Robust Covariance Neural Networks

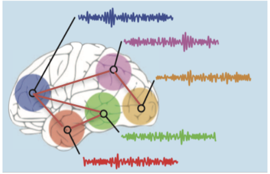
Andrea Cavallo*, Ayushman Raghuvanshi†, Sundeep Prabhakar Chepuri†, Elvin Isufi*

*Delft University of Technology, Delft, Netherlands
†Indian Institute of Science, Bangalore, India

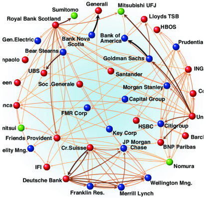
a.cavallo@tudelft.nl

Covariance relationships between data points

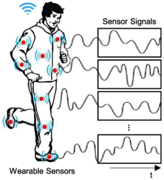
- Data usually contains latent interconnections



Brain



Financial



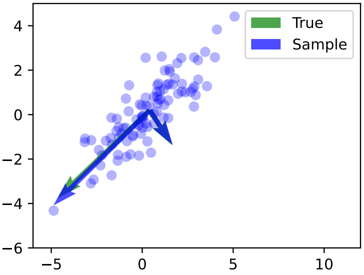
Body movement

- One way to capture these relations is through the covariance matrix

  $\Rightarrow \mathbf{C} = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^{\mathsf{T}}]$

Principal Component Analysis (PCA)

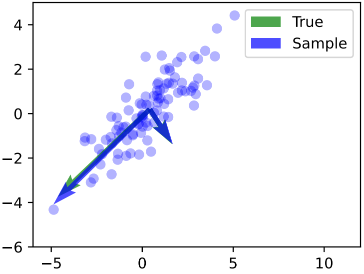
- Project the data onto the covariance eigenspace
  - ⇒ $\mathbf{C} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$
  - ⇒ $\tilde{\mathbf{x}} = \mathbf{V}^\top\mathbf{x}$
- Select (filter) the directions that maximize the variance of data points
  - ⇒ Used for dimensionality reduction by selecting only a few eigenvectors



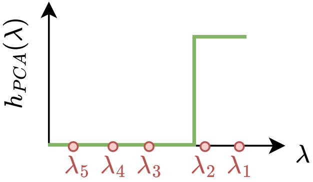
Principal directions

Principal Component Analysis (PCA)

- Project the data onto the covariance eigenspace
  - $\Rightarrow \mathbf{C} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$
  - $\Rightarrow \tilde{\mathbf{x}} = \mathbf{V}^\top \mathbf{x}$
- Select (filter) the directions that maximize the variance of data points
  - $\Rightarrow$ Used for dimensionality reduction by selecting only a few eigenvectors



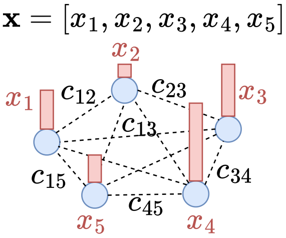
Principal directions



PCA filtering

Covariance Filters

- Data sample $\mathbf{x} = [x_1, \ldots, x_N]^\mathsf{T}$ with covariance $\mathbf{C}$
  - ⇒ Build a graph where:
  - ⇒ the features are the node signals x_i
  - ⇒ the edges are the covariance values c_{ij} → fully-connected graph



Covariance Filters

- Definition: Graph convolution covariance filters

$$\mathbf{z} = \mathbf{H}(\hat{\mathbf{C}})\mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k \mathbf{x}$$

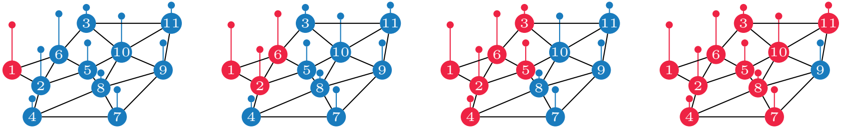
⇒ learnable parameters: h_k

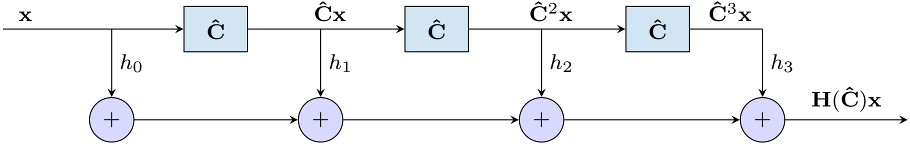
Covariance Filters

- Definition: Graph convolution covariance filters

$$\mathbf{z} = \mathbf{H}(\hat{\mathbf{C}})\mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k \mathbf{x}$$

$\Rightarrow$ learnable parameters: h_k



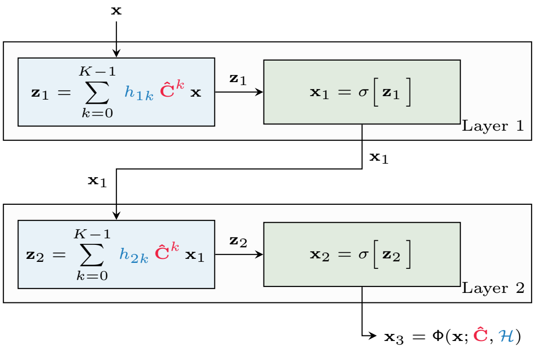


- $\hat{\mathbf{C}}^k\mathbf{x}$ shifts signal $\mathbf{x}$ k times over the covariance graph $\hat{\mathbf{C}}$

Covariance Neural Networks (VNNs)

- Definition: Covariance filters followed by pointwise nonlinearities σ

$$\mathbf{x}^l = \sigma\left(\mathbf{H}^l(\hat{\mathbf{C}})\mathbf{x}^{l-1}\right) \quad l = 1, \ldots, L.$$



Connections to PCA

- The covariance filter matrix has the form

$$\mathbf{H}(\hat{\mathbf{C}})\mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k \mathbf{x}$$

Connections to PCA

- The covariance filter matrix has the form

$$\mathbf{H}(\hat{\mathbf{C}})\mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k \mathbf{x}$$

- Taking the eigendecomposition $\hat{\mathbf{C}} = \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}^\top$ we get

$$\mathbf{H}(\hat{\mathbf{V}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}^\top)\mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}^k\hat{\mathbf{V}}^\top \mathbf{x}$$

Connections to PCA

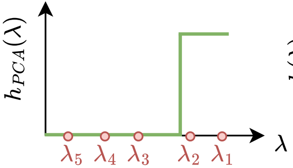
- The covariance filter matrix has the form

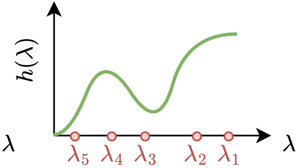
$$\mathbf{H}(\hat{\mathbf{C}})\mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k \mathbf{x}$$

- Taking the eigendecomposition $\hat{\mathbf{C}} = \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}^\top$ we get

$$\mathbf{H}(\hat{\mathbf{V}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}^\top)\mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}^k \hat{\mathbf{V}}^\top \mathbf{x}$$

$\Rightarrow$ The covariance filter processes the principal components $\hat{\mathbf{V}}^\top \mathbf{x}$!



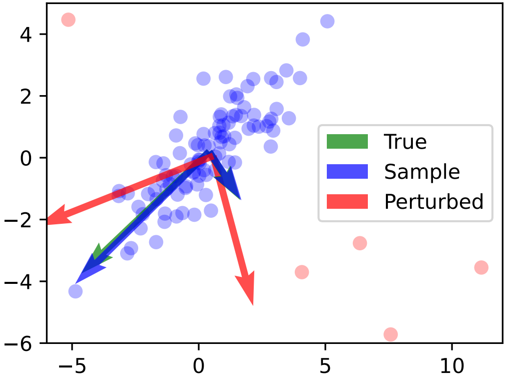


Outliers

- However, data might contain outliers or missing values

Outliers

- However, data might contain outliers or missing values
  - ⇒ PCA estimation is heavily affected
  - ⇒ VNNs do not work reliably



Robust Covariance Neural Networks

Desiderata:

- Use covariances for data processing
  - ⇒ with a learnable filter function
- Be robust to outliers and missing values
  - ⇒ via covariance correction terms learned end-to-end
- Be stable to finite-sample estimation errors

Outline

- Robust Covariance Neural Networks
- Theoretical results
- Experiments
- Conclusions

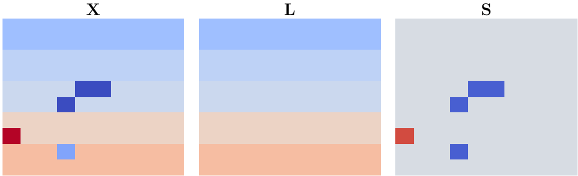
Outline

- **Robust Covariance Neural Networks**
- Theoretical results
- Experiments
- Conclusions

Data Perturbation Model

- Following robust PCA:
  - ⇒ Data $\mathbf{L}$ is low-rank
  - ⇒ Perturbations $\mathbf{S}$ are sparse

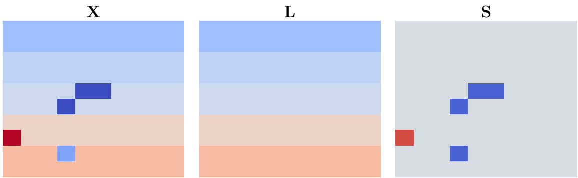
$$\mathbf{X} = \mathbf{L} + \mathbf{S}$$



Data Perturbation Model

- Following robust PCA:
  - ⇒ Data $\mathbf{L}$ is low-rank
  - ⇒ Perturbations $\mathbf{S}$ are sparse

$$\mathbf{X} = \mathbf{L} + \mathbf{S}$$



- The observed covariance $\hat{\mathbf{C}}$ is

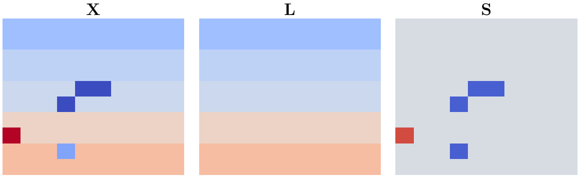
$$\hat{\mathbf{C}} = \mathbf{X}\mathbf{X}^\top / T = (\mathbf{L}+\mathbf{S})(\mathbf{L}+\mathbf{S})^\top / T = (\mathbf{L}\mathbf{L}^\top + \mathbf{S}\mathbf{L}^\top + \mathbf{L}\mathbf{S}^\top + \mathbf{S}\mathbf{S}^\top)/T$$

⇒ where $\mathbf{S}\mathbf{L}^\top + \mathbf{L}\mathbf{S}^\top$ is low-rank and $\mathbf{S}\mathbf{S}^\top$ is sparse

Data Perturbation Model

- Following robust PCA:
  - ⇒ Data $\mathbf{L}$ is low-rank
  - ⇒ Perturbations $\mathbf{S}$ are sparse

$$\mathbf{X} = \mathbf{L} + \mathbf{S}$$



- The observed covariance $\hat{\mathbf{C}}$ is

$$\hat{\mathbf{C}} = \mathbf{X}\mathbf{X}^\intercal/T = (\mathbf{L}+\mathbf{S})(\mathbf{L}+\mathbf{S})^\intercal/T = (\mathbf{L}\mathbf{L}^\intercal + \mathbf{S}\mathbf{L}^\intercal + \mathbf{L}\mathbf{S}^\intercal + \mathbf{S}\mathbf{S}^\intercal)/T$$

  - ⇒ where $\mathbf{S}\mathbf{L}^\intercal + \mathbf{L}\mathbf{S}^\intercal$ is low-rank and $\mathbf{S}\mathbf{S}^\intercal$ is sparse
- We reconstruct the clean covariance $\hat{\mathbf{C}} = \mathbf{L}\mathbf{L}^\intercal/T$ as

$$\hat{\mathbf{C}} = \hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l \quad (1)$$

  - ⇒ where $\mathbf{E}_s$ is sparse and $\mathbf{E}_l$ is low-rank.

Robust Covariance Neural Networks (RVNNs)

- RVNNs are VNNs Φ trained with the following objective

$$\min_{\mathcal{H},\mathbf{E}_s,\mathbf{E}_l} \mathcal{L}(\Phi(\mathcal{H}, \hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l), \mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}}) + \gamma_s \|\mathbf{E}_s\|_1 + \gamma_l \|\mathbf{E}_l\|_*$$

Robust Covariance Neural Networks (RVNNs)

- RVNNs are VNNs Φ trained with the following objective

$$\min_{\mathcal{H},\mathbf{E}_s,\mathbf{E}_l} \mathcal{L}(\Phi(\mathcal{H}, \hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l), \mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}}) + \gamma_s \|\mathbf{E}_s\|_1 + \gamma_l \|\mathbf{E}_l\|_*$$

$\Rightarrow$ prediction loss $\mathcal{L}$ on a training set $(\mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}})$

$\Rightarrow$ the VNN operates on the corrected covariance $\hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l$

Robust Covariance Neural Networks (RVNNs)

- RVNNs are VNNs Φ trained with the following objective

$$\min_{\mathcal{H}, \mathbf{E}_s, \mathbf{E}_l} \mathcal{L}(\Phi(\mathcal{H}, \hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l), \mathbf{X}_{\mathrm{tr}}, \mathbf{y}_{\mathrm{tr}}) + \gamma_s \|\mathbf{E}_s\|_1 + \gamma_l \|\mathbf{E}_l\|_*$$

$\Rightarrow$ prediction loss $\mathcal{L}$ on a training set $(\mathbf{X}_{\mathrm{tr}}, \mathbf{y}_{\mathrm{tr}})$

$\Rightarrow$ the VNN operates on the corrected covariance $\hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l$

$\Rightarrow$ promote sparsity for $\mathbf{E}_s$ via the 1-norm

Robust Covariance Neural Networks (RVNNs)

- RVNNs are VNNs Φ trained with the following objective

$$\min_{\mathcal{H},\mathbf{E}_s,\mathbf{E}_l} \mathcal{L}(\Phi(\mathcal{H}, \hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l), \mathbf{X}_{\mathrm{tr}}, \mathbf{y}_{\mathrm{tr}}) + \gamma_s \|\mathbf{E}_s\|_1 + \gamma_l \|\mathbf{E}_l\|_*$$

⇒ prediction loss $\mathcal{L}$ on a training set $(\mathbf{X}_{\mathrm{tr}}, \mathbf{y}_{\mathrm{tr}})$

⇒ the VNN operates on the corrected covariance $\hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l$

⇒ promote sparsity for $\mathbf{E}_s$ via the 1-norm

⇒ promote low-rank structure for $\mathbf{E}_l$ via the nuclear norm

Outline

- Robust Covariance Neural Networks
- **Theoretical results**
- Experiments
- Conclusions

Stability of RVNNs

- The observed covariance $\hat{\mathbf{C}}$ contains multiple perturbations w.r.t. the true covariance $\mathbf{C}$

  $\Rightarrow$ due to outliers

  $\Rightarrow$ due to finite-sample estimation

Stability of RVNNs

- The observed covariance $\hat{\mathbf{C}}$ contains multiple perturbations w.r.t. the true covariance $\mathbf{C}$
  - $\Rightarrow$ due to outliers
  - $\Rightarrow$ due to finite-sample estimation

- Stability of RVNN to covariance perturbations

$$\|\mathbf{H}(\hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l) - \mathbf{H}(\mathbf{C})\| \leq P\sqrt{N}(1 + \sqrt{N})(\mathcal{O}(T^{-1/2}) + \delta)$$

Stability of RVNNs

- The observed covariance $\hat{\mathbf{C}}$ contains multiple perturbations w.r.t. the true covariance $\mathbf{C}$
  - $\Rightarrow$ due to outliers
  - $\Rightarrow$ due to finite-sample estimation

- Stability of RVNN to covariance perturbations

$$\|\mathbf{H}(\hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l) - \mathbf{H}(\mathbf{C})\| \leq P\sqrt{N}(1 + \sqrt{N})(\mathcal{O}(T^{-1/2}) + \delta)$$

  - $\Rightarrow$ Stability improves with increasing number of samples T

Stability of RVNNs

- The observed covariance $\hat{\mathbf{C}}$ contains multiple perturbations w.r.t. the true covariance $\mathbf{C}$
  - ⇒ due to outliers
  - ⇒ due to finite-sample estimation

- Stability of RVNN to covariance perturbations

$$\|\mathbf{H}(\hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l) - \mathbf{H}(\mathbf{C})\| \leq P\sqrt{N}(1 + \sqrt{N})(\mathcal{O}(T^{-1/2}) + \delta)$$

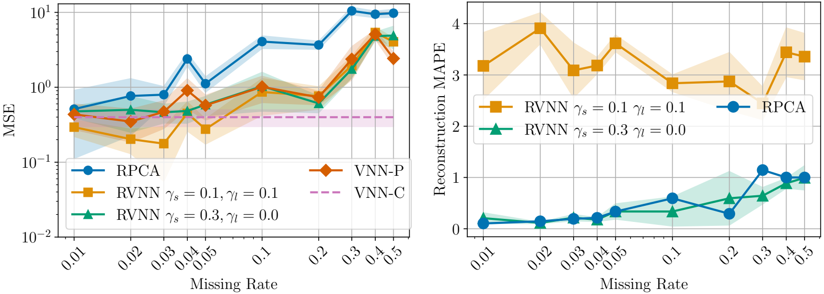
  - ⇒ Stability improves with increasing number of samples T
  - ⇒ $\delta = \|\tilde{\mathbf{C}} - \hat{\mathbf{C}} - \mathbf{E}_s - \mathbf{E}_l\|$ measures the quality of reconstruction of the clean covariance matrix

Outline

- Robust Covariance Neural Networks
- Theoretical results
- **Experiments**
- Conclusions

Synthetic Dataset

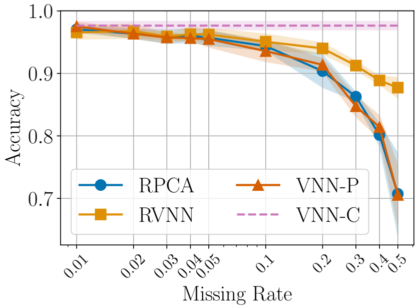
- Setup: Regression task, varying size of missing data
- Baselines: VNN on clean data (VNN-C), VNN on perturbed data (VNN-P), RPCA+VNN

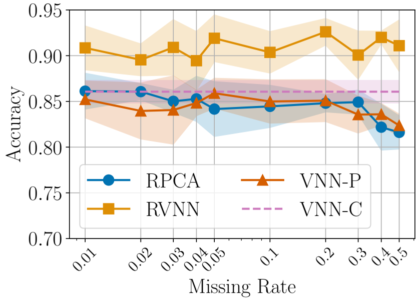


- Results:
  - ⇒ RVNN matches or improves performance of VNN-C for missing rate ≤ 0.05
  - ⇒ Based on γ_s, γ_l, covariance reconstruction is good or bad

Real Datasets

- **Datasets**:
  - ⇒ Brain recordings before and after epilepsy seizure – binary classification
  - ⇒ Motion sensor recordings – activity classification





- **Results**:
  - ⇒ RVNN less affected by missing values than other models

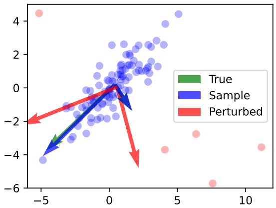
Outline

- Robust Covariance Neural Networks
- Theoretical results
- Experiments
- **Conclusions**

Conclusions

- Outliers and missing values make covariance estimation difficult

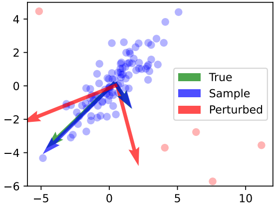
  $\Rightarrow$ PCA and VNN are unreliable



Conclusions

- Outliers and missing values make covariance estimation difficult

  $\Rightarrow$ PCA and VNN are unreliable



- Robust Covariance Neural Networks (RVNNs)

  $\Rightarrow$ Sparse and low-rank covariance correction learned end-to-end

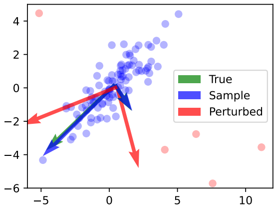
  $\Rightarrow$ Stable to finite-sample estimation errors

  $\Rightarrow$ Maintain downstream performance for varying missing rates

$$\|\mathbf{H}(\hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l) - \mathbf{H}(\mathbf{C})\| \leq P\sqrt{N}(1+\sqrt{N})(\mathcal{O}(T^{-1/2}) + \delta)$$

Conclusions

- Outliers and missing values make covariance estimation difficult
  - $\Rightarrow$ PCA and VNN are unreliable



- Robust Covariance Neural Networks (RVNNs)
  - $\Rightarrow$ Sparse and low-rank covariance correction learned end-to-end
  - $\Rightarrow$ Stable to finite-sample estimation errors
  - $\Rightarrow$ Maintain downstream performance for varying missing rates

$$\|\mathbf{H}(\hat{\mathbf{C}} + \mathbf{E}_s + \mathbf{E}_l) - \mathbf{H}(\mathbf{C})\| \leq P\sqrt{N}(1+\sqrt{N})(\mathcal{O}(T^{-1/2}) + \delta)$$

- Thank you for the attention!
  - $\Rightarrow$ a.cavallo@tudelft.nl