How Randomness Helps Us Do "Data Science"

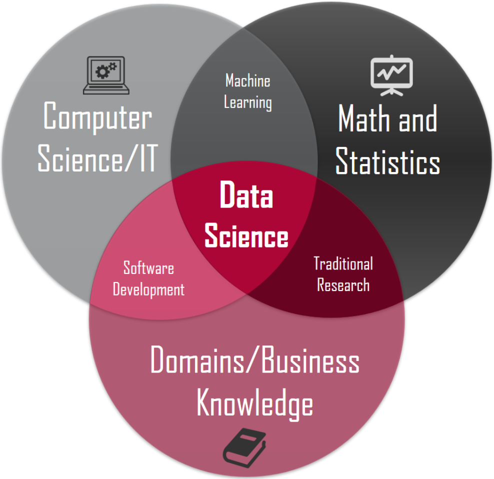
Examples in deep learning and graph analysis

Fred Roosta

School of Mathematics and Physics
University of Queensland

What Is Data Science?

What Is Data Science?



Source: https://towardsdatascience.com/introduction-to-statistics-e9d72d818745

Randomized Algorithms vs. Randomized Analysis

To analyze data, one often (implicitly) works with models....

- **Randomized Methods:**

  How to efficiently compute with models

- **Randomized Analysis:**

  Why models work or not

Randomized Analysis...

How randomized analysis helps answer “data sciency” questions?

Examples:

- Deep Learning
- Graph Analysis

Deep Learning...

Invariance of Weight Distributions in Rectified MLPs
(ICML, 2018)

Russell Tsuchida
(UQ)

Marcus Gallagher
(UQ)

Graph Analysis...

Out-of-sample extension of graph adjacency spectral embedding (ICML, 2018)

Keith Levin (Michigan)

Michael Mahoney (Berkeley)

Carey E. Priebe (Johns Hopkins)

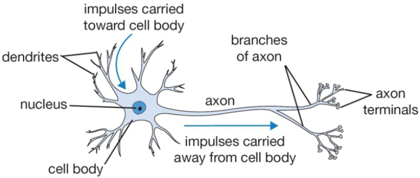
Randomized Analysis...

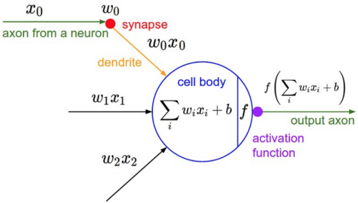
How randomized analysis helps answer “data sciency” questions?

Examples:

- Deep Learning
- Graph Analysis

Randomized Analysis: Deep Learning

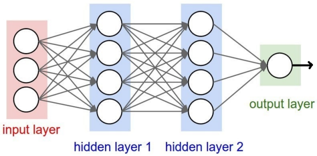




Source: https://isaacchanghau.github.io/post/activation_functions/

Neural Nets

Neural Nets: Composition of Nonlinear Functions

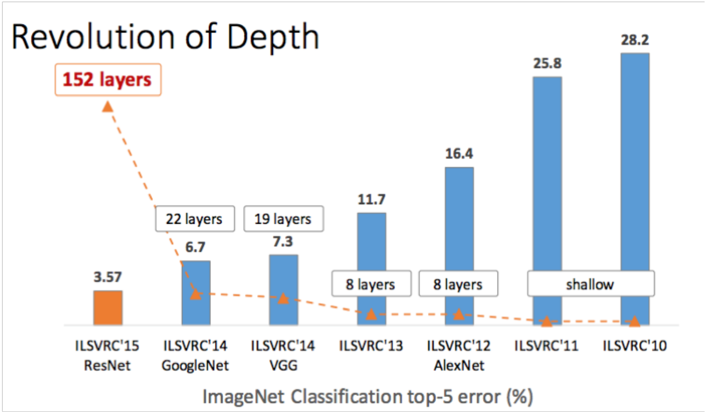


$$\implies \quad \hat{y} = \sigma\Big(\mathbf{W}_3\sigma\big(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x})\big)\Big)$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_1} \end{bmatrix}, \ \mathbf{W}^{(\ell)} = \begin{bmatrix} w_{1,1}^{(\ell)} & w_{1,2}^{(\ell)} & \cdots & w_{1,n_\ell}^{(\ell)} \\ w_{2,1}^{(\ell)} & w_{2,2}^{(\ell)} & \cdots & w_{2,n_\ell}^{(\ell)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_{\ell+1},1}^{(\ell)} & w_{n_{\ell+1},2}^{(\ell)} & \cdots & w_{n_{\ell+1},n_\ell}^{(\ell)} \end{bmatrix}, \ \ell = 1, 2, 3$$

$$n_1 = 3, \ n_2 = 4, \ n_3 = 4, \ n_4 = 1$$

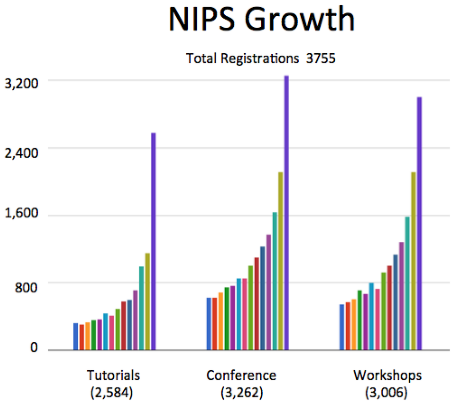
Neural Nets: Deep Learning Revolution



Source: https://medium.com/@Lidinwise/the-revolution-of-depth-facf174924f5


I WAS WINNING IMAGENET
UNTIL A DEEPER MODEL CAME ALONG
imgflip.com

Neural Nets: Deep Learning Revolution



Source: https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html

Neural Nets: Deep Learning Revolution

"*Neural networks are the second best way to do almost anything!* "

JS Denker

Deep Learning: Depth is good..but

Is it all rosy?

Deep Learning: Problems with Depth

Beyond many computational constraints, there are other inherent issues with increasing depth...

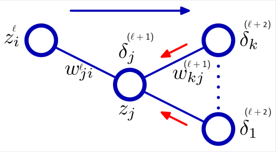
Deep Learning: Problems with Depth

Vanishing and Exploding Gradient Problem: Algebraic

$$\hat{y} = \sigma\left(\mathbf{W}_3\sigma\left(\mathbf{W}_2\sigma\left(\mathbf{W}_1\mathbf{x}\right)\right)\right), \; L(\mathbf{W}) = \frac{1}{2}(y - \hat{y})^2$$

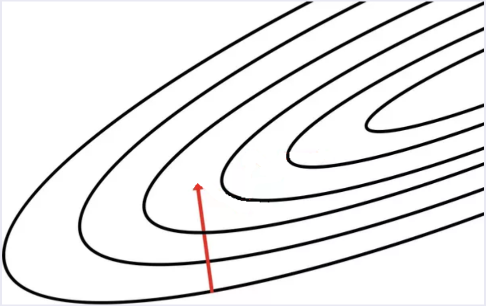
$$\frac{\partial L(\mathbf{W})}{\partial w_{ji}^{(\ell)}} = \underbrace{\delta_j^{(\ell+1)}}_{\text{errors at } \ell+1 \text{ layer}} \times \underbrace{z_i^{(\ell)}}_{\text{activation output of at } \ell \text{ layer}}$$

$$\delta_j^{(\ell+1)} = \sigma'(a_j^{(\ell+1)}) \sum_{k=1}^{n_{\ell+2}} w_{k,j}^{(\ell+1)} \delta_k^{(\ell+2)}$$



Deep Learning: Problems with Depth

Vanishing and Exploding Gradient Problem: Geometric



Deep Learning: Problems with Depth

Vanishing and Exploding Gradient Problem

The problem has largely been overcome via

- Rectified Linear Units (ReLU)
- Careful Initialization
- Small Learning Rates (step-size)
- Batch Normalization
- Skip Connections, e.g., ResNet, Highway Networks
- etc...

Deep Learning: Problems with Depth

Most of these aim at mitigating the issues with depth from an algebraic and/or geometric point of view.

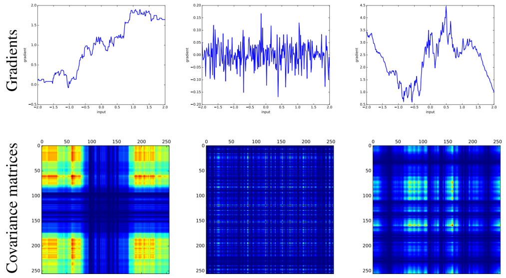
Are these all the view points that there is?

No: statistical/randomized point of view.

Deep Learning: Problems with Depth

Shattered Gradients Problem [Balduzzi et al., 2017]

Depth $\uparrow$ $\implies$ Gradients[a] $\approx$ White Noise



(a) 1-layer feedforward. (b) 24-layer feedforward. (c) 50-layer resnet.

[a] Gradients w.r.t the inputs...also, only at initialization

Deep Learning: Problems with Depth

Shattered Gradients Problem [Balduzzi et al., 2017]

Correlations between gradients decrease as

- Feedforward Rectifier Networks: $(1/2)^L$
- Resnet (No Batch Normalization): $(3/4)^L$
- Resnet (With Batch Normalization): $1/\sqrt{L}$

Deep Learning: Problems with Depth

- Algebraic/Geometric:
  - Exploding/Vanishing Gradient Problem
- Randomized/Statistical:
  - Shattered Gradient Problem
  - Kernelized Reducing Angle Problem (KRAP)

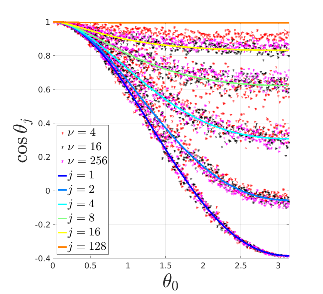
Deep Learning: Problems with Depth

Deep (rectified) feedforward nets are “KRAPY”!

Deep Learning: Problems with Depth

Kernelized Reducing Angle Problem (KRAP) [Tsuchida et al., 2018]

Depth ↑ $\implies$ Kernelized angle[a] between inputs ↓



[a] Only at initialization

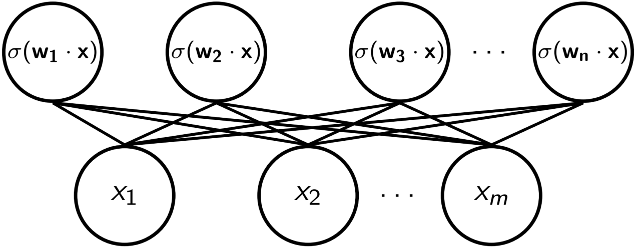
Deep Learning: Problems with Depth

Suppose $\sigma(t) = t$, i.e., linear activation function...

Recall: Power Iteration or Krylov Subspace Methods

$$\text{Start with any } \mathbf{x}_0, \mathbf{y}_0$$
$$\mathbf{x}_{k+1} \leftarrow \mathbf{A}\mathbf{x}_k, \ \mathbf{y}_{k+1} \leftarrow \mathbf{A}\mathbf{y}_k$$
$$\lim_{k \to \infty} \cos(\mathbf{x}_k, \mathbf{y}_k) \in \{\pm 1\}$$

Neural Nets: Universal Kernel



$$\boldsymbol{\sigma}_n(\mathbf{x}) \triangleq \begin{bmatrix} \sigma\left(\langle \mathbf{x}, \mathbf{w}_1 \rangle\right) \\ \sigma\left(\langle \mathbf{x}, \mathbf{w}_2 \rangle\right) \\ \vdots \\ \sigma\left(\langle \mathbf{x}, \mathbf{w}_n \rangle\right) \end{bmatrix} \implies \underbrace{\langle \boldsymbol{\sigma}_n(\mathbf{x}), \boldsymbol{\sigma}_n(\mathbf{y}) \rangle}_{\text{“angle” at output}} = \sum_{i=1}^{n} \sigma\left(\langle \mathbf{x}, \mathbf{w}_i \rangle\right) \sigma\left(\langle \mathbf{y}, \mathbf{w}_i \rangle\right)$$

Neural Nets: Universal Kernel

$$\lim_{n\to\infty} \frac{1}{n} \langle \boldsymbol{\sigma}_n(\mathbf{x}), \boldsymbol{\sigma}_n(\mathbf{y}) \rangle \overset{\text{LLN}}{=} \underbrace{\int_{\mathcal{W}\subseteq\mathcal{R}^m} \sigma(\langle \mathbf{x}, \mathbf{w}\rangle)\sigma(\langle \mathbf{y}, \mathbf{w}\rangle) f(\mathbf{w}) \mathrm{d}\mathbf{w}}_{\text{inner product in feature space}}$$

$$\triangleq \underbrace{\kappa(\mathbf{x}, \mathbf{y})}_{\text{the unique kernel of the unique RKHS}}$$

E.g.,

$$\phi(\mathbf{x}) \triangleq \sigma(\langle \mathbf{x}, .\rangle)\sqrt{f(.)} \in \mathcal{H}_\kappa = \{h : \mathcal{W} \to \mathcal{R}\}$$

$\phi(\mathbf{x})$: a mapping from the input space into a Hilbert Space, i.e., we can think of an MLP as a member of $\mathcal{H}_\kappa$

Neural Nets: Universal Kernel

Arc-Cosine Kernel: Gaussian, ReLU [Cho and Saul, 2009]

$$\kappa(\mathbf{x}, \mathbf{y}) = \frac{\sigma^2 \|\mathbf{x}\| \|\mathbf{y}\|}{2\pi} \left(\sin\theta_0 + (\pi - \theta_0)\cos\theta_0\right),$$

where

$$\sigma^2 = \mathbb{E}[W^2], \quad \theta_0 = \cos^{-1}\left(\frac{\langle \mathbf{x}, \mathbf{y}\rangle}{\|\mathbf{x}\| \|\mathbf{y}\|}\right)$$

Neural Nets: Universal Kernel

Arc-Cosine Kernel: Rotationally-Inv, ReLU [Tsuchida et al., 2018]

$$\kappa(\mathbf{x}, \mathbf{y}) = \frac{\sigma^2 \|\mathbf{x}\| \|\mathbf{y}\|}{2\pi} \left(\sin\theta_0 + (\pi - \theta_0)\cos\theta_0\right),$$

where

$$\sigma^2 = \mathbb{E}[W^2], \quad \theta_0 = \cos^{-1}\left(\frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|}\right)$$

- Examples of rotationally-invariant: Gaussian, multivariate t, symmetric multivariate Laplace, symmetric multivariate stable

Neural Nets: Universal Kernel

Arc-Cosine Kernel: Rotationally-Inv [Tsuchida et al., 2018]

Equivalent formulation for (L)ReLU:

$$\kappa(\mathbf{x}, \mathbf{y}) = \mathbb{E}\left(\sigma(Z_1)\sigma(Z_2)\right),$$

where

$$\mathbf{Z} = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \Sigma),$$

$$\Sigma = \mathbb{E}(W_i^2) \begin{bmatrix} \|\mathbf{x}\|^2 & \|\mathbf{x}\| \, \|\mathbf{y}\| \cos\theta_0 \\ \|\mathbf{x}\| \, \|\mathbf{y}\| \cos\theta_0 & \|\mathbf{x}\|^2 \end{bmatrix}.$$

Neural Nets: Universal Kernel

Arc-Cosine Kernel: More general weights [Tsuchida et al., 2018]

Convergence in distribution:

For any a.e continuous σ, under certain assumptions, with $\mathbf{W}^{(m)} \in \mathcal{R}^m$, iid, $\mathbb{E}(W_i) = 0$, and $E|W_i^3| < \infty$, we have

$$\sigma\left(\left\langle \mathbf{W}^{(m)}, \mathbf{x}^{(m)} \right\rangle\right) \sigma\left(\left\langle \mathbf{W}^{(m)}, \mathbf{y}^{(m)} \right\rangle\right) \xrightarrow[m\to\infty]{\text{d}} \sigma(Z_1)\sigma(Z_2),$$

where Z_1, Z_2 and Σ are as the non-asymptotic case.

Neural Nets: Universal Kernel

Arc-Cosine Kernel: More general weights [Tsuchida et al., 2018]

Convergence in expectation:

For ReLU/LReLU/ELU, under certain assumptions, with $\mathbf{W}^{(m)} \in \mathcal{R}^m$, iid, $\mathbb{E}(W_i) = 0$, and $E|W_i^3| < \infty$, we have

$$\mathbb{E}\left[\sigma\left(\left\langle \mathbf{W}^{(m)}, \mathbf{x}^{(m)} \right\rangle\right) \sigma\left(\left\langle \mathbf{W}^{(m)}, \mathbf{y}^{(m)} \right\rangle\right)\right] \xrightarrow[m \to \infty]{} \mathbb{E}\left(\sigma(Z_1)\sigma(Z_2)\right),$$

where Z_1, Z_2 and Σ are as the non-asymptotic case.

Deep Learning: Problems with Depth

LReLU: $\sigma(z) = (a + (1-a)\mathbf{1}_{z\geq 0})\, z,\ \ a \in [0,1]$

Kernel: $\kappa(\mathbf{x},\mathbf{y}) = \mathbb{E}[W^2]\, \|\mathbf{x}\|\, \|\mathbf{y}\| \left[\frac{(1-a)^2}{2\pi}\left(\sin\theta_0 + (\pi - \theta_0)\cos\theta_0\right) + a\cos\theta_0\right]$

Normalized Kernel: $\cos\theta_1 = \frac{\kappa(\mathbf{x},\mathbf{y})}{\sqrt{\kappa(\mathbf{x},\mathbf{x})\kappa(\mathbf{y},\mathbf{y})}} = f(\theta_0)$

Recursively applied:

$$\cos\theta_j = \frac{1}{1+a^2}\left[\frac{(1-a)^2}{2\pi}\left(\sin\theta_{j-1} + (\pi - \theta_{j-1})\cos\theta_{j-1}\right) + a\cos\theta_{j-1}\right]$$

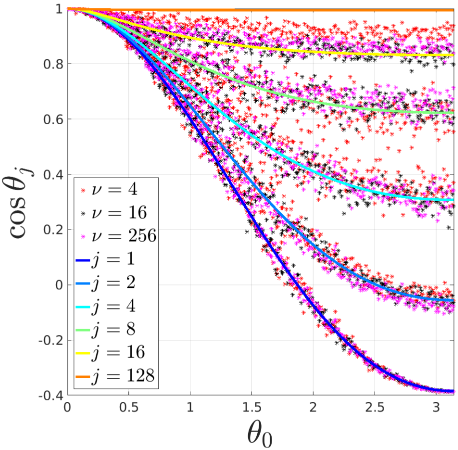
Deep Learning: KRAP

Kernelized Reducing Angle Problem (KRAP) [Tsuchida et al., 2018]

The normalized kernel corresponding to LReLU activations converges to a fixed point at $\theta^\star = 0$[a].

[a] Theoretically holds for rotationally-invariant weights and empirically holds for more general weights.

Deep Learning: KRAP



σ: ReLU, **w**: Multivariate t-distribution

ν: Degrees of freedom, j: Depth

Deep Learning: KRAP

Randomly initialized deep feedforward networks

- map all inputs to “similar” points in the Hilbert space
- erase all information in the input signal
- are hard to train (at least initially)

Deep Learning: Initialization

$$\frac{1}{n}\langle \boldsymbol{\sigma}_n(\mathbf{x}), \boldsymbol{\sigma}_n(\mathbf{y})\rangle \approx \kappa(\mathbf{x},\mathbf{y}) \Longrightarrow \|\boldsymbol{\sigma}_n(\mathbf{x})\| \approx \sqrt{n\kappa(\mathbf{x},\mathbf{x})}$$

$$\Longrightarrow \| \underbrace{\boldsymbol{\sigma}_n(\mathbf{x})}_{\substack{\approx\text{mapping}\\ \text{from}\\ \mathbf{x}\to\mathcal{H}_k}} \| \approx \|\mathbf{x}\| \sqrt{\frac{n\mathbb{E}[W^2](1+a^2)}{2}}$$

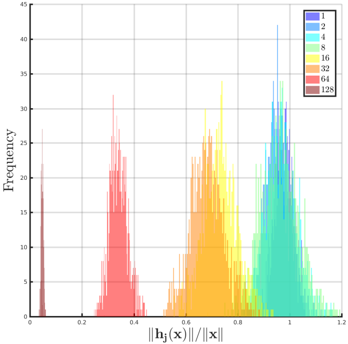
Initialization [Tsuchida et al., 2018]

Initialize from any rotationally-invariant weights with

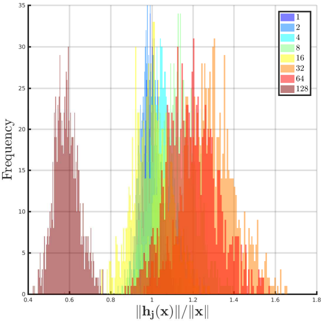
$$\mathbb{E}[W^2] = \frac{2}{(1+a^2)n}.$$

For $a = 0$, i.e., ReLU, this coincides with [He et al., 2015].

Deep Learning: Initialization...LReLU with $a = 0.2$



(f) [He et al., 2015]



(g) [Tsuchida et al., 2018]

Deep Learning: How about training?

How about training? Weights are no longer iid, etc!

Training NNs with ReLU (on arXiv soon):

- For certain class of optimization procedures, e.g., SGD
  - They maintain a certain invariance property, i.e.,
    - layer-wise kernel remains arc-cosine during training
    - full network's kernel remains approximately constant
- For others, e.g., Adam, RMSPorp
  - They exhibit a sharp phase transition as ϵ changes
- related to the "covariance between weights" (i.e., energy in each layer: the maximum of squared average of the weights connecting to each neuron)
- Relation to [Bartlett et al., 2017] and [Martin et al., 2017]?

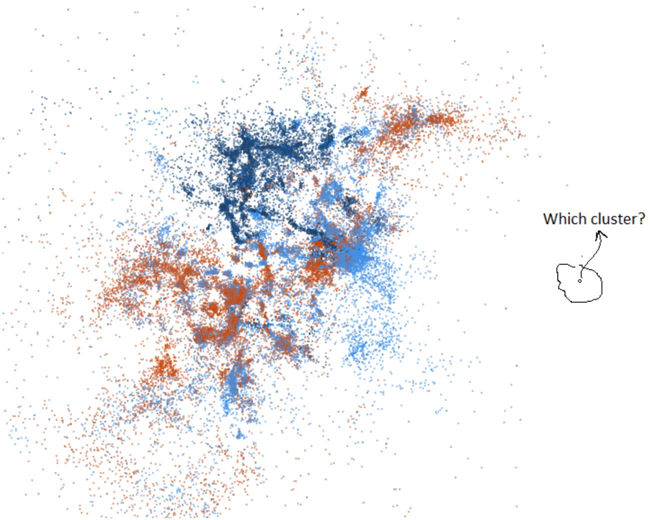
What Is Data Science?

How randomized analysis helps answer "data sciency" questions?

Examples:

- Deep Learning
- Graph Analysis

Graph Analysis



Graph Analysis

How To Analyze Graph Data?

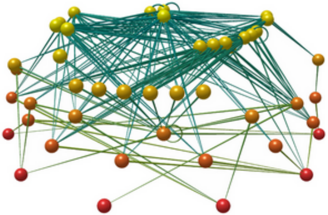
Graphs $\neq$ Data in classical statistics $\Longrightarrow$ Need new tools

Graph Analysis

Q: How To Analyze Graph Data?

Option 1: Graph-Specific Techniques

- Develop statistical/combinatorial/geometric model for graphs
- Develop machinery for that model
- Appealing, but lots of work!

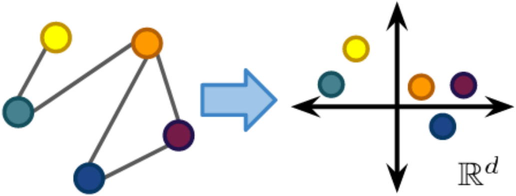


Graph Analysis

Q: How To Analyze Graph Data?

Option 1: Classical Techniques

- Graph $\Longrightarrow$ Classical Object, e.g., $\mathcal{S} \subseteq \mathcal{R}^d$
- Apply existing methods for classical, e.g., Euclidean, data
- Easier and also faster!



Graph Embedding

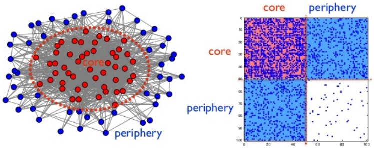
Graph Embedding

Graph $\Longrightarrow$ Classical Object

- Graph: $G = (V, E)$ on n vertices
- Find mapping $\mathcal{M} : G \to \mathcal{S} \subseteq \mathcal{R}^d$
- Such that "Geometry" is $\mathcal{S}$ reflects the "topology" of G

Graph Embedding

- Graph embedding using
  - Laplacian matrix
  - Adjacency matrix
- Both produce low-dimensional representations of V in G
- Which embedding to use?...Depends on the downstream task
  - Vertex Classification in SBM: Adjacency $\not\leq \not\geq$ Laplacian
  - Core-Periphery Graphs: Adjacency $\geq$ Laplacian



SHL, M. Cucuringu, and M. A. Porter, Phys. Rev. E **89**, 032810 (2014);
M. Cucuringu, M. P. Rombach, **SHL**, and M. A. Porter, e-print arXiv:1410.6572.

P. Csermely, A. London, L.-Y. Wu, and B. Uzzi, J. Complex Networks 1, 93 (2013);
M. P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha, SIAM J. App. Math **74**, 167 (2014).

Adjacency Spectral Embedding [Sussman et al, 2012]

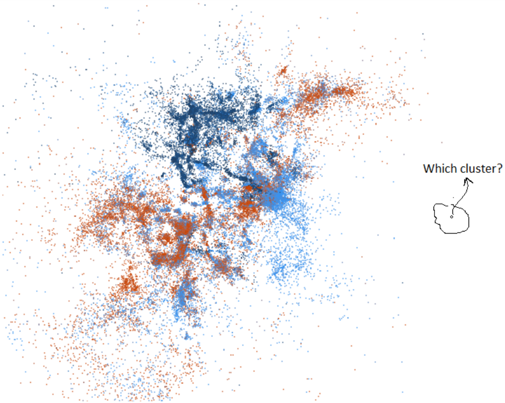
Adjacency Spectral Embedding (ASE)

- Adjacency matrix: $\mathbf{A} \in \{0, 1\}^{n \times n}$
- Eigen-decomposition: $\mathbf{A} = \mathbf{U}\boldsymbol{S}\mathbf{U}^T$
- $\boldsymbol{S}_d \in \mathcal{R}^{d \times d}$: Truncate $\boldsymbol{S} \in \mathcal{R}^{n \times n}$ by top d eigen-values
- $\mathbf{U}_d \in \mathcal{R}^{d \times d}$: Truncate $\mathbf{U} \in \mathcal{R}^{n \times n}$ by top d eigen-vectors
- ASE: $v_i \Longrightarrow i^{\text{th}}$ rows of $\mathbf{U}_d \boldsymbol{S}_d^{1/2} \in \mathcal{R}^{n \times d}$

Graph Embedding

Out-of-sample (OOS) Embedding for a Graph

- Suppose we already have $\mathscr{M} : G \Longrightarrow \mathcal{S} \subseteq \mathcal{R}^d$
- How to find an embedding for a new vertex v?



OOS Graph Embedding

Q: How to find an embedding for a new vertex v?

Option 1: Naive Approach

- $\mathscr{M} : G \Longrightarrow \mathcal{S} \subseteq \mathcal{R}^d$
- Discard the old embedding...and restart from stretch
- $\tilde{G} = (V \cup v, E \cup E_v)$
- $\mathscr{M}^+ : \tilde{G} \Longrightarrow \mathcal{S} \subseteq \mathcal{R}^d$
  - Expensive when $n \gg 1$
  - Similarity matrix $K \in \mathcal{R}^{n \times n}$ might no longer be available

OOS Graph Embedding

Q: How to find an embedding for a new vertex v?

Option 2: Leverage Existing Embedding

- $\mathscr{M} : G \Longrightarrow \mathcal{S} \subseteq \mathcal{R}^d$
- Use the old embedding $\mathcal{M}$...
- $\widetilde{\mathscr{M}}(v; \mathcal{M}) \Longrightarrow \hat{\mathbf{w}} \in \mathcal{S} \subseteq \mathcal{R}^d$
  - Fast specially when $n \gg 1$
  - But how accurate is this OOS embedding?

$$\widetilde{\mathscr{M}}(v; \mathscr{M}) \overset{?}{\approx} \mathscr{M}^+(v)$$

  - Statistics helps us study this question...

Edge Independent Random Graphs

Random Dot-Product Graphs, [Young and Scheinerman, 2007]

- $\forall v \in V \implies \mathbf{x}_i \in \mathcal{X} \subseteq \mathcal{R}^d$
- $\mathcal{X}$: Latent Space
- $\mathcal{X}$: Need not be finite
- $\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X} : \langle \mathbf{x}_i, \mathbf{x}_j \rangle \in [0, 1]$
- $\forall v_i, v_j \in V,\ \mathbf{Pr}((v_i, v_j) \in E) = p_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$

RDPG [Young and Scheinerman, 2007]

We can consider a distribution on $\mathcal{X}$...

RDPG: General Definition

- $\mathcal{X}$: Latent Space s.t. $\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X} : \langle \mathbf{x}_i, \mathbf{x}_j \rangle \in [0, 1]$
- F: distribution on $\mathcal{X}$
- $\{\mathbb{x}_1, \mathbb{x}_2, \ldots, \mathbb{x}_n\} \subset \mathcal{R}^d \overset{\text{iid}}{\sim} F$
- $\mathbb{X} = [\mathbb{x}_1, \mathbb{x}_2, \ldots, \mathbb{x}_n]^T \in \mathcal{R}^{n \times d}$
- Adjacency Matrix: $\mathbb{A} \in \{0, 1\}^{n \times n}$

$$\mathbf{Pr}\left(\mathbb{A} \mid \mathbb{X}\right) = \prod_{1 \leq i < j \leq n} \left(\langle \mathbb{x}_i, \mathbb{x}_j \rangle \right)^{\mathbb{A}_{ij}} \left(1 - \langle \mathbb{x}_i, \mathbb{x}_j \rangle \right)^{1 - \mathbb{A}_{ij}}$$

- $(\mathbb{A}, \mathbb{X}) \sim \text{RDPG}(F, n)$

RDPG [Young and Scheinerman, 2007]

RDPG: Inherent Nonidentifiability

- $\mathbb{X} \in \mathcal{R}^{n \times d}$
- For any orthonormal matrix $\mathbf{Q} \in \mathcal{R}^{d \times d} \implies \mathbb{X}\mathbf{Q} \in \mathcal{R}^{n \times d}$
- $\mathbb{X}\mathbb{X}^T = (\mathbb{X}\mathbf{Q})(\mathbb{X}\mathbf{Q})^T = \mathbb{E}[\mathbb{A} \mid \mathbb{X}]$
- $\mathbf{Pr}(\mathbb{A} \mid \mathbb{X}) = \mathbf{Pr}(\mathbb{A} \mid \mathbb{X}\mathbf{Q})$

ASE on RDGP

Adjacency Spectral Embedding (ASE):

$$\mathbb{A} = \mathbb{USU} \Longrightarrow \widehat{\mathbb{X}} = \mathbb{U}_d \mathbb{S}_d^{1/2} \in \mathcal{R}^{n \times d}$$

Theorem (Lyzinski et al., 2014)

With probability of at least $1 - c/n^2$, there exists an orthogonal matrix $\mathbb{Q} \in \mathcal{R}^{d \times d}$ for which

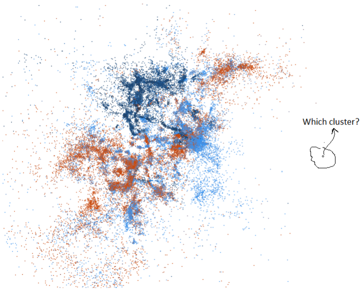
$$\|\widehat{\mathbb{X}} - \mathbb{X}\mathbb{Q}\|_{2\to\infty} = \max_{1 \le i \le n} \|\widehat{\mathbf{x}}_i - \mathbb{Q}\mathbf{x}_i\| \le cn^{-1/2} \log n.$$

Out-of-sample (OOS) Embedding

Recall:

Out-of-sample (OOS) Embedding for a Graph

- We only have $\widehat{\mathbf{X}}$, i.e., no longer have $\mathbf{A}$, etc...
- We are given a new vertex v with the edges incident on it $\mathbf{a}_v$
- How do we embed v?



OOS for ASE: Linear Least Squares Approach

LS-OOS [Levin et al., 2018]

$$\hat{\mathbb{x}}_v = \arg\min_{\mathbf{y}\in\mathcal{R}^d} \left\| \widehat{\mathbb{X}}\mathbf{y} - \mathbf{a}_v \right\|^2$$

- $\widehat{\mathbb{X}} \in \mathcal{R}^{n\times d}$: Estimator of the true latent positions $\mathbb{X}$
- $\mathbf{a}_v \in \{0,1\}^n$: Random vector for the edges incident on v

OOS for ASE: Maximum Likelihood Approach

ML-OOS [Levin et al., 2018]

$$\hat{\mathbf{x}}_v = \arg\max_{\mathbf{y} \in \mathcal{R}^d} \sum_{i=1}^{n} \mathbf{a}_v[i] \log\left(\langle \hat{\mathbf{x}}_i, \mathbf{y} \rangle\right) + (1 - \mathbf{a}_v[i]) \log\left(1 - \langle \hat{\mathbf{x}}_i, \mathbf{y} \rangle\right)$$

- $\hat{\mathbf{x}}_i \in \mathcal{R}^d$: Estimator of the true latent position $\mathbf{x}_i$
- $\mathbf{a}_v[i] \in \{0, 1\}$: Random variable for the edge between (v, v_i)
- $\mathbf{a}_v[i] \sim \text{Bernoulli}(\langle \mathbf{x}_i, \mathbf{x}_v \rangle)$

OOS for ASE, [Levin et al., 2018]

Recall:

Theorem (Lyzinski et al., 2014)

With probability of at least $1 - c/n^2$, there exists an orthogonal matrix $\mathbb{Q} \in \mathcal{R}^{d \times d}$ for which

$$\|\widehat{\mathbb{X}} - \mathbb{X}\mathbb{Q}\|_{2\to\infty} = \max_{1 \le i \le n} \|\widehat{\mathbb{x}}_i - \mathbb{Q}\mathbb{x}_i\| \le cn^{-1/2} \log n$$

.

Theorem (Levin et al., 2018)

Let $\mathbf{x}_v \in Supp(F)$. For both methods, w.h.p, we have

$$\|\hat{\mathbb{x}}_v - \mathbb{Q}\mathbf{x}_v\| \le cn^{-1/2} \log n,$$

where $\mathbb{Q}$ is the same as given in [Lyzinski et al., 2014].

OOS for ASE, [Levin et al., 2018]

Theorem (CLT for LLS OOS, Levin et al., 2018)

Given the true latent position $\mathbf{x}_v$*, we have*

$$\sqrt{n}\,(\hat{\mathbb{x}}_v - \mathbb{Q}_n \mathbf{x}_v) \xrightarrow[n\to\infty]{d} \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{x}_v}),$$

where

$$\boldsymbol{\Sigma}_{\mathbf{x}_v} = \boldsymbol{\Delta}_1^{-1} \mathbb{E}\left[\langle \mathbb{x}_1, \mathbf{x}_v \rangle \Big(1 - \langle \mathbb{x}_1, \mathbf{x}_v \rangle \Big) \mathbb{x}_1 \mathbb{x}_1^T \right] \boldsymbol{\Delta}^{-1},$$

and $\boldsymbol{\Delta} = \mathbb{E}(\mathbb{x}\mathbb{x}_1^T)$.

OOS for ASE, [Levin et al., 2018]

Theorem (CLT for LLS OOS, Levin et al., 2018)

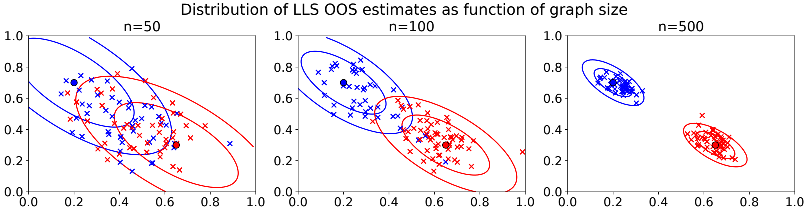
Suppose $(\mathbb{A}, \mathbb{X}) \sim RDPG(F, n)$ *and, independently, the true latent position* $\mathbb{x}_v \sim F$*, we have*

$$\sqrt{n}\,(\hat{\mathbb{x}}_v - \mathbb{Q}_n \mathbb{x}_v) \xrightarrow[n\to\infty]{d} \int \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_\mathbf{x})dF(\mathbf{x}),$$

where $\mathbf{\Sigma}_\mathbf{x}$ *is as before.*

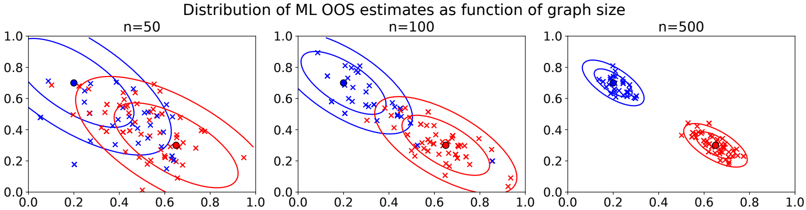
Experiments: How fast does CLT kick in?

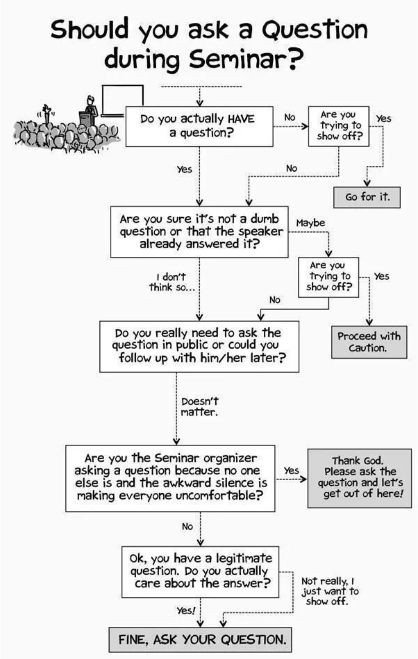
- $n + 1$ latent positions drawn iid
  $F = 0.4 \cdot (0.2, 0.7)^T + 0.6 \cdot (0.65, 0.3)^T$
- Embed first n vertices via ASE
- Apply LS OOS extension to vertex $n + 1$, correct for non-identifiability
- Repeat 100 trials, plot 100 OOS estimates
- CLT predicts mixture of normals (indicated by isoclines)



Experiments: What about the ML OOSE?

- $n + 1$ latent positions drawn iid $F = 0.4 \cdot (0.2, 0.7)^T + 0.6 \cdot (0.65, 0.3)^T$
- Embed first n vertices via ASE
- Apply ML OOS extension to vertex $n + 1$, correct for non-identifiability
- Repeat 100 trials, plot 100 OOS estimates
- CLT predicts mixture of normals (indicated by isoclines)




Should you ask a Question during Seminar?
Do you actually HAVE a question?
No
Are you trying to show off?
Yes
Go for it.
No
Yes
Are you sure it's not a dumb question or that the speaker already answered it?
Maybe
Are you trying to show off?
Yes
Proceed with Caution.
No
I don't think so...
Do you really need to ask the question in public or could you follow up with him/her later?
Doesn't matter.
Are you the Seminar organizer asking a question because no one else is and the awkward silence is making everyone uncomfortable?
Yes
Thank God. Please ask the question and let's get out of here!
No
Ok, you have a legitimate question. Do you actually care about the answer?
Not really, I just want to show off.
Yes!
FINE, ASK YOUR QUESTION.

THANK YOU!