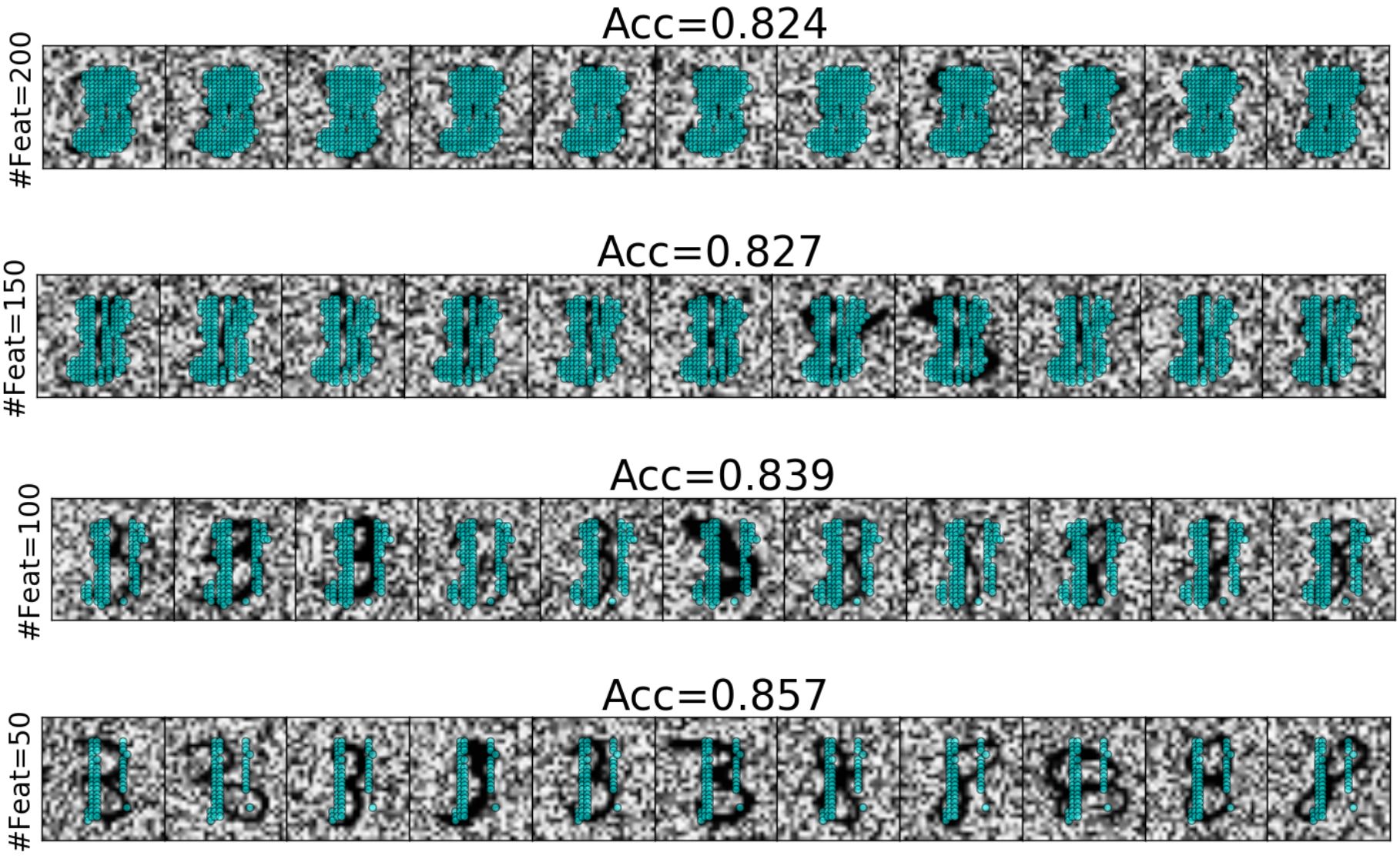
Results: MNIST

Selected features and clustering accuaracy



Results: MNIST

Selected features and clustering accuaracy



Results: Real Data

K-means accuracy on benchmark datasets[3] (# of selected features)

Datasets	LS	MCFS	NDFS	LLCFS	SRCFS	CAE	**DUFS**	All	Dim/Samples/Classes
GISETTE	75.8 (50)	56.5 (50)	69.3 (250)	72.5 (50)	68.5 (50)	77.3 (250)	**99.5 (50)**	74.4	4955 / 6000 / 2
PIX10	76.6 (150)	75.9 (200)	76.7 (200)	69.1 (300)	76.0 (300)	**94.1 (250)**	88.4 (50)	74.3	10000 / 100 / 10
COIL20	55.2 (250)	59.7 (250)	60.1 (300)	48.1 (300)	59.9 (300)	65.6 (200)	**65.8 (250)**	53.6	1024 / 1444 / 20
Yale	42.7 (300)	41.7 (300)	42.5 (300)	42.6 (300)	46.3 (250)	45.4 (250)	**47.9 (200)**	38.3	1024 / 165 / 15
TOX-171	47.5 (200)	42.5 (100)	46.1 (100)	46.7 (250)	45.8 (150)	44.4 (150)	**49.1 (50)**	41.5	2000 / 62 / 4
ALLAML	73.2 (150)	68.4 (100)	69.4 (100)	**77.8 (50)**	67.7 (250)	72.2 (200)	74.5 (100)	67.3	7192 / 72 / 2
PROSTATE	57.5 (300)	57.3 (300)	58.3 (100)	57.8 (50)	60.6 (50)	56.9 (250)	**64.7 (150)**	58.1	5966 / 102 / 2
RCV1	54.9 (300)	50.1 (150)	55.1 (150)	55.0 (300)	53.7 (300)	54.9 (300)	**60.2 (300)**	50.0	47236 / 21232 / 2
SRBCT	41.1(300)	43.7(250)	41.0(50)	34.6(150)	33.49(50)	**62.6 (200)**	51.7 (50)	39.6	2308 / 83 / 4
BIASE	83.8 (200)	95.5 (300)	100 (100)	52.2 (300)	50.8 (50)	85.1 (250)	**100 (50)**	41.8	25683 / 56 / 4
INTESTINE	43.2 (300)	48.2 (300)	42.3 (100)	63.3 (200)	58.1 (300)	51.9 (50)	**71.9 (250)**	54.8	3775 / 238 / 13
FAN	42.9 (150)	45.5 (150)	48.8 (100)	29.0 (50)	29.0 (100)	35.2 (300)	**49.0 (50)**	37.5	25683 / 56 / 8
POLLEN	46.9 (150)	**66.5 (300)**	48.9 (50)	35.0 (100)	34.9 (300)	58.0 (250)	60.2 (50)	54.9	21810 / 301 / 4
Median rank	4	6	4	4	5	3	**1**		
Mean rank	4.1	6	3.9	4.6	4.7	3.4	**1.3**		

[3] https://jundongl.github.io/scikit-feature/

Conclusion and Future Work

- "Cleaning" the Laplacian prior to calculation of the LS is cruical
- DUFS is also applicable for Manifold Learning
- Potential applications in computational biology, medicine
- Extending the method to handle correlated features
- Code available at https://github.com/Ofirlin/DUFS