Maximum Entropy, Spectra of Graphs & Understanding the loss surface of DNN's with Spectral Techniques

D. Granziol[1,2]

[1]Machine Learning Research Group
Information Engineering
University of Oxford

[2]Oxford-Man Institute of Quantitative Finance
University of Oxford

December 10, 2018

Outline

Outline

Outline

Outline

Outline

Outline

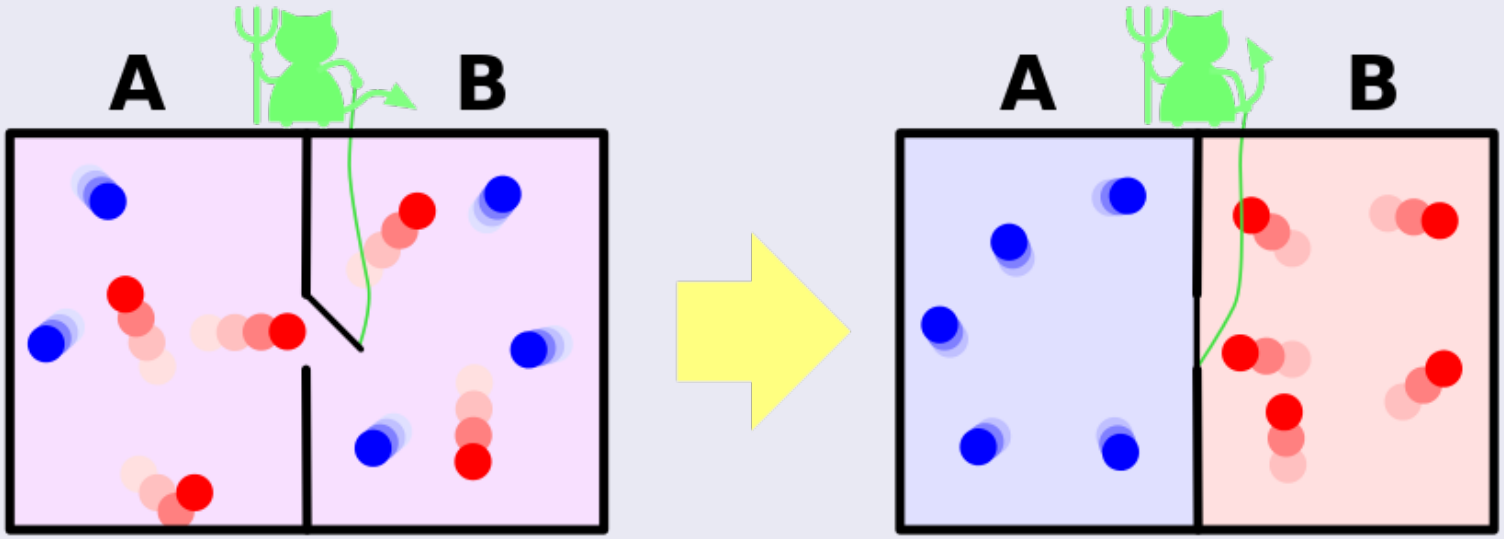
Outline

Outline

Outline

Outline

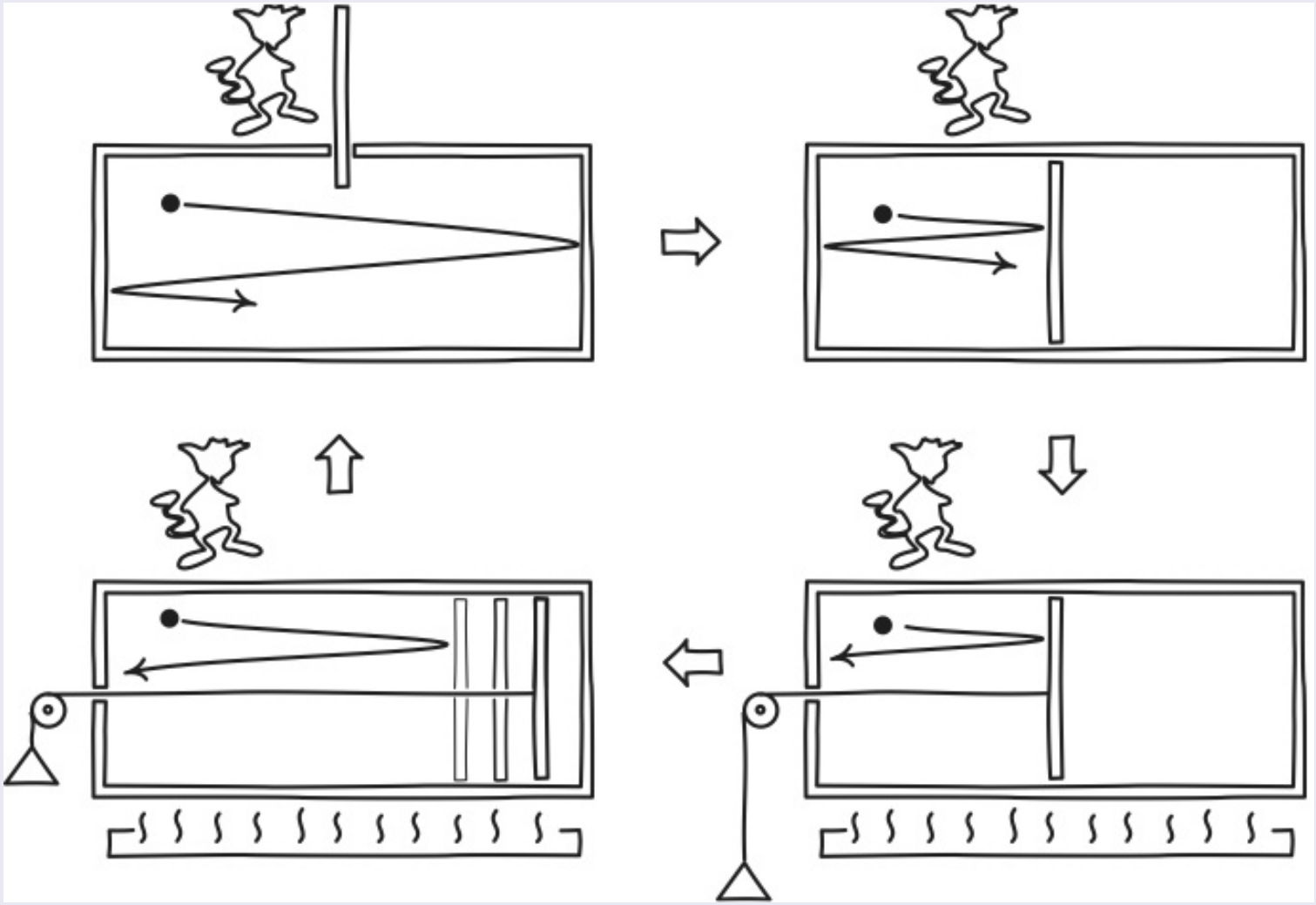
Maxwells Demon



Consequences

- Violate the Second Law of Thermodynamics
- Move heat from cold to hot

Szilard's engine - 1929



Shannon Entropy

Axioms

Axiom 1. S is a real continuous function of the probabilities.

Axiom 2. If all the p_i's are equal then $S = F(n)$ which is a monotonically increasing function of n.

Axiom 3. For all possible groupings g =1...N of the states i=1...n we must have

$$S[P] = S_g[P] + \sum_g P_g S_g[p|g] \tag{1}$$

How we label and sub-group the states does not affect entropy

Conclusion

$$S_G[P] = -k \sum_{i=1}^{N} P_g log P_g \tag{2}$$

Proof is well known and hence omitted

Maximum Entropy

Edwin Jaynes

Maximise the entropic functional $\mathcal{S}$ w.r.t $p(x)$ given the constraints

$$S = -\sum_{x\in\mathbb{D}} p(x)\log p(x)dx - \alpha\left[\sum_{x\in\mathbb{D}} p(x) - 1\right] - \sum_i \lambda_i\left[\sum_{x\in\mathbb{D}} p(x)f_i(x)dx - \mu_i\right] \tag{3}$$

Properties

- Johnson and Shore Axioms
- Least biased estimate
- Wide range of applications

Example

$$\mathrm{S} = -\int p(x)\log p(x)dx - \alpha\left[\int p(x)dx - 1\right] - \beta\left[\int p(x)xdx - \mu\right]$$

Deriving Bayes Rule from Maximum Relative Entropy

PMU - the principle of minimal update

Bayes rule is often quoted as a restatement of the product rule.

$$q(\theta|x') = q(\theta)\frac{q(x'|\theta)}{q(x')} \tag{4}$$

This formula is not a posterior, but a prior conditional probability. We invoke the Principle of Minimal updating, which stipulates that the web of beliefs need only be revised to the extent required by new data.
All distributions of the form

$$p(\theta, x') = \delta(x - x')p(\theta|x') \tag{5}$$

fit the data, we invoke the PMU by saying no further revision is needed and that $p(\theta|x') = q(\theta|x')$ thus

$$p(\theta) = \int dx\delta(x - x')q(\theta|x') = q(\theta|x') = q(\theta)\frac{q(x'|\theta)}{q(x')} \tag{6}$$

Deriving Bayes Rule from Maximum Relative Entropy

PMU - the principle of minimal update

$$S[p,q] = -\int dx d\theta p(x,\theta) \log \frac{p(x,\theta)}{q(x,\theta)}$$

$$\delta\{S + \alpha[\int dx d\theta p(x,\theta) - 1] + \int dx \lambda(x)[\int d\theta p(x,\theta) - \delta(x - x')]\} = 0$$

$$p(x,\theta) = q(x,\theta)\frac{e^{\lambda(x)}}{Z}$$

$$\int p(x,\theta)d\theta = p(x) = \delta(x - x') = q(x)\frac{e^{\lambda(x)}}{Z} \rightarrow e^{\lambda(x)} = \delta(x - x')\frac{Z}{q(x)}$$

$$P(x,\theta) = \delta(x - x')q(\theta|x) \rightarrow p(\theta) = q(\theta|x') = q(\theta)\frac{q(x'|\theta)}{q(x')} \tag{7}$$

Entropic Trace Estimation - Accepted ECML!

What is the problem?

Calculating the Log Determinant of a Matrix A is an $\mathcal{O}(n^3)$ problem.
Bottleneck for many machine learning algorithms.
Can we approximate quickly and accurately?
We know the first two eigenvalue raw moments exactly.

$$\mathbb{E}(\lambda) = (1/n)\,Tr(A)\,,\ \mathbb{E}(\lambda^2) = (1/n)\sum_{i,j} a_{i,j}^2 \tag{8}$$

Can get noisy estimates of $\mathbb{E}(\lambda^m)$ using Stochastic trace estimation in $\mathcal{O}(n^2)$

$$\mathbb{E}(\lambda^m) = (1/n)\,Tr(A^m) \tag{9}$$

$$\begin{aligned}(1/n)\,Tr(A^m) &= (1/n)\,Tr(U^t A^m U)\,(UU^t = I)\\ &= (1/n)\,Tr(D^m) = (1/n)\sum_{i}^{n}\lambda^m\end{aligned} \tag{10}$$

Entropic Trace Estimation - Accepted ECML!

Stochastic Trace Estimation

Consider a random variable z with mean m and variance σ using the property of the expectations of quadratic forms, the expectation

$$\mathbb{E}[zz^t] = \sigma + mm^t = I \tag{11}$$

Where we have assumed that the variable is zero-mean and unit-variance. This allows us to calculate the trace of any matrix power A^m as

$$\begin{aligned} Tr(A^m) &= Tr(A^m I) = Tr(A^m \mathbb{E}[zz^t] \\ &= \mathbb{E}[Tr(A^m zz^t)] = \mathbb{E}[z^t A^m z] \approx= \frac{1}{d}\sum_{i=1}^{d} z_i^t A....A z_i \end{aligned} \tag{12}$$

E.g z is a gaussian, total computational cost $\mathcal{O}(nnz * m * d)$

Entropic Trace Estimation - Accepted ECML!

Combine with Maximum Entropy

$$\log\det A = \log \Pi_i \lambda_{i=1}^{n}$$

$$\sum_i \log \lambda_i = n\frac{1}{n}\sum_{i=1}^{n} \log \lambda_i \tag{13}$$

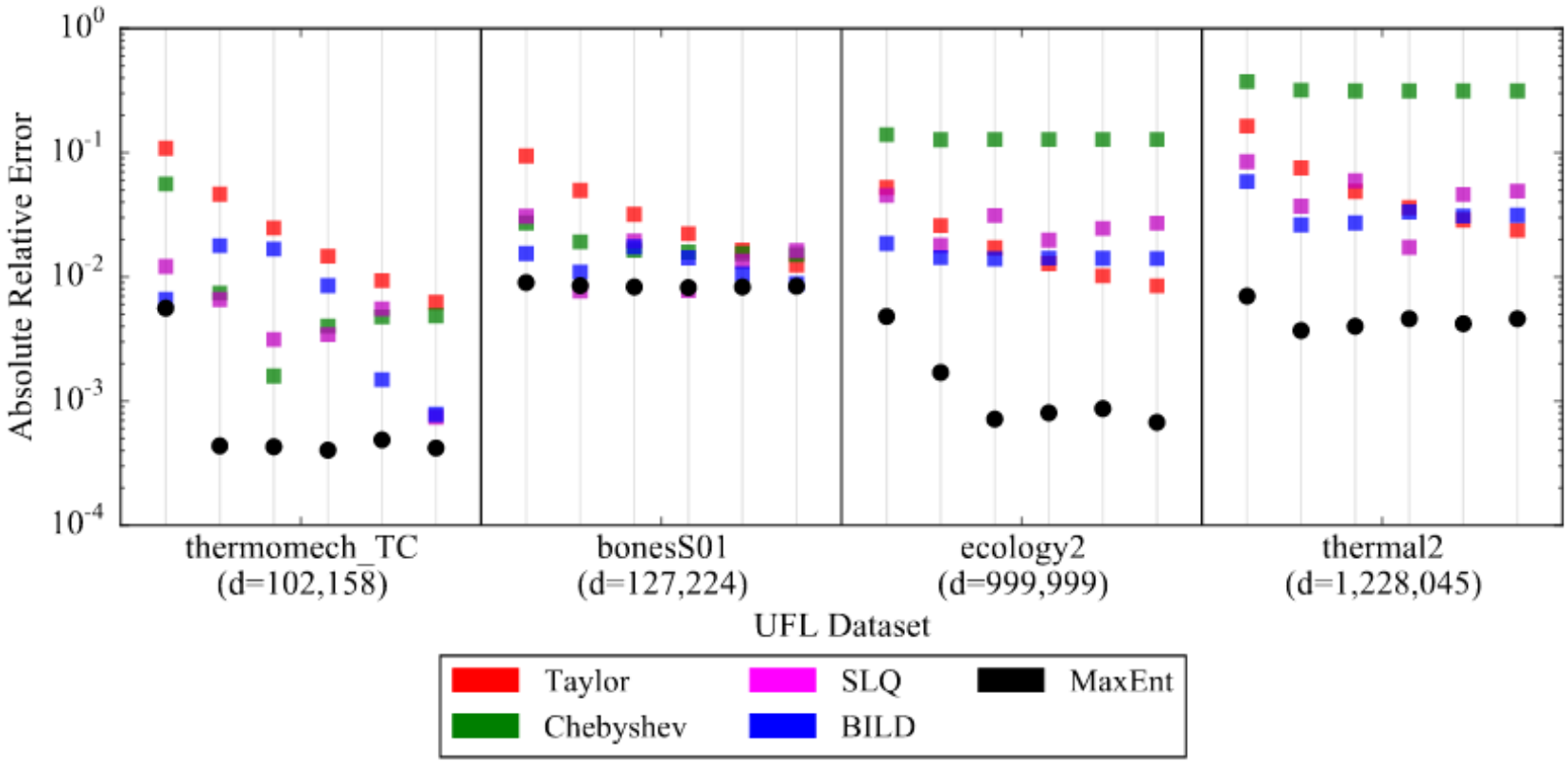
$$\rightarrow n\mathbb{E}_p(\log \lambda) \approx n\mathbb{E}_q(\log \lambda)$$

Where the measure under which we evaluate the expectation is the MaxEnt distribution

$$q(\lambda) = \exp(\sum_{i=0}^{m} \alpha_i \lambda^i) \tag{14}$$

Entropic Trace Estimation - Accepted ECML!

Results



Entropic Trace Estimation - Accepted ECML!

Results

Dataset	Dimension	Taylor	Chebyshev	SLQ	BILD	MaxEnt
shallow_water1	81,920	**0.0023**	0.7255	0.0058	0.0163	0.0030
shallow_water2	81,920	0.5853	0.9846	0.9385	1.1054	**0.0051**
apache1	80,800	0.4335	0.0196	0.4200	0.1117	**0.0057**
finan512	74,752	0.1806	0.1158	0.0142	**0.0005**	0.0171
obstclae	40,000	0.0503	0.5269	0.0423	0.0733	**0.0026**
jnlbrng1	40,000	0.1084	0.2079	0.0465	0.0805	**0.0158**

Entropic Spectral Learning

Graph Theory

Let $G = (V, E)$ be an undirected graph with vertex set $V = v_1, ...v_N$
Edge between vertices v_i and v_j carries weight $w_{ij} > 0$

Graphs as Matrices

- The **Adjacency matrix** is defined as $W = (w_{ij})$ with $i, j = 1, ...n$.
- The degree of a vertex $v_i \in V$ is defined as $d_i = \sum_{j=1}^{n} w_{ij}$
- The **degree matrix** D is defined as a diagonal matrix that contains the degrees of the vertices along diagonal, i.e., $D_{ii} = d_i$.
- The **unnormalised graph Laplacian** is defined as $L = D - W$
- We work with **normalised graph Laplacian** $0 \leq \lambda_i \leq 2 \ \forall i$

$$L_{\text{norm}} \quad = \quad D^{-1/2} L D^{-1/2} \tag{15}$$

What about Graph Spectra?

Issue with Relative Divergence

$$p(\lambda) = \frac{1}{n}\sum_{i}^{n}\delta(\lambda - \lambda_i) \approx \sum_{i}^{m} w_i \delta(\lambda - \lambda_i) \tag{16}$$

Relative Entropy is infinite for densities that are mutually singular.

$$\mathbb{D}_{kl}(p||q) = \int_{\lambda\in\mathcal{D}} p(\lambda)\log\frac{p(\lambda)}{q(\lambda)} \tag{17}$$

- an infinite (or maximal) divergence upon the removal/addition of a single edge in a large network
- an infinite (or maximal) divergence upon the removal/addition of a single node in a large network
- an infinite (or maximal) divergence between two randomly generated graphs with identical hyper-parameters

What about Graph Spectra?

Kernel Smoothing is Moment Destroying

To alleviate some of the problems presented previously for spectral divergences, authors typically convolve the dirac mixture with a smooth kernel

$$\int K(\lambda - \lambda', s) \sum_i^n w_i \delta(\lambda' - \lambda_i) d\lambda' = \sum_i^n w_i K(\lambda - \lambda_i, s) \tag{18}$$

The moments of the dirac mixture, are given as

$$\langle \lambda^m \rangle = \sum_i^n w_i \int_{\lambda \in \mathcal{D}} \delta(\lambda - \lambda_i) \lambda^m d\lambda = \sum_i^n w_i \lambda_i^m \tag{19}$$

For simplicity we assume the kernel function to be symmetric, defined on the real line and permit all moments, these conditions are satisfied for the commonly used Gaussian kernel.

What about Graph Spectra?

Kernel Smoothing is Moment Destroying

$$\begin{aligned}\langle \tilde{\lambda}^m \rangle &= \sum_i w_i \int_{-\infty}^{\infty} K(\lambda - \lambda_i, s)\lambda^m d\lambda \\ &= \sum_i w_i \int_{-\infty}^{\infty} K(\lambda', s)(\lambda' + \lambda_i)^m d\lambda' \\ &= \langle \lambda^m \rangle + \sum_i^n w_i \int_{-\infty}^{\infty} \sum_{2j=2}^{2j=r} \binom{r}{2j} \mathbb{E}_{K(\lambda,s)}(\lambda^{2j}) \lambda_i^{m-2j}\end{aligned} \tag{20}$$

Where $\mathbb{E}_{K(\lambda,s)}(\lambda^{2j})$ denotes the $2j$'th central moment of the kernel function $K(\lambda, s)$. We have used the fact that the infinite domain is invariant under shift re-paramatrisation and that the odd moments of any symmetric distribution are 0. The sum over j is up to r which is m if m is even and $m-1$ if m is odd.

What about Graph Spectra?

How informative are the moments of the underlying process?

For matrices with independent entries a_{ij}, $\forall i > j$, with common element wise bound K, common expectation μ, variance σ^2 and diagonal expectation $\mathbb{E}a_{ii} = \nu$, it can be shown that the corrections to the semi-circle law for the moments of the eigenvalue distribution

$$\langle \mu_n, \lambda^k \rangle = \int_{\mathcal{R}} \lambda^k d\mu(x) = \frac{1}{n} \mathrm{Tr} X_n^k \tag{21}$$

have a corrective factor bounded by

$$\frac{K^2 k^6}{2\sigma^2 n^2} \tag{22}$$

hence, the finite size effects are larger for higher moments than the lower counterparts.

What about Graph Spectra?

How informative are the moments of the underlying process?

spectral moments of a random matrix and their expectation under the generating process, $\langle \hat{\mu}_n, x^k \rangle$.

$$P\left(|\langle \mu_n, x^k \rangle - \langle \hat{\mu}_n, x^k \rangle| > \epsilon\right) \leq \frac{1}{\epsilon^2} \left| \mathbf{E}\left(\langle \mu_n, x^k \rangle\right)^2 - \left(\mathbf{E}\langle \mu_n, x^k \rangle\right)^2 \right| \tag{23}$$

$$\frac{1}{n^2}\left[\mathbf{E}((\mathrm{Tr}X_n^k)^2 - (\mathbf{E}\mathrm{Tr}X_n^k)^2\right] = \frac{1}{n^2}\sum_{i,i'}\left[\mathbf{E}\zeta_i\zeta_{i'} - \mathbf{E}\zeta_i\mathbf{E}\zeta_{i'}\right] \tag{24}$$

where ζ_i is shorthand for the product $\zeta_{i_1 i_2}...\zeta_{i_k i_1}$ with $i_1, ..., i_k \in \{1, ..., n\}$, where each pair i, i' generates a graph with vertices $V_{i,i'} = \{i_1, ..i_k\} \cup \{i'_1, ..., i'k\}$ and edges $E_{i,i'} = \{i_1 i_2, .., i_k i_1\} \cup \{i'_1 i'_2, ..., i'_k i'_1\}$.

If you want the proofs - "Methods of Proof in Random Matrix Theory" Harvard

What about Graph Spectra?

How informative are the moments of the underlying process?

It can be easily shown that there are no non-zero pairs of weights $t \geq k+1$. By the boundedness of the moments of the entries of M_n the contribution of each term is $\mathcal{O}(1/n^{k+2})$ so the greatest corrective term for large n is $t = k$ for which there are equivalent pairs

$$w = n(n-1)..(n-t+1) = \frac{n!}{(n-t)!} \tag{25}$$

using Stirlings approximation so the total correction goes as

$$\sum_{t=1}^{k} \frac{1}{n^{k-t+2}} \exp\left(n \sum_{i=2}^{\infty} \left(\frac{t}{n}\right)^{i} \frac{1}{i(i+1)}\right) \tag{26}$$

What about Graph Spectra?

How informative are the moments of the underlying process?

It can be easily shown that there are no non-zero pairs of weights $t \geq k+1$. By the boundedness of the moments of the entries of M_n the contribution of each term is $\mathcal{O}(1/n^{k+2})$ so the greatest corrective term for large n is $t = k$ for which there are equivalent pairs

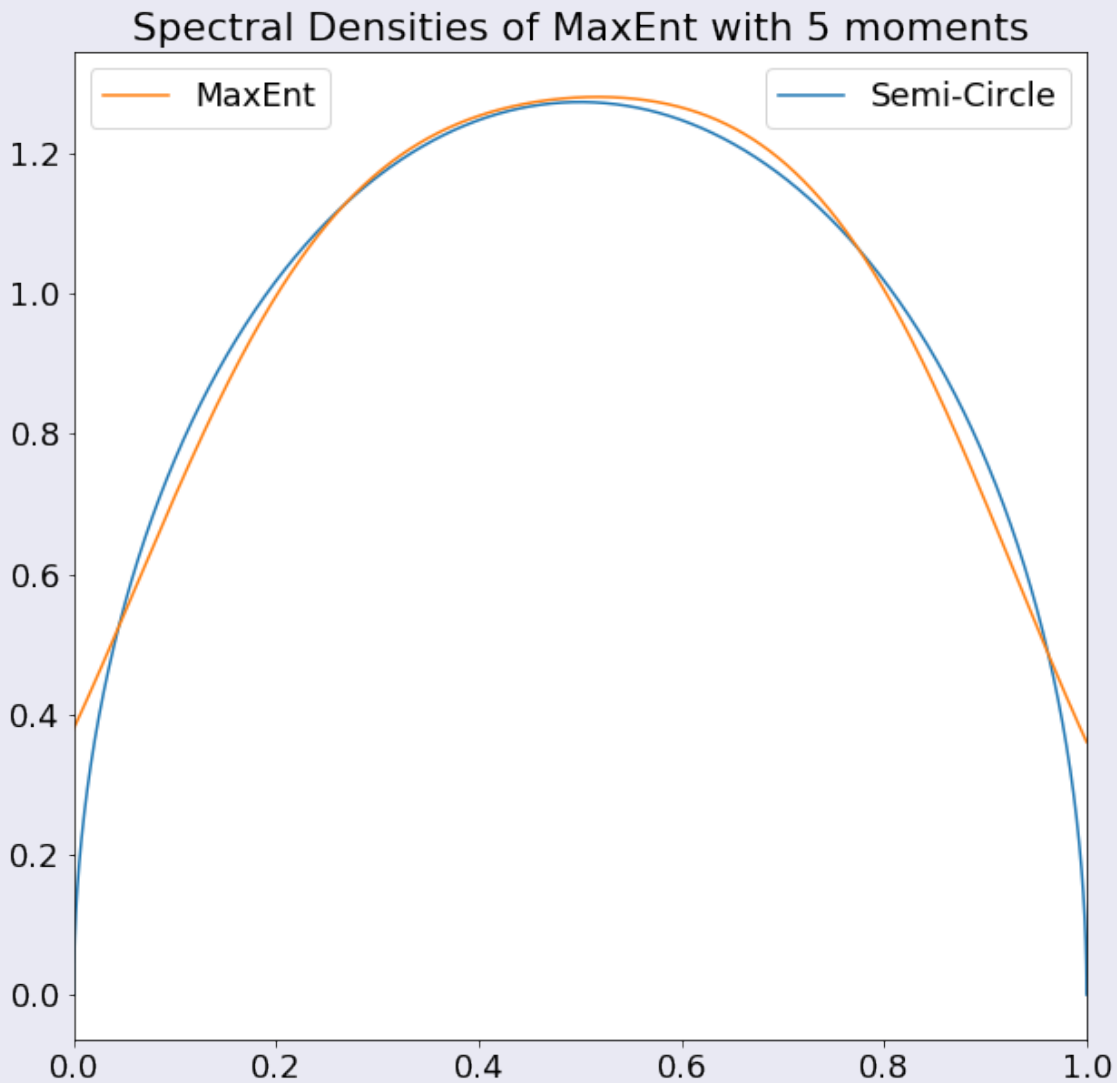
$$w = n(n-1)..(n-t+1) = \frac{n!}{(n-t)!} \tag{27}$$

using Stirlings approximation so the total correction goes as

$$\sum_{t=1}^{k} \frac{1}{n^{k-t+2}} \exp\left(n \sum_{i=2}^{\infty} \left(\frac{t}{n}\right)^{i} \frac{1}{i(i+1)} \right) \tag{28}$$

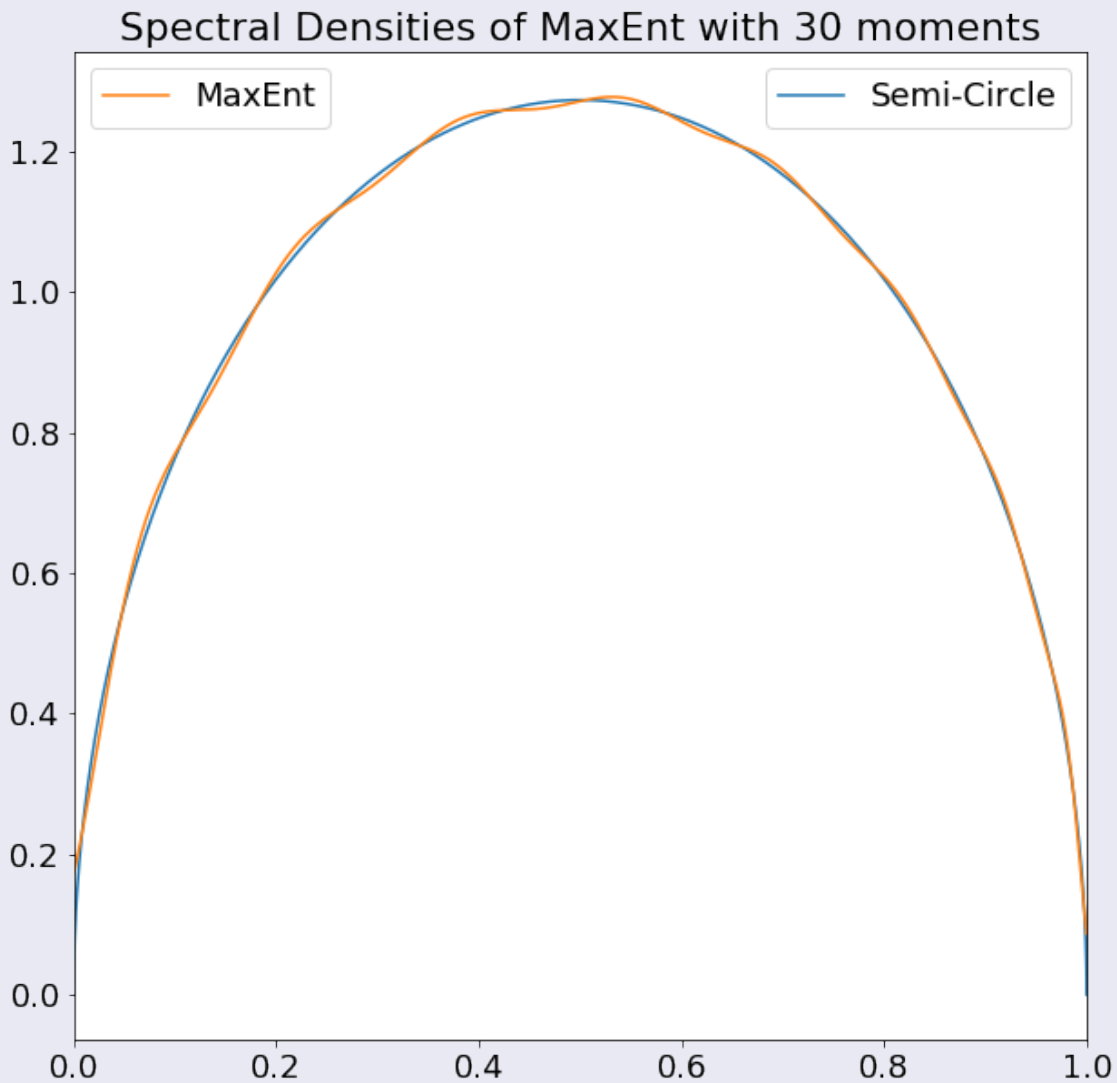
What about Graph Spectra?

Semi-Circle - MaxEnt with 5 moments



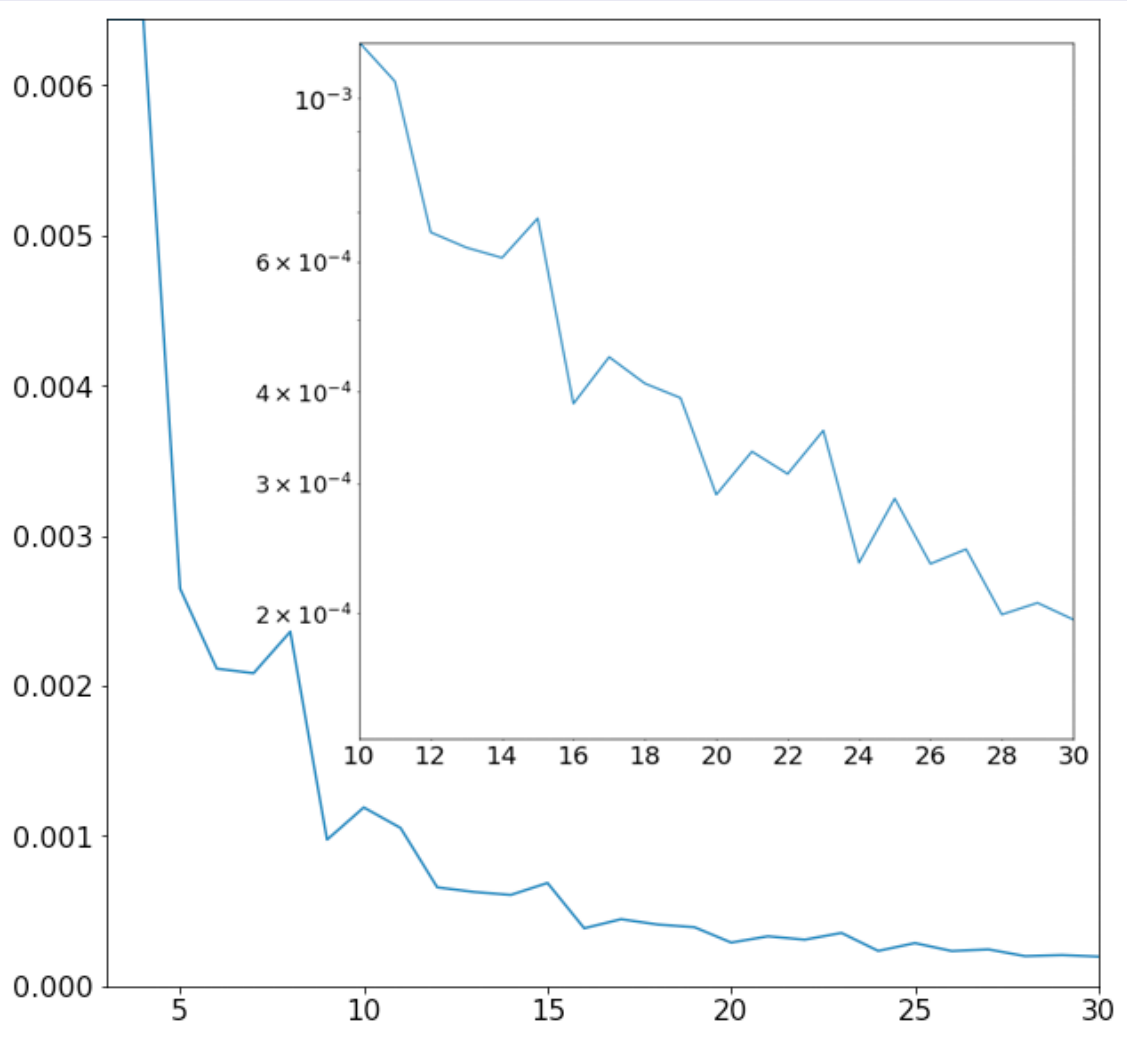
What about Graph Spectra?

Semi-Circle - MaxEnt with 30 moments



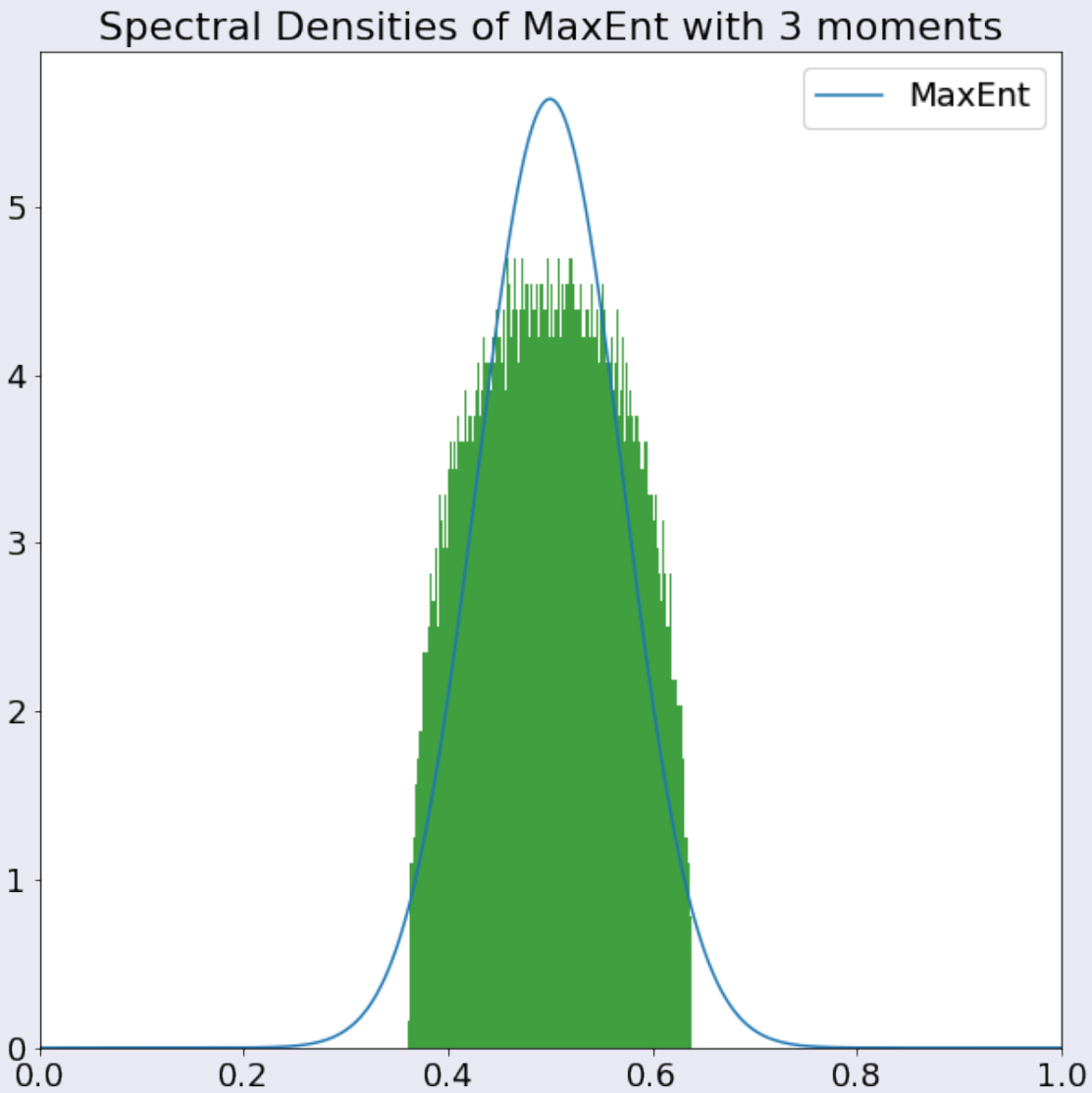
What about Graph Spectra?

Semi-Circle - KL divergence



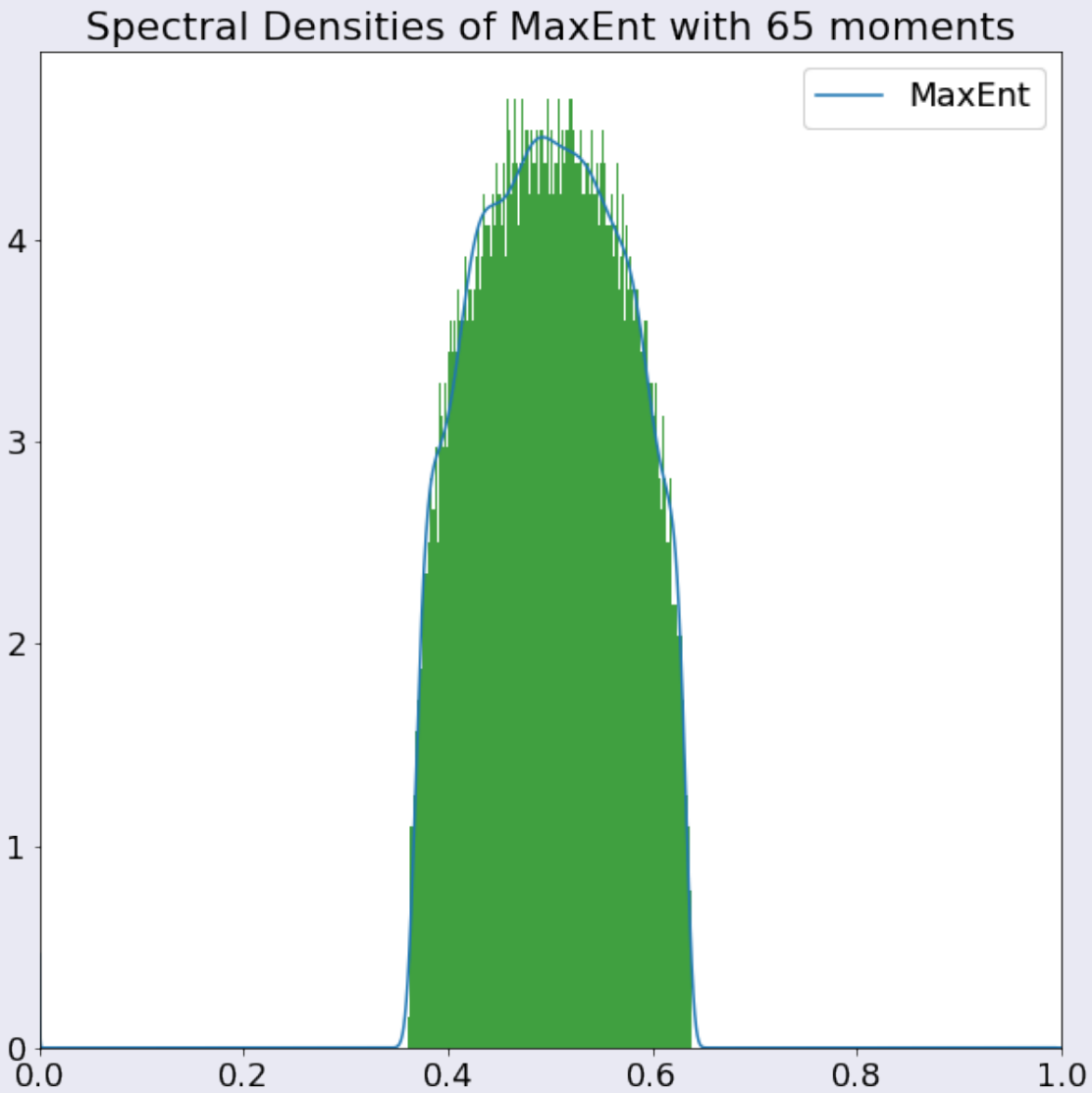
What about Graph Spectra?

Erdos Renyi - 3 moments



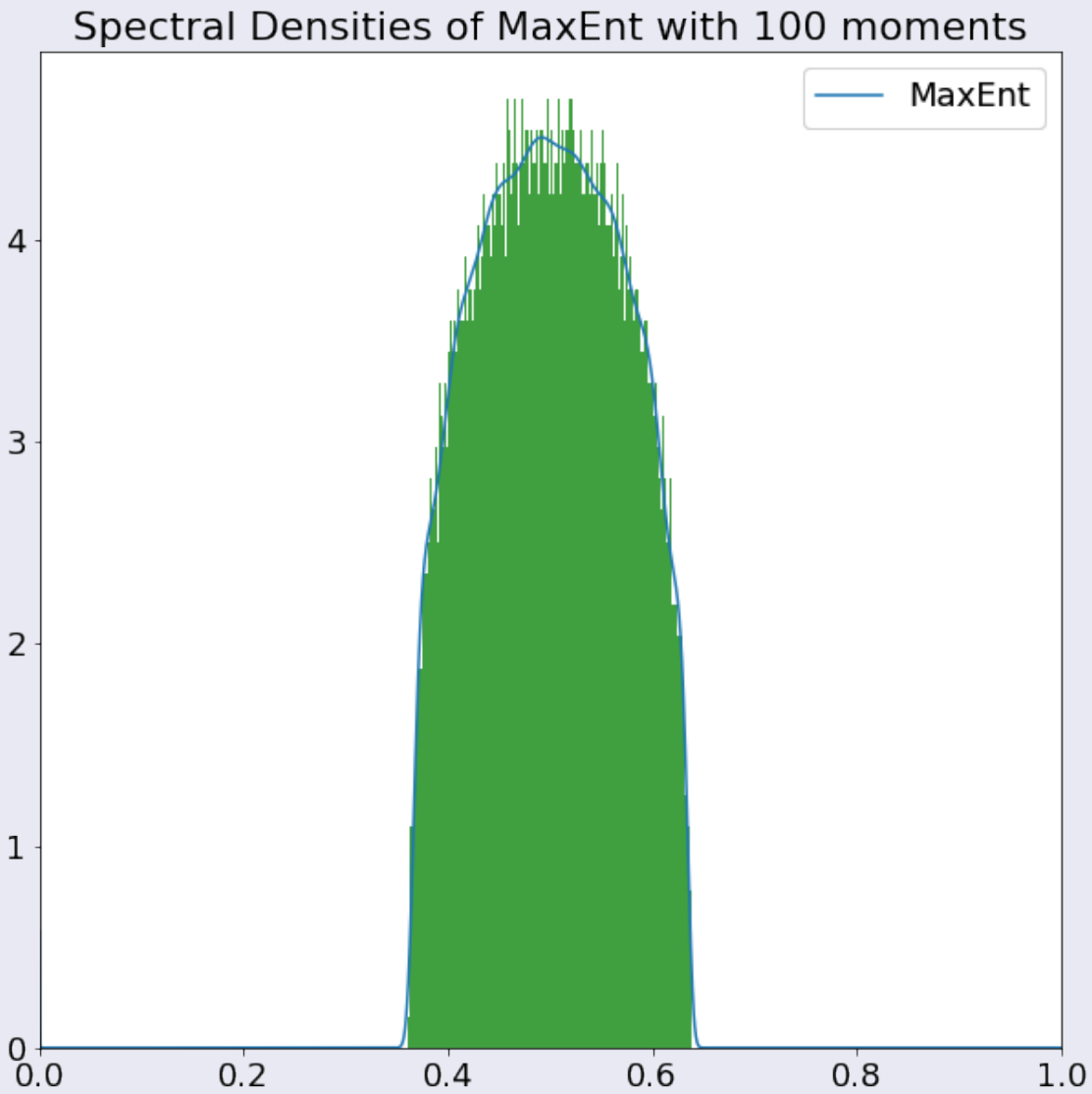
What about Graph Spectra?

Erdos Renyi - 65 moments



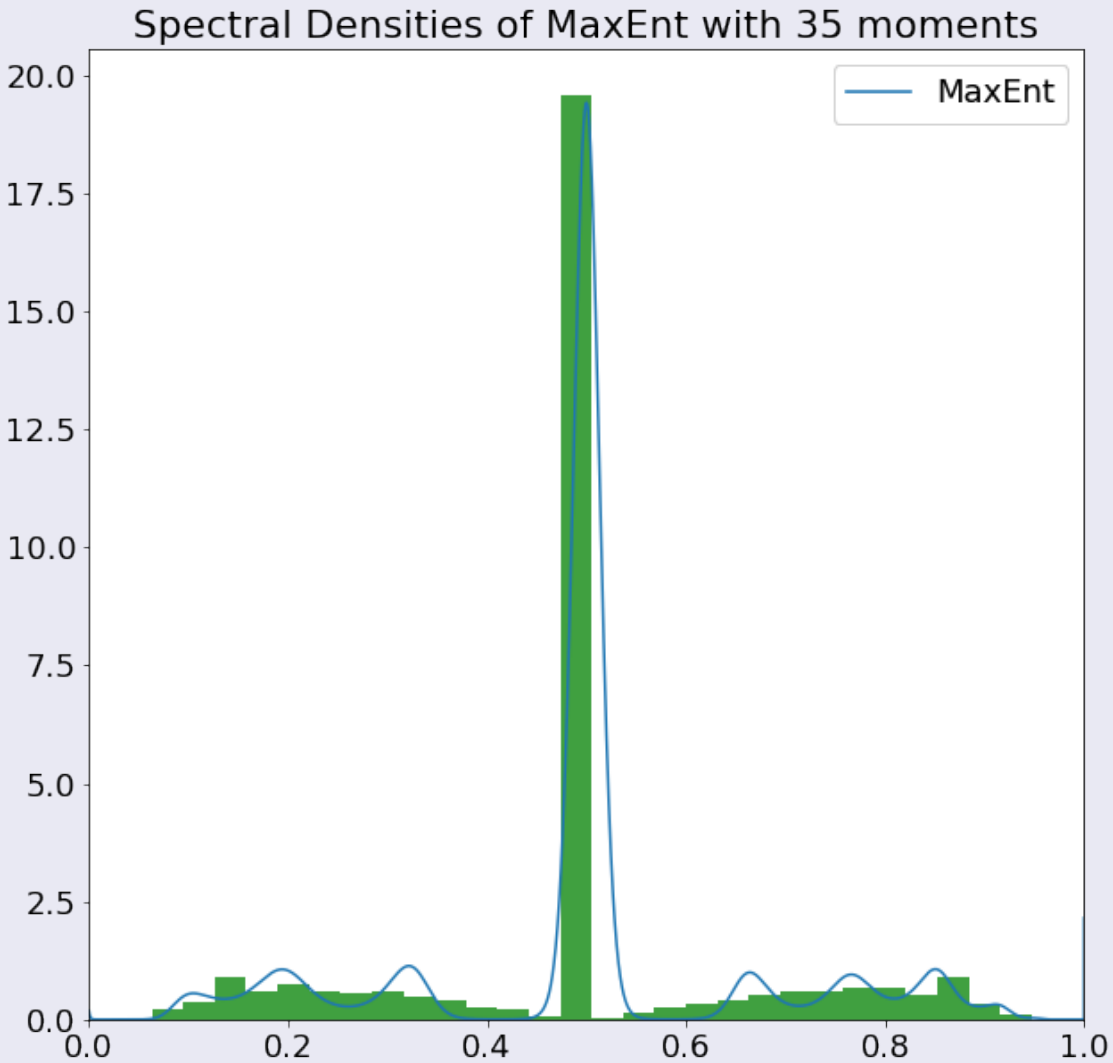
What about Graph Spectra?

Erdos Renyi - 100 moments



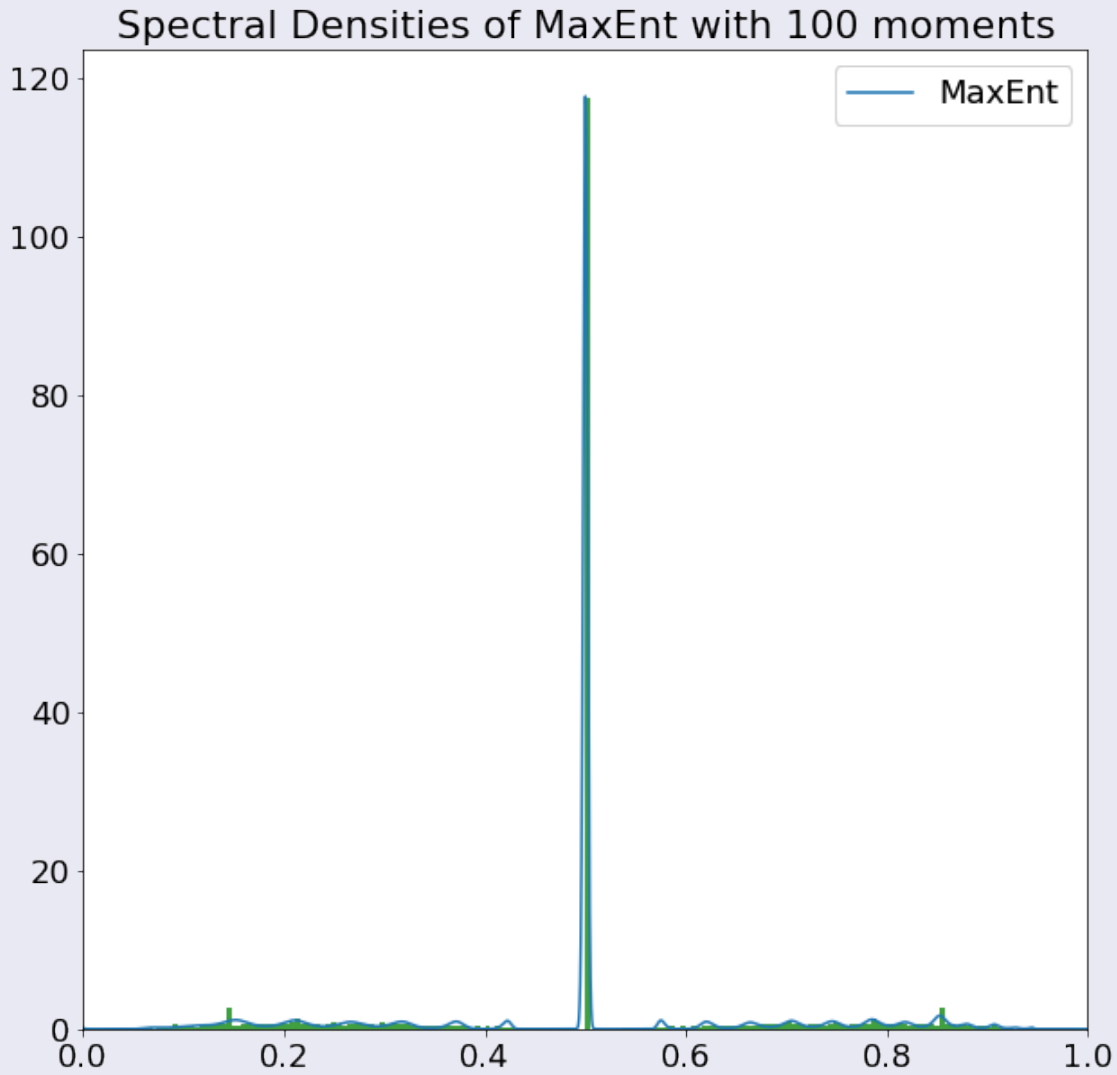
What about Graph Spectra?

Scale Free - 35 moments



What about Graph Spectra?

Scale Free - 100 moments



Entropic Spectral Learning

Clustering using the Eigenspectra

Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of the Laplacian $L \in \mathcal{R}^{n \times n}$ is equal to the number of connected components $A_1, ...A_k$ in the graph. The eigenspace of the eigenvalue 0 is spanned by the indicator vectors $1_{A_1}, ...1_{A_k}$.

Proof

$$u_i = \sum_{j=1} L_{ij} u_j \xrightarrow{u_j=1} \sum_{j=1} \left(1_{i=j} \sum_{k=1} w_{ik} - w_{ij} \right) = 0 \tag{29}$$

For k connected components, the matrix L has a block diagonal form

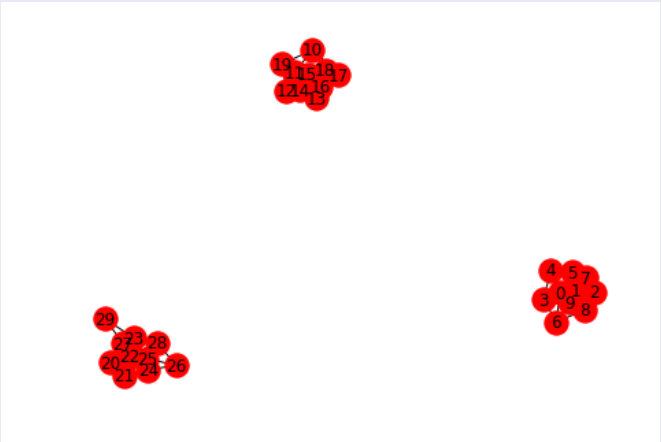
$$\begin{bmatrix} L_1 & \dots & \dots \\ \dots & \ddots & \dots \\ \dots & \dots & L_k \end{bmatrix}$$

Entropic Spectral Learning

Clustering using the Eigenspectra

Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of the Laplacian $L \in \mathcal{R}^{n \times n}$ is equal to the number of connected components $A_1, ...A_k$ in the graph. The eigenspace of the eigenvalue 0 is spanned by the indicator vectors $1_{A_1}, ...1_{A_k}$.

What does this mean



$$\lambda_1 = \lambda_2 = \lambda_3 = 0 < \lambda_n \tag{30}$$

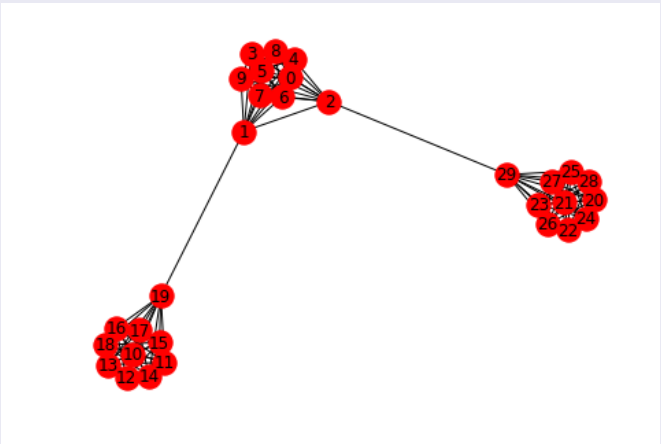
Entropic Spectral Learning

Extension using Matrix Perturbation Theory

$$\lambda_i' - \lambda_i = \lambda_i' = \mathbf{u}_i^T \delta L \mathbf{u}_i \leq ||\delta L||\,||\mathbf{u}_i^T \mathbf{u}_i|| = ||\delta L|| \tag{31}$$

- $||\delta L|| \ll ||L||$
- $L_{\text{natural}} = (D - A)/\sqrt{n}$
- bound goes as $R/\sqrt{n}$ by using the Frobenius norm

What does this mean



Experiments: Maximum Entropy vs Lanczos

Eigen-spectra for cluster detection

1. Compute L_{norm} of a network
2. Run Entropic Spectral Learning (m moments) and Lanczos Algorithm (m steps) to approximate the eigen-spectrum
3. Identify the eigen-gap and estimate number of clusters

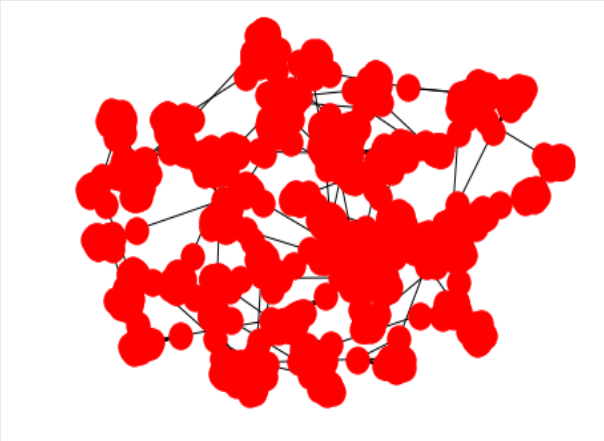
Symettrised KL divergence for network classification

$$\mathcal{D}_{kl}(p,q) = \frac{\mathcal{D}_{kl}(p\|q) + \mathcal{D}_{kl}(q\|p)}{2} \tag{32}$$

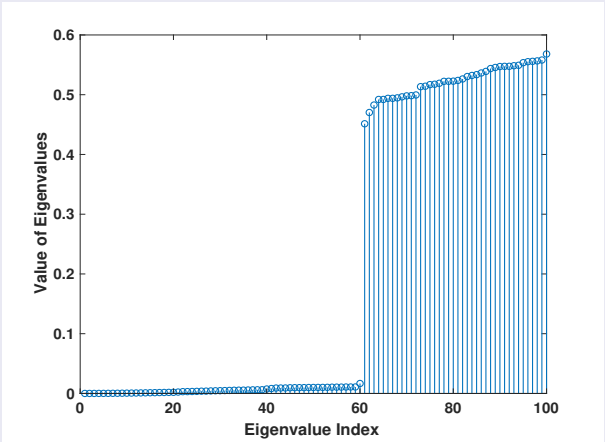
Minimize $\mathcal{D}_{kl}(p,q)$ between eigen-spectra of two networks to recover parameters or network type

Experiments: Eigen-spectra for Cluster Detection

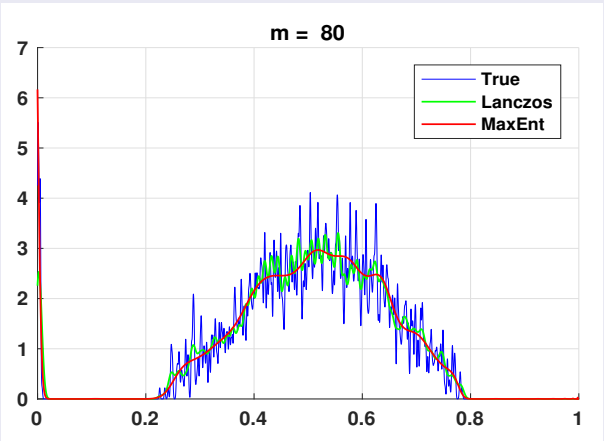
Synthetic networks



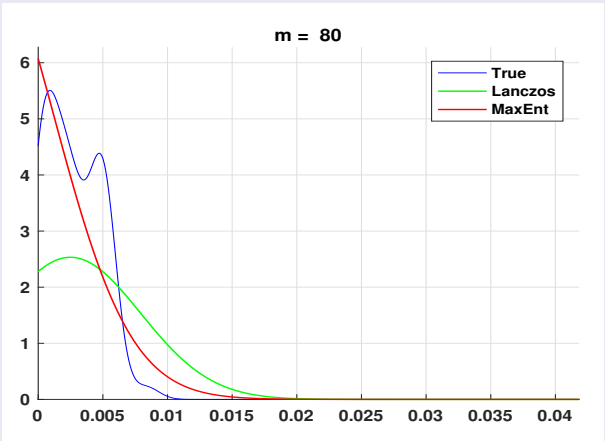
(a) A 60-cluster network



(b) Stem graph of eigenvalues



(c) Spectral densities



(d) Enlarged spectral densities

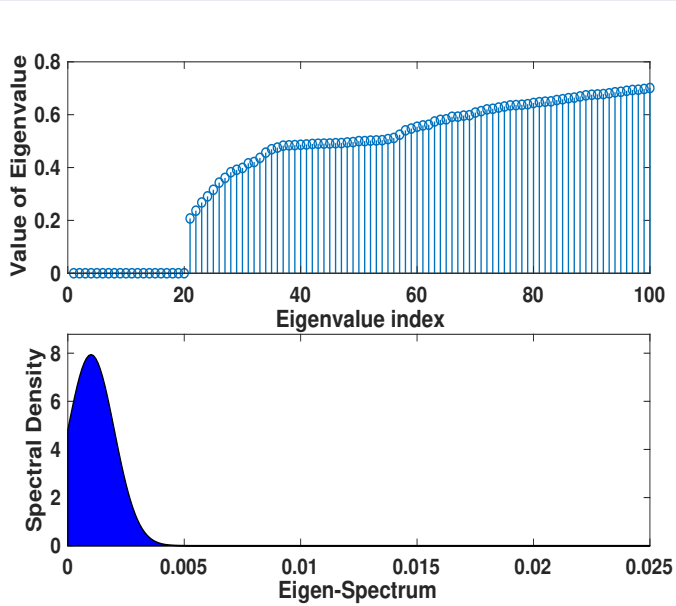
Experiments: Eigen-spectra for Cluster Detection

Synthetic networks

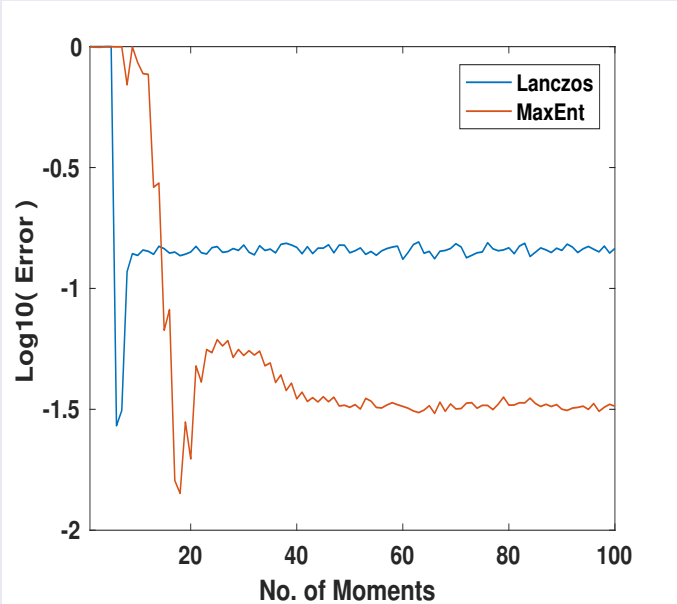
Experiments: Eigen-spectra for Cluster Detection

Small real-world networks

Email network: email communication data among $1,005$ members of a large European research institution (Leskovec et al., 2007b)



(a) The smallest 100 eigenvalues



(b) Error of cluster detection

Experiments: Eigen-spectra for Cluster Detection

Large real-world networks

YouTube network: $n = 1,134,890$ (Leskovec et al., 2007b)

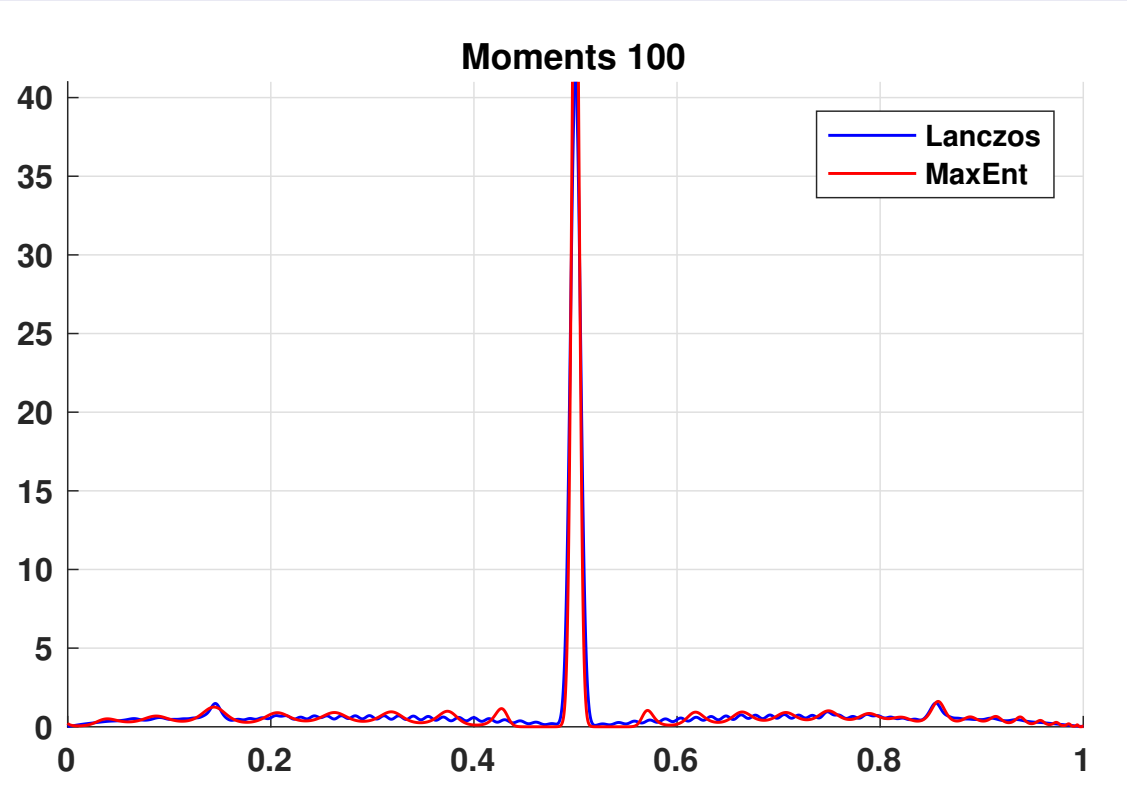


Figure: Spectral densities of YouTube network by MaxEnt and Lanczos

Experiments: Eigen-spectra for Cluster Detection

Large real-world networks

Table: Cluster prediction by MaxEnt for DBLP ($n = 317,080$), Amazon ($n = 334,863$) and YouTube ($n = 1,134,890$).

MOMENTS	40	70	100
DBLP	2.215×10^4	8.468×10^3	8.313×10^3
AMAZON	2.351×10^4	1.146×10^4	1.201×10^4
YOUTUBE	4.023×10^3	1.306×10^4	1.900×10^4

Experiments: Sym KL divergence

Learn parameter of synthetic networks

Table: Average parameters estimated by MaxEnt for the 3 types of network

Number of nodes n	50	100	150
Erdös-Rényi ($p = 0.6$)	0.600	0.598	0.6040
Watts-Strogatz ($p = 0.4$)	0.4683	0.4537	0.4129
Barabási-Albert ($r = 0.4n$)	19	40	58

Experiments: Sym-KL divergence

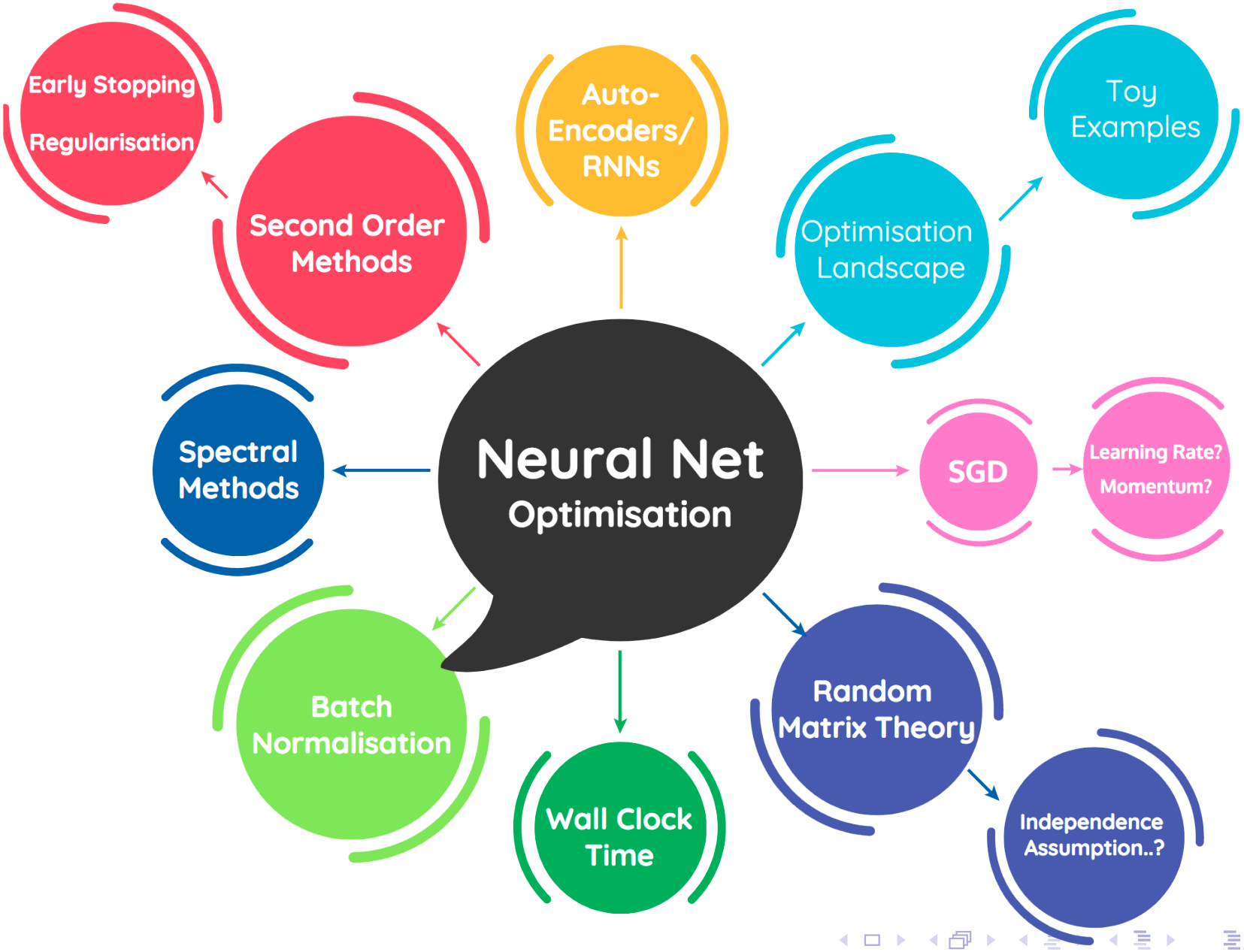
Classify networks

Table: Symmetric KL divergence between entropic spectrum of Barabási-Albert, YouTube and those of synthetic networks

	Barabási-Albert ($n = 5,000$)	Youtube ($n = 1,134,890$)
Erdös-Rényi	2.662	7.728
Watts-Strogatz	7.6123	9.735
Barabási-Albert	**2.001**	**7.593**

Defeating the Deep Learning Dragon

Opening the Black Box of Deep Learning Optimisation



Neural Network loss Surfance analysis

Previous Work

- Toy examples limited to 2-hidden layers 5k parameters
  - storing the Hessian has memory cost $\mathcal{O}(n^2)$ and eigendecomposition cost $\mathcal{O}(n^3)$
- Theoretical spin glass analysis relies on unrealistic assumptions
- **Our Contribution:** Extend analysis to million parameter networks

Answer = Lanczos-Stieltjes

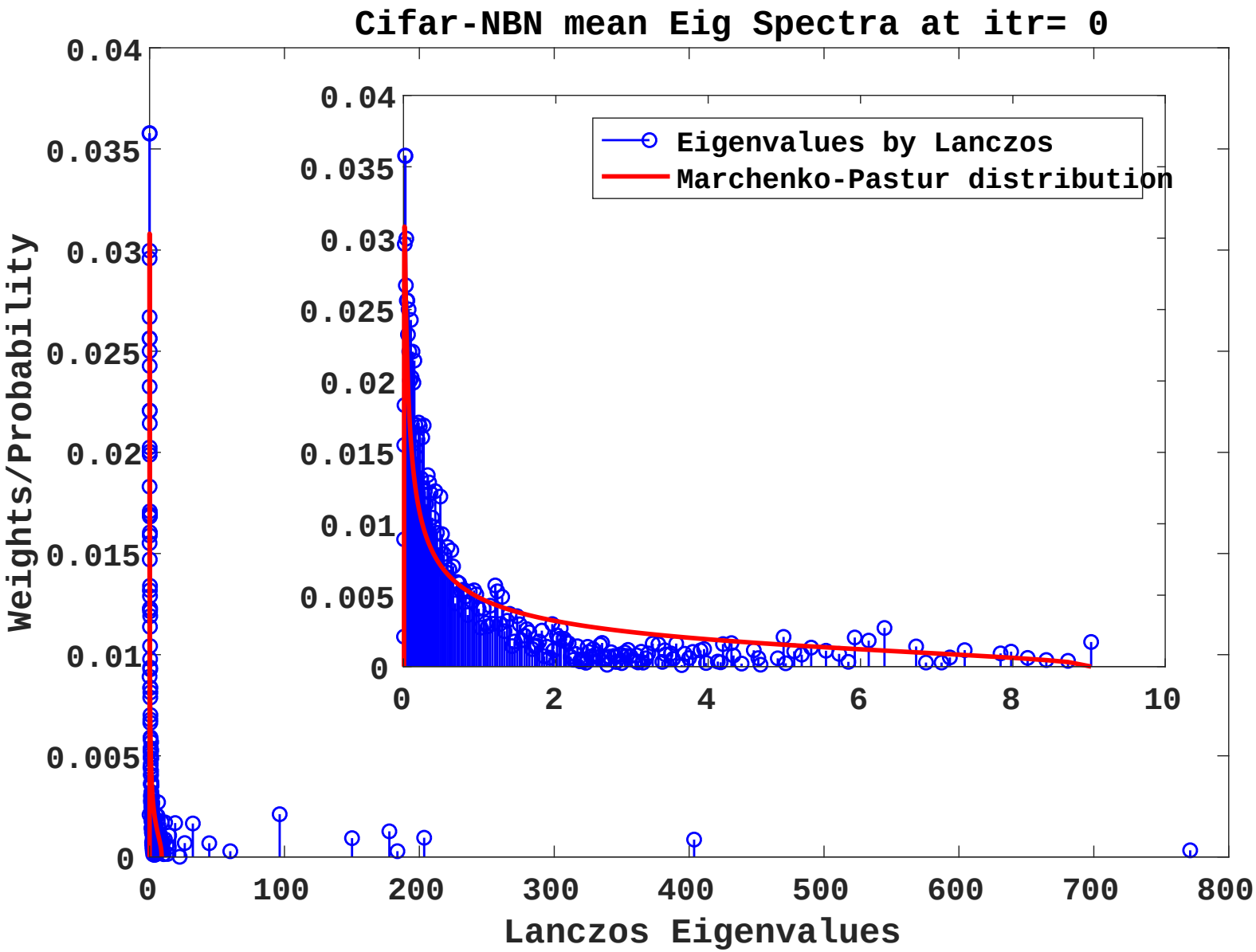
- use Pearlmutter trick (cost equivalent to 2 gradient evaluations) to calculate a Hessian vector product in $\mathcal{O}(n)$
- use Hessian vector product as input to the Lanczos algorithm and obtain tridiagonal matrix $\mathbb{T}^{m \times m}$

Neural Network loss Surfance analysis

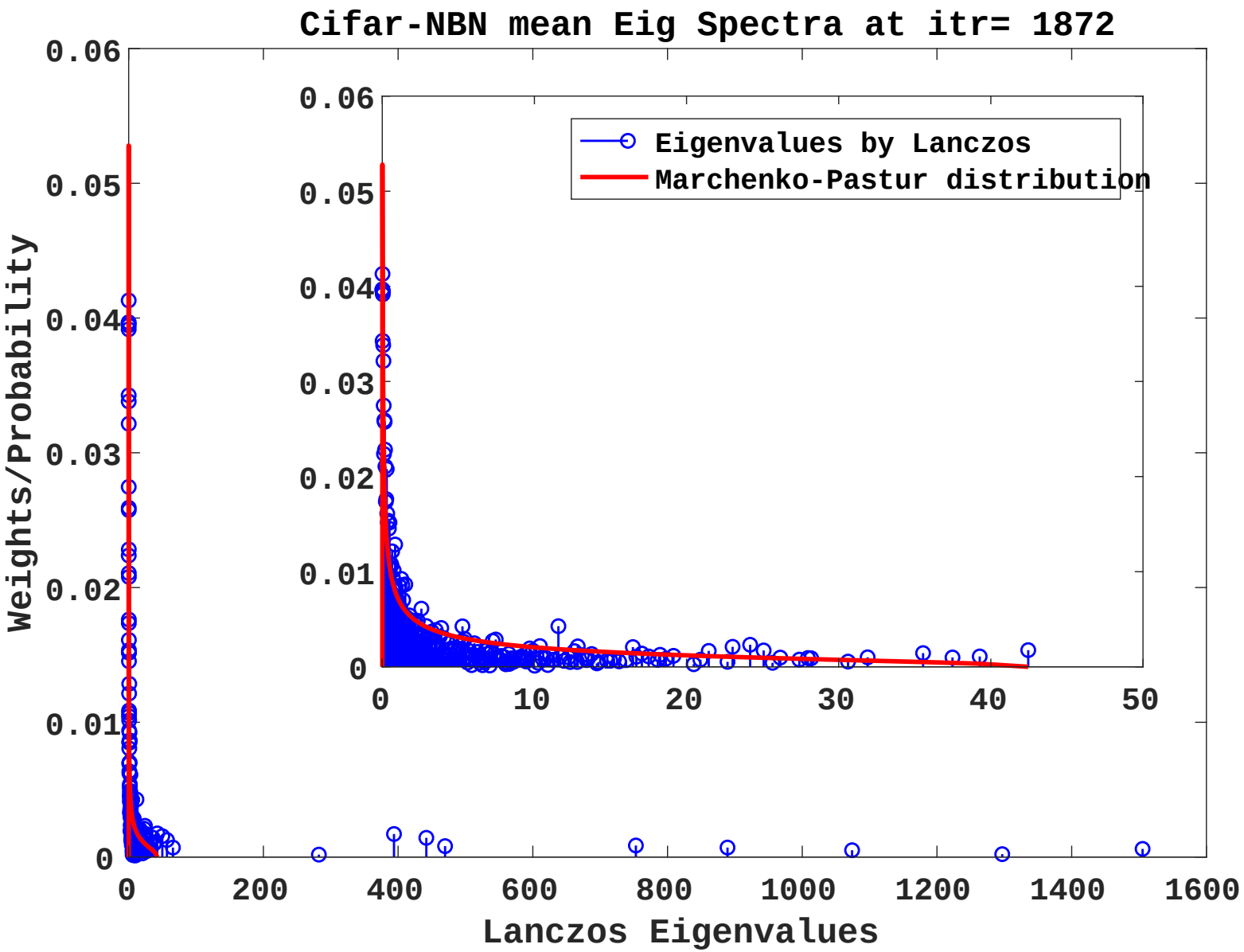
Lanczos-Stieltjes continued

- $(\mathbb{M})_{ij} = v^T \mathbb{H}^{i+j-2} v$
- **insight:** for zero mean unit variance random vectors
- $\mathbb{E}_v(v^T \mathbb{H}^m v) = \sum_{i=1}^{n} \lambda_i^m \approx \frac{1}{n_v} \sum_{j=1}^{n_v} v_j^T \mathbb{H} v_j$
- $p(\lambda) = \frac{1}{n_v} \sum_{l=1}^{n_v} \left(\sum_{k=1}^{m} (\tau_k^{(l)})^2 \delta(\lambda - \lambda_k^{(l)}) \right)$
- **as if by magic:** the Lanczos algorithm constructs a discrete measure that gives us an estimate of the spectral density
- Now lets look as some spectral plots

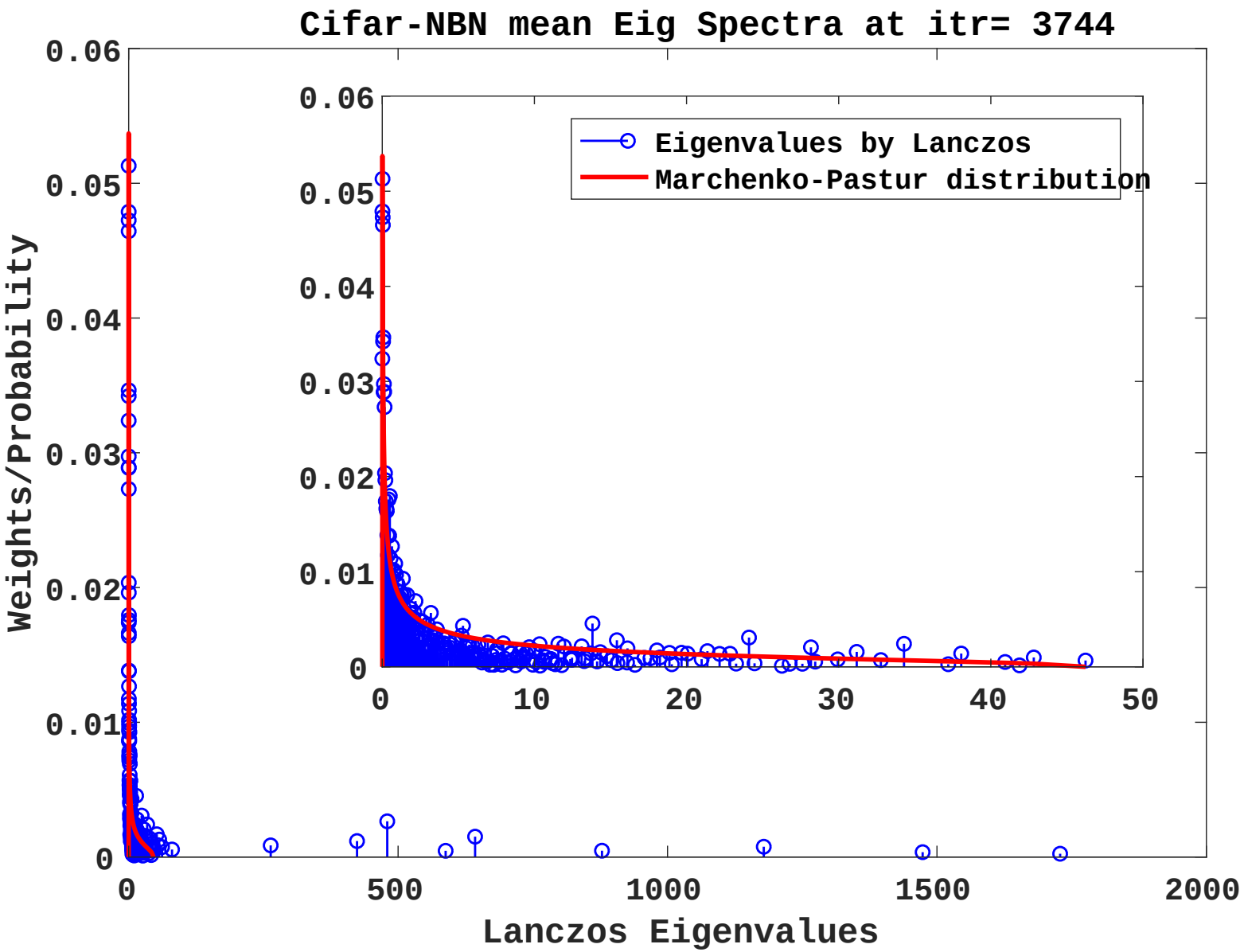
Cifar-10 with no Batch Norm



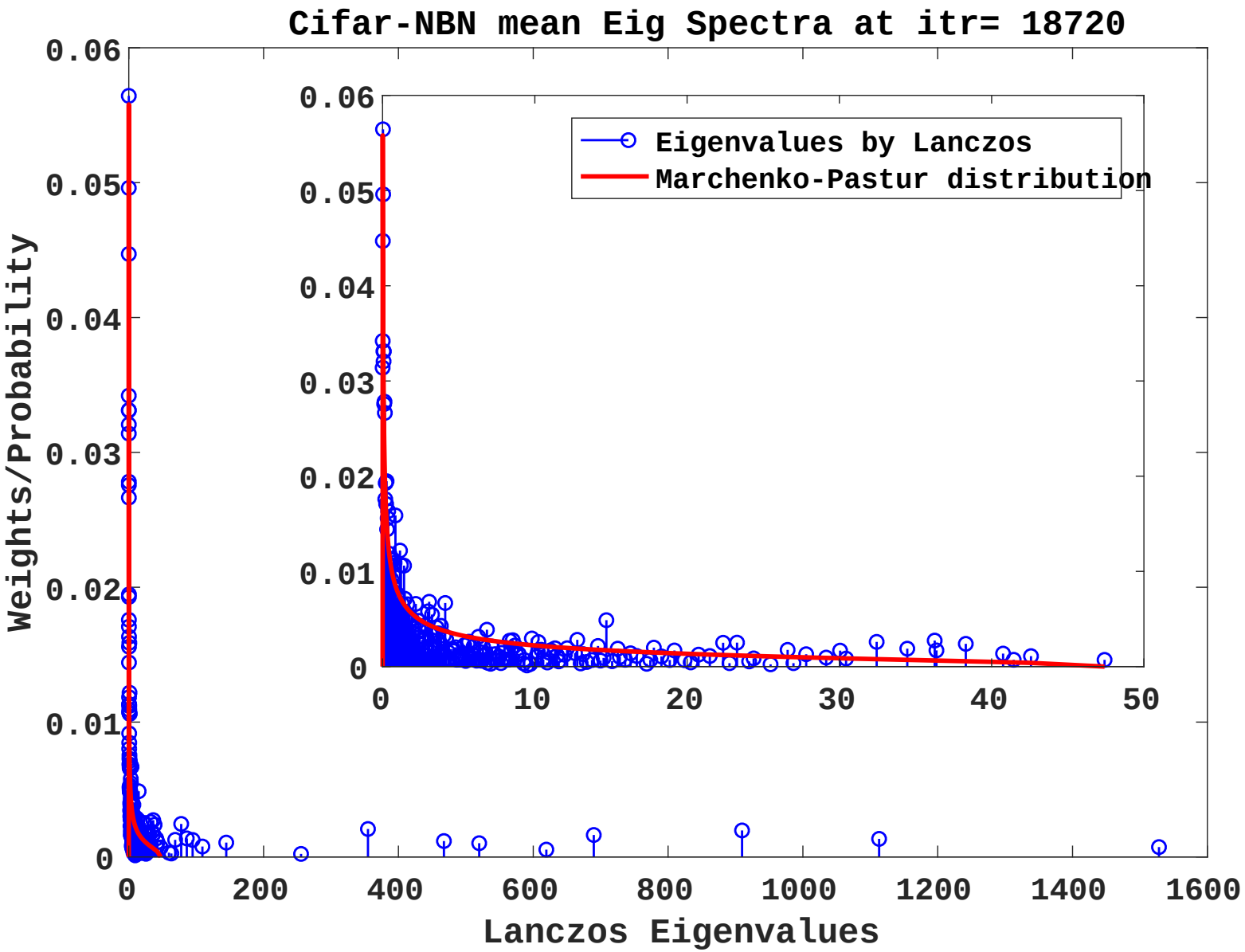
Cifar-10 with no Batch Norm



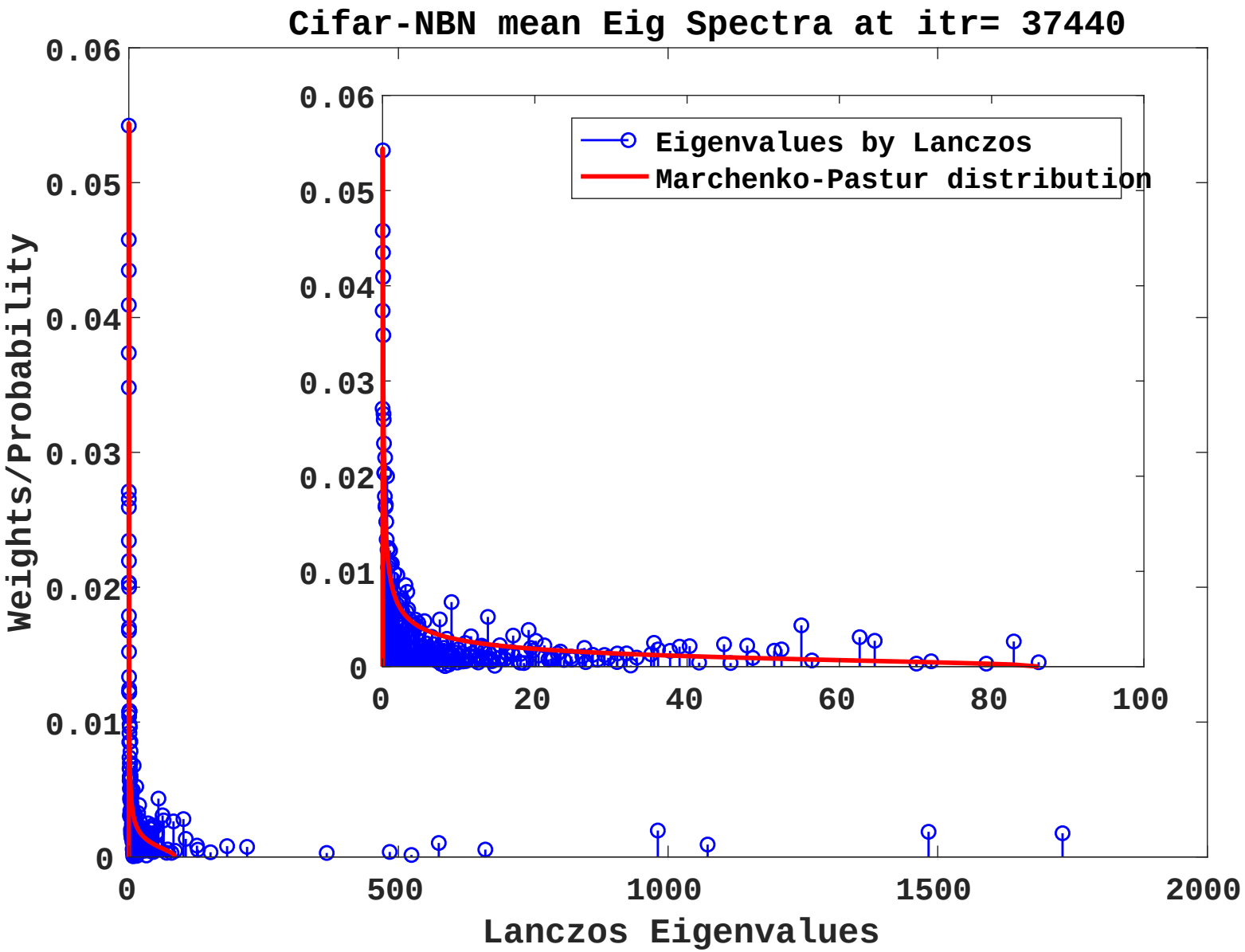
Cifar-10 with no Batch Norm



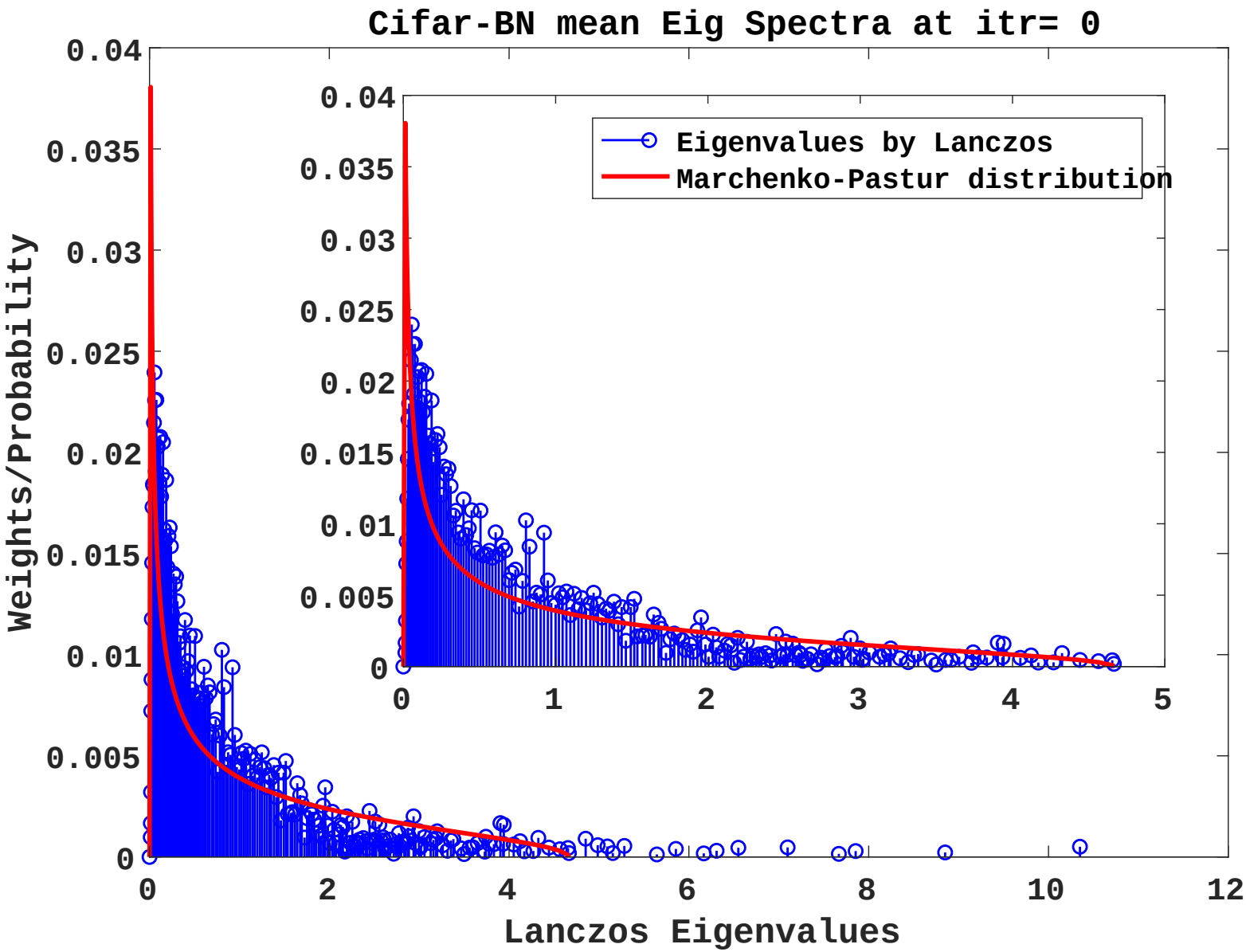
Cifar-10 with no Batch Norm



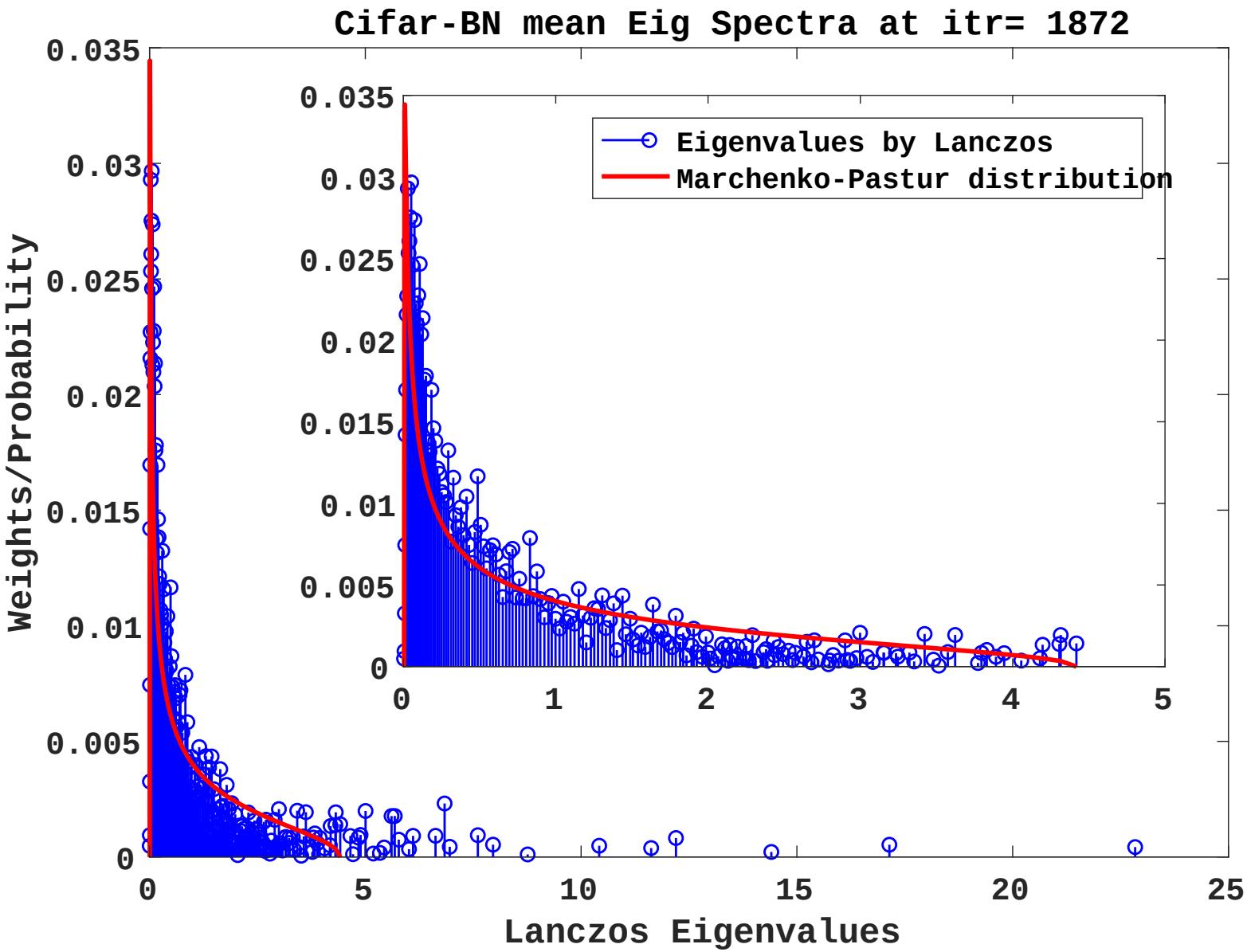
Cifar-10 with no Batch Norm



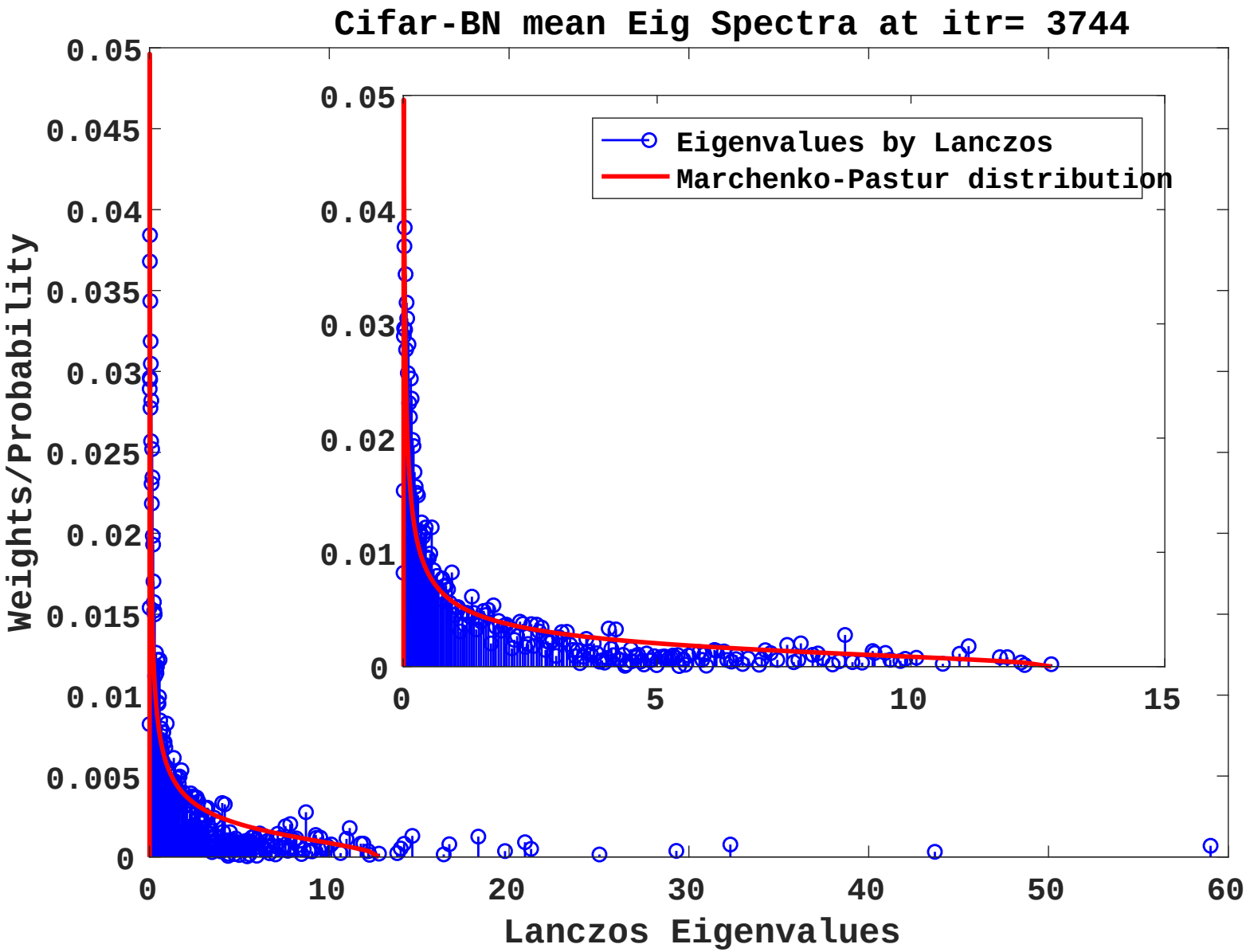
Cifar-10 with Batch Norm



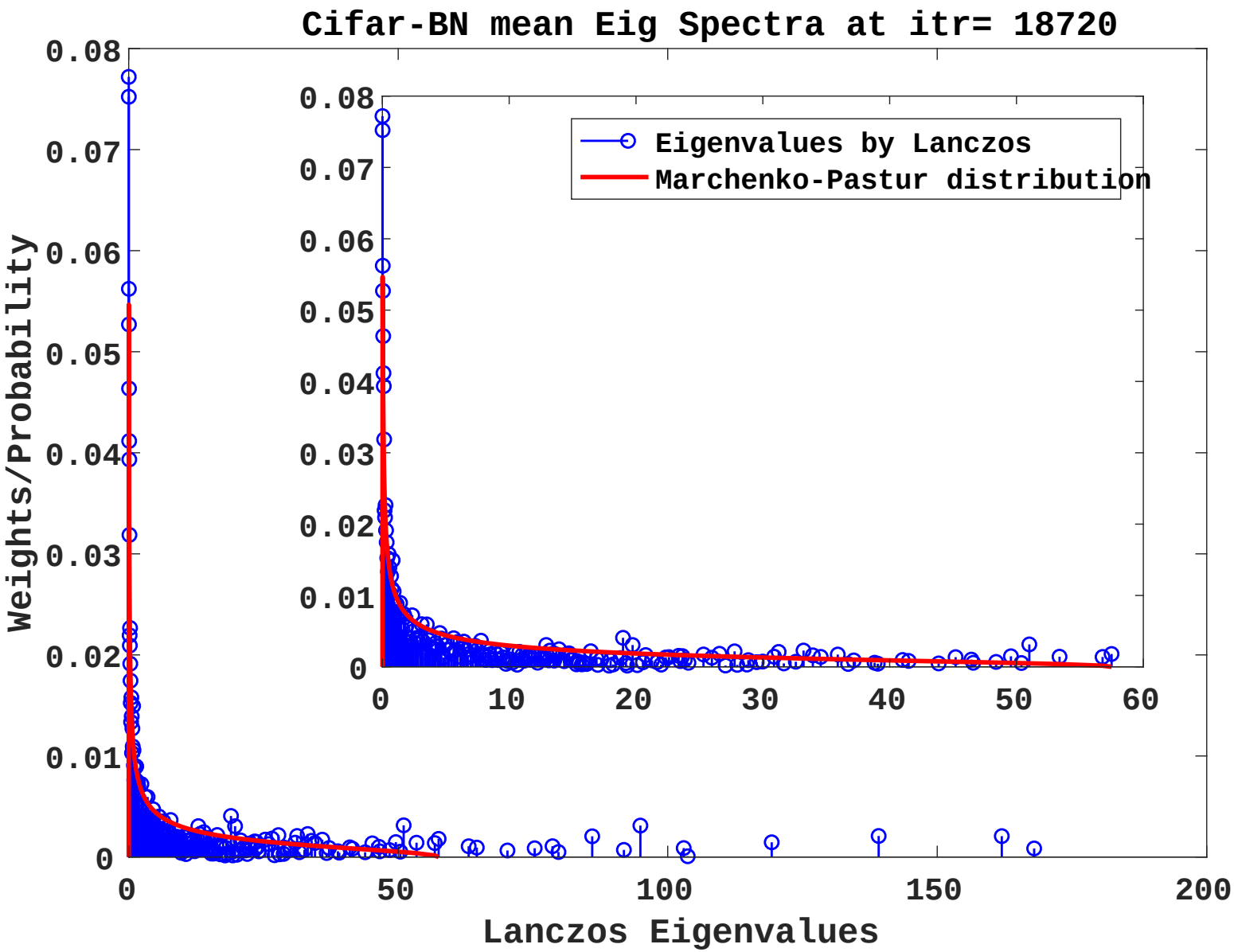
Cifar-10 with Batch Norm



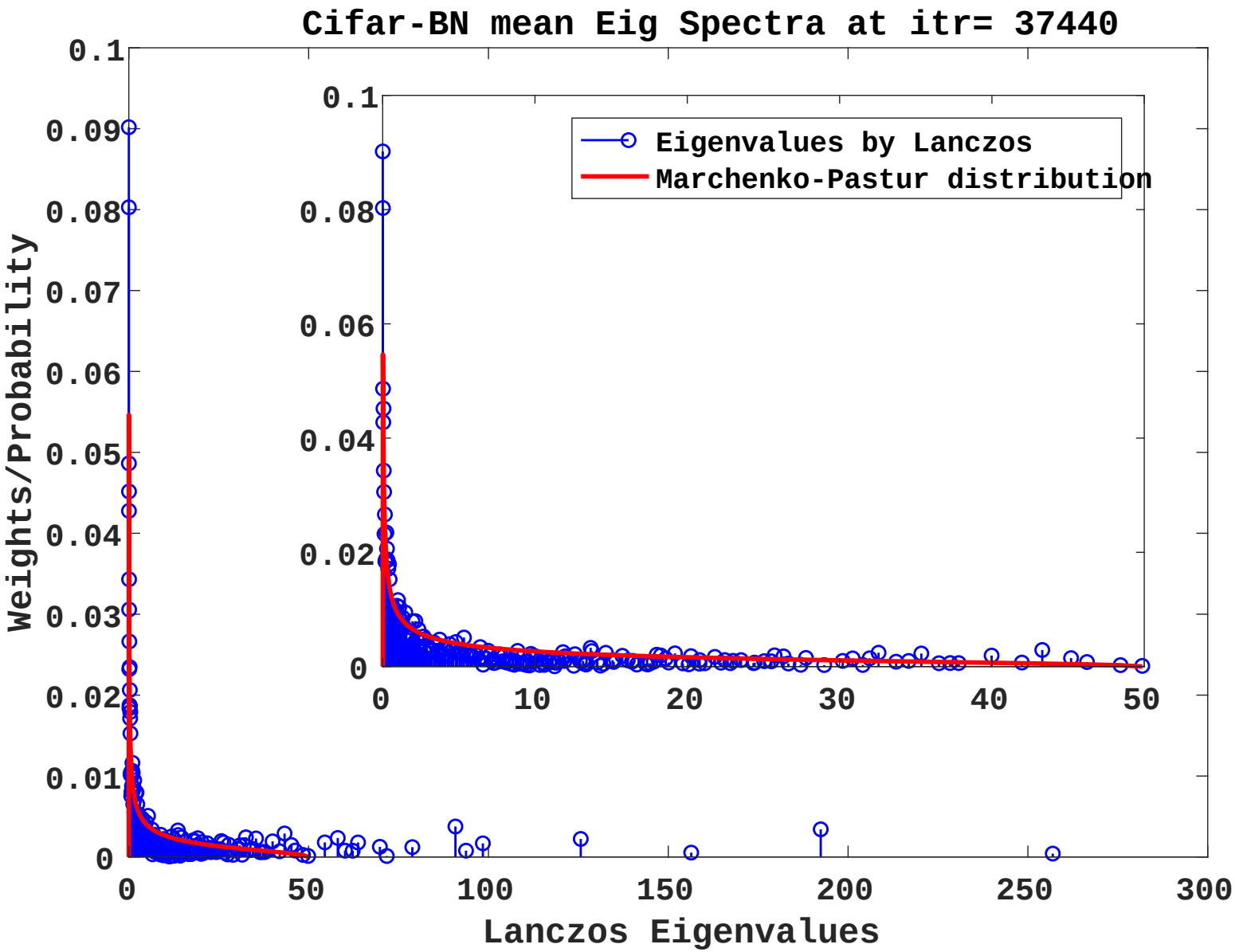
Cifar-10 with Batch Norm



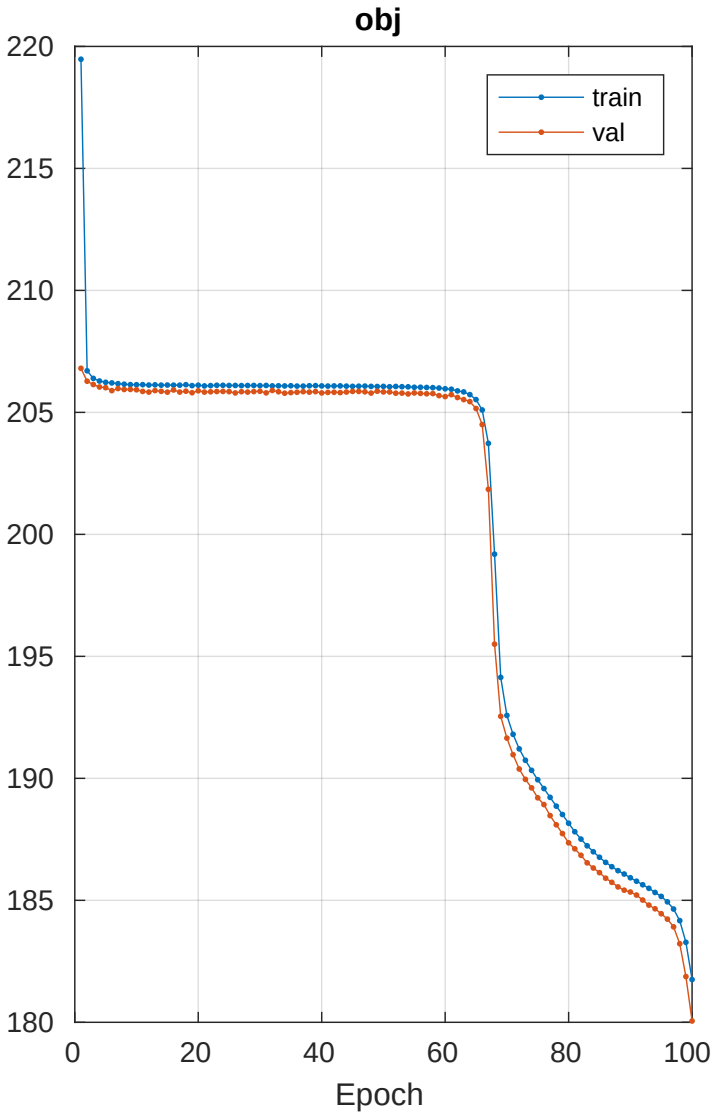
Cifar-10 with Batch Norm

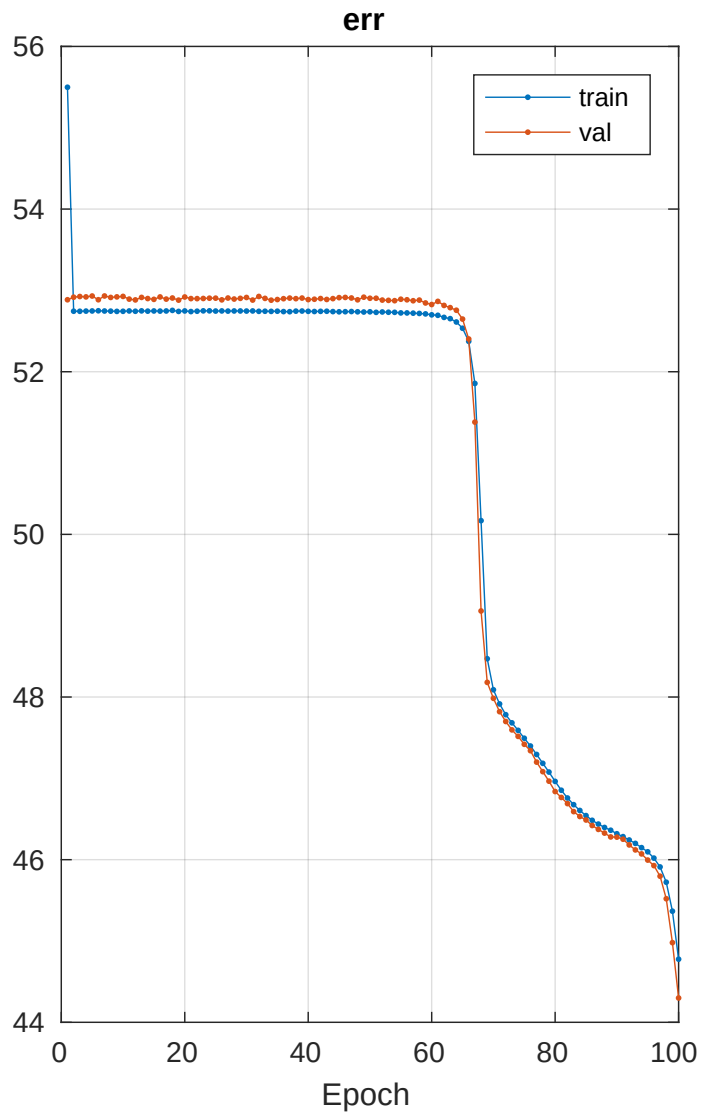


Cifar-10 with Batch Norm

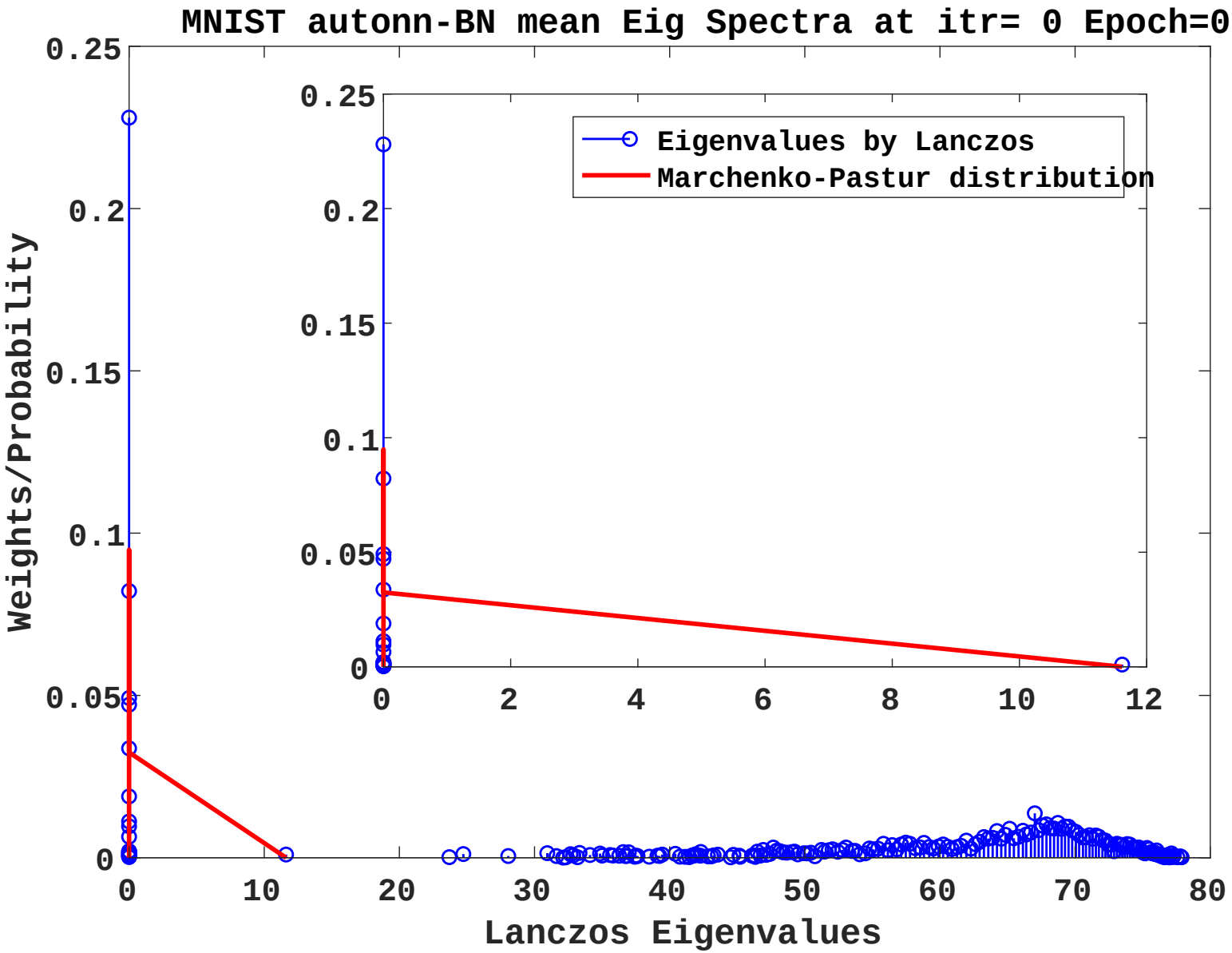


MNIST autoencoder with SGD performance

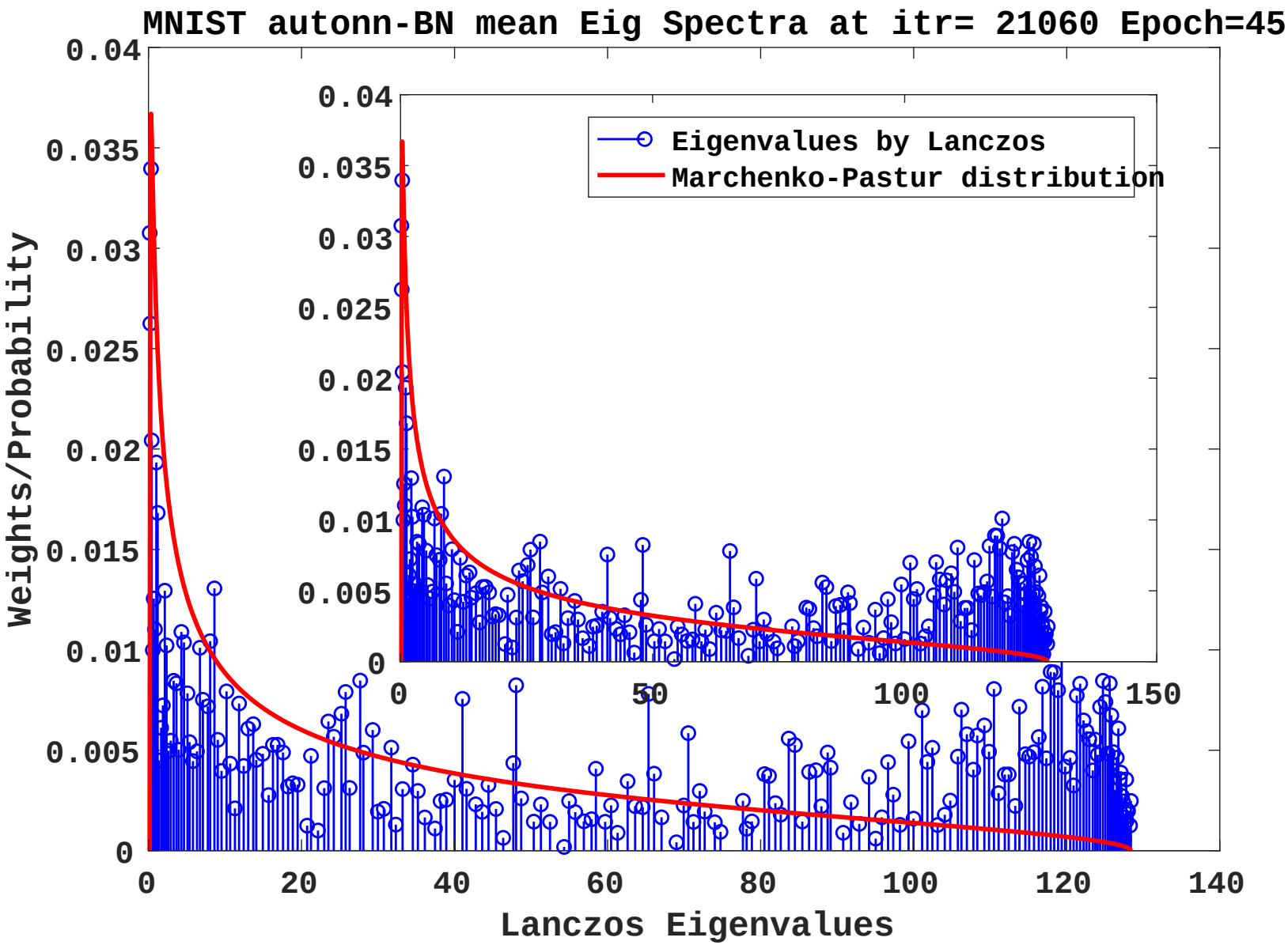




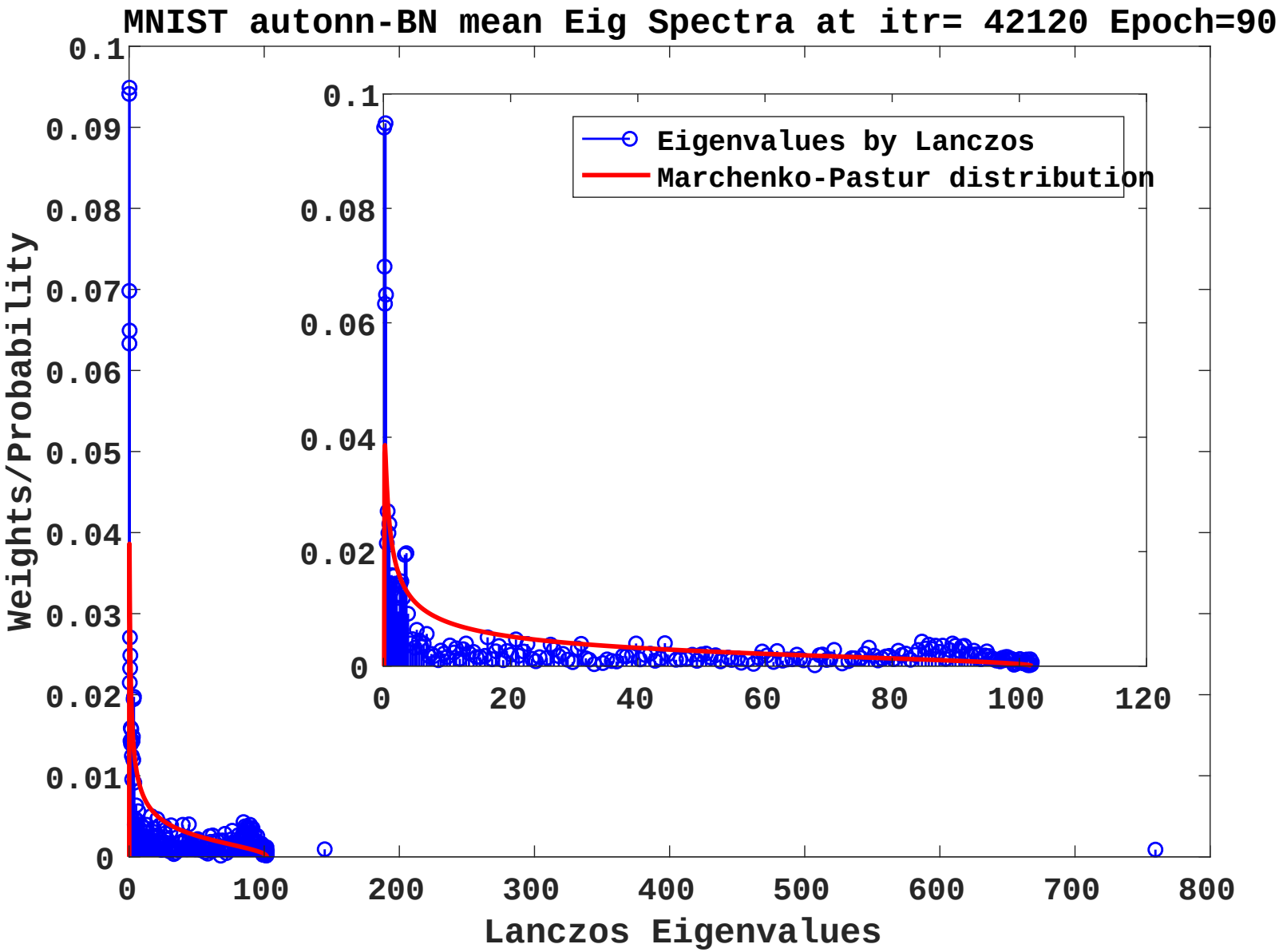
MNIST autoencoder eigenspectrum



MNIST autoencoder eigenspectrum



MNIST autoencoder eigenspectrum



Convergence

Analysis of SGD on quadratic

- If objective is convex solving quadratic always reduces objective
- $\alpha = \frac{2}{\lambda_1 + \lambda_n}$
- convergence $\frac{\lambda_1/\lambda_n - 1}{\lambda_1/\lambda_n + 1}$

Second Order Methods

Introduction

- For a generic loss L, weights p and perturbation d

$$
\begin{aligned}
d^* &= \text{argmin}_d L(p+d) \\
L(p+d) &= L(p) + \nabla L^T d + \frac{1}{2} d^T \mathbb{H} d \\
d &= -\mathbb{H}^{-1} \nabla L(p) = -\sum_i^N \frac{1}{\lambda_i} u_i u_i^T \nabla L(p)
\end{aligned}
\tag{34}
$$

- Use PSD approximation to $\mathbb{H}^{n \times n}$
  - Gauss-Newton $G + \beta I$, or Fisher, or Absolute Hessian or just add $\mathbb{H} + \beta I$
- Use Conjugate Gradient or Lanczos

Second Order Methods

Unpacking the secrets

- What is the effect of heuristic early stopping criteria m?
- Why do we need to add the identity to the Hessian or Gauss-Newton?

Random Matrix Theory

- Can you estimate the eigenvalues and eigenvectors of $\mathbb{H}^{n \times n}$ with $T << N$ samples.
  - Spiked Covariance Models
  - If the eigenvalue/eigenvector pair resides far from the bulk then yes
  - in the $n \xrightarrow{lim} \infty$ limit empirical covariance matrix is not a consistent estimator of the true

Second Order Methods

Adding the Identity

$$\bar{\Xi} = \text{argmin}_{\Xi} ||H^{-1}J - \Xi(M)^{-1}J|| \leq \left(\text{argmin}||H^{-1} - \Xi(M)^{-1}||\right)||J|| \tag{35}$$

$$\hat{\xi}_i = \sum_{j=1}^{N} \langle u_i | u_j \rangle^2 \lambda_j \tag{36}$$

Linear Shrinkage Estimator

$$\bar{H} = \gamma \bar{H} + (1-\gamma) I \tag{37}$$

$$\bar{H} = \bar{H} + \beta I \tag{38}$$

$$\frac{\rho}{\lambda_i + \beta} = \frac{1}{\gamma(\lambda_i - 1) + 1} \therefore \rho = \frac{1}{\beta + 1} \tag{39}$$

LORG

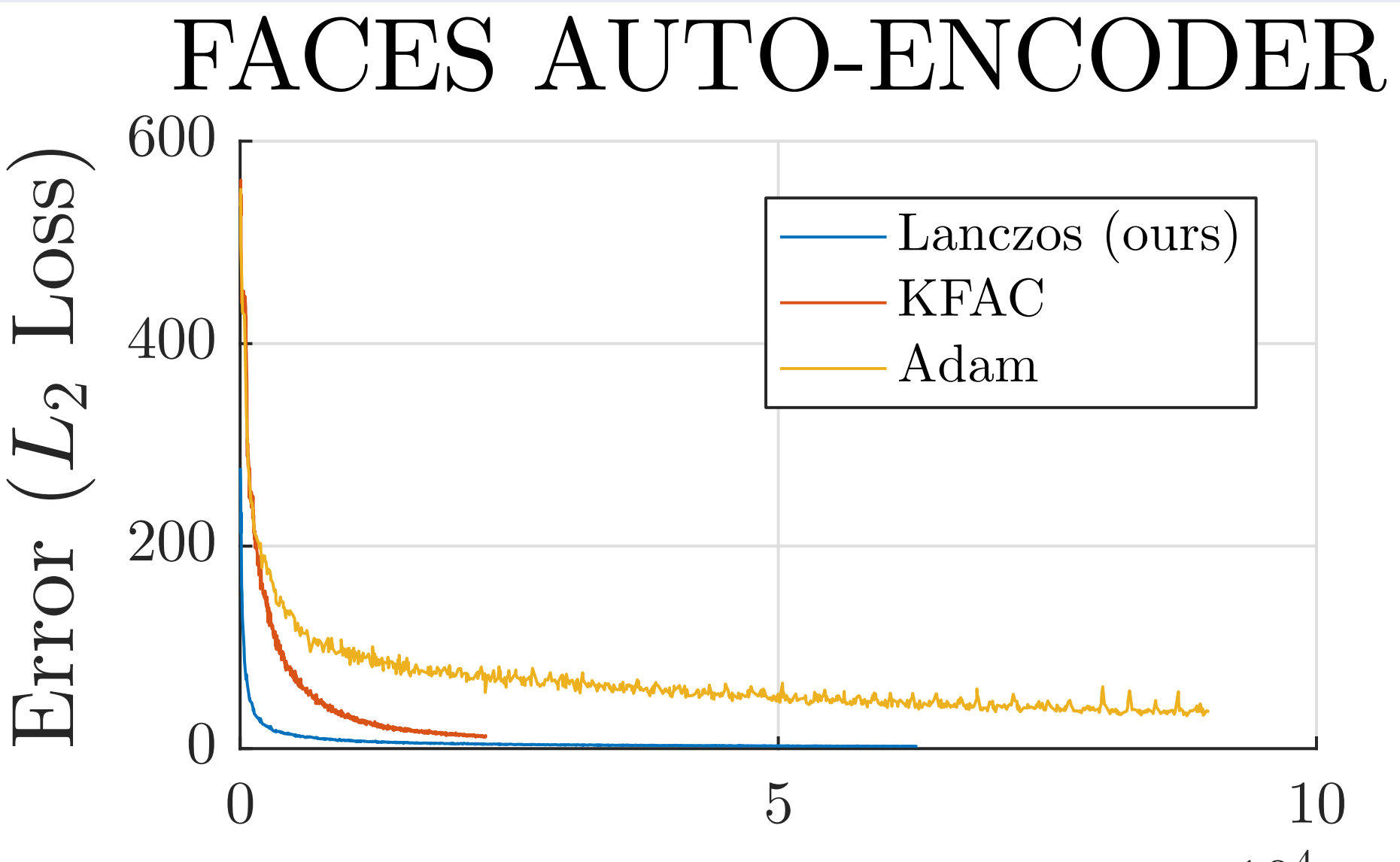
Lanczos Outlier learner

$$d = -\sum_{i}^{m} \frac{1}{\lambda_i + \beta} u_i u_i^T \nabla L \tag{40}$$

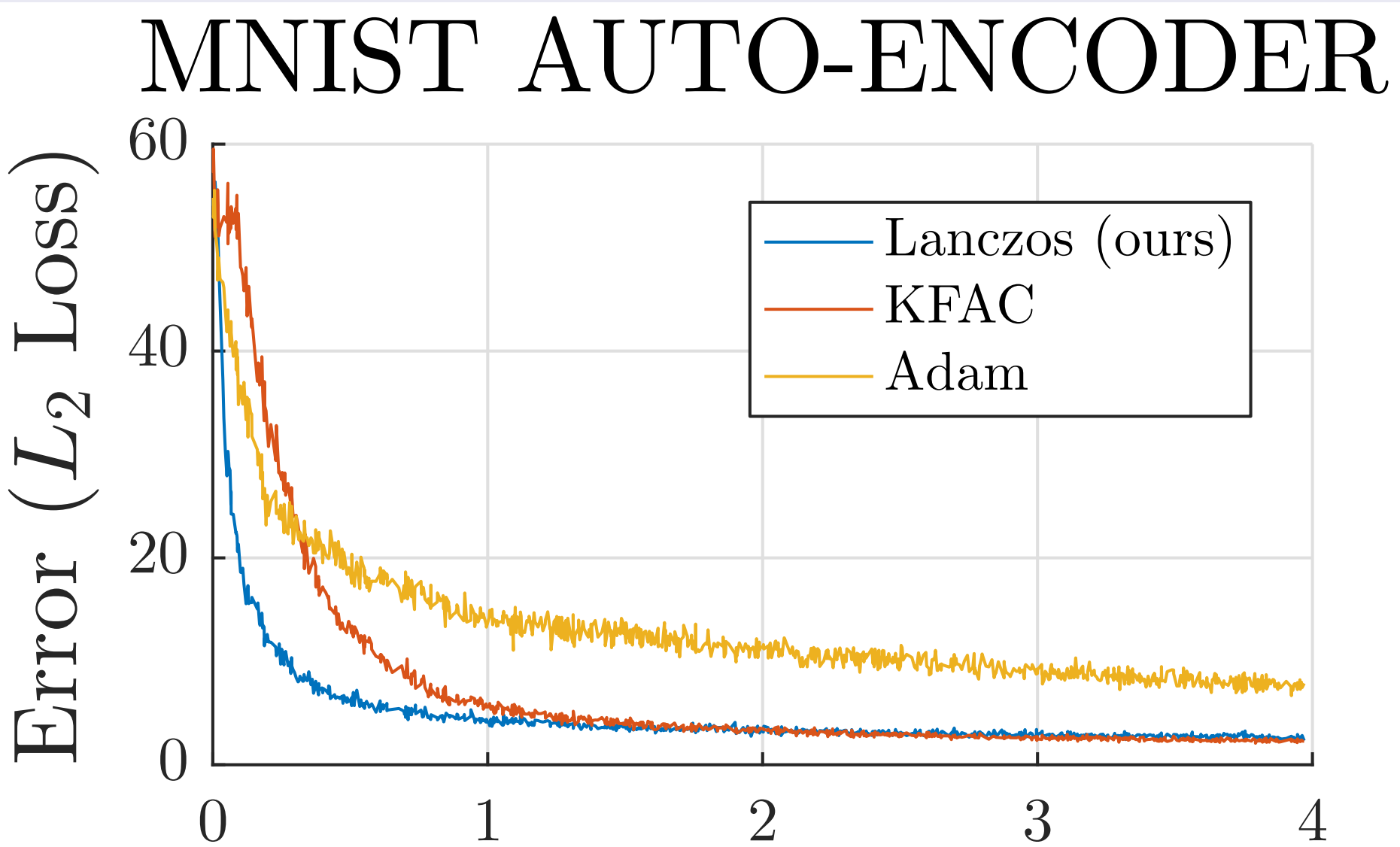
Lanczos Outlier Reinjected Gradient

$$\begin{aligned} &-\sum_{i=0}^{m} \left(\frac{u_i^T \nabla L}{\lambda_i + \beta} \right) u_i - \frac{1}{\beta} \sum_{m+1}^{n} u_i u_i^T \nabla L \\ &= -\sum_{i}^{m} \left(\frac{1}{\lambda_i + \beta} - \frac{1}{\beta} \right) u_i u_i^T \nabla L - \frac{1}{\beta} \nabla L \end{aligned} \tag{41}$$

Lanczos Outlier Reinjected Gradient



Lanczos Outlier Reinjected Gradient



Leaping Stochastic Gradient Descent

Why Momentum Really works?

Consider applying SGD on the convex quadratic loss $L(w) = \frac{1}{2} w^T \bar{H} w + b^T w$, as $\nabla L(w) = \bar{H} w + b$, the updates can be written as

$$w^{k+1} = w^k - \alpha(\bar{H} w + b) \tag{42}$$

Changing the co-ordinate frame into the eigenbasis of $\bar{H}$, i.e $x_i^k = Q^T(w^k - w^*)$, $Q = [q_1, .., q_n]$

$$\begin{aligned} w^k - w^* &= \sum_{i=1}^{n} x_i^0 (1 - \alpha\lambda_i)^k q_i \\ L(w^k) - L(w^*) &= \sum_{i}^{n} x_i^0 (1 - \alpha\lambda_i)^{2k} \lambda_i (x_i^0)^2 \end{aligned} \tag{43}$$

Leaping Stochastic Gradient Descent

Why Momentum Really works?

For convergence, all step sizes must satisfy $0 < \alpha\lambda_i < 2$ and by symmetry the optimal rate is attained when both the largest eigenvalue, λ_1, and the smallest eigenvalue, λ_n, converge at the same rate, i.e $\alpha = 2/(\lambda_1 + \lambda_n)$, with an optimal rate of

$$\frac{\lambda_1/\lambda_n - 1}{\lambda_1/\lambda_n + 1}. \tag{44}$$

Leaping Stochastic Gradient Descent

For Momentum

$$\begin{aligned} z^{k+1} &= \beta z^k + \nabla f(w^k) \\ w^{k+1} &= w^k - \alpha z^{k+1} \end{aligned} \tag{45}$$

$$R^k \begin{bmatrix} y_i^0 \\ x_i^0 \end{bmatrix} = \begin{bmatrix} y_i^k \\ x_i^k \end{bmatrix} \tag{46}$$

$$R = \begin{bmatrix} \beta & \lambda_i \\ \alpha\beta & 1 - \alpha\lambda_i \end{bmatrix} \tag{47}$$

$$\alpha = \left(\frac{2}{\sqrt{\lambda_1} + \sqrt{\lambda_n}} \right)^2, \beta = \left(\frac{\sqrt{\lambda_1} - \sqrt{\lambda_n}}{\sqrt{\lambda_1} + \sqrt{\lambda_n}} \right)^2 \tag{48}$$

Leaping Stochastic Gradient Descent

For Momentum - with Hessian Broadening

$$\begin{aligned} z^{k+1} &= \beta z^k + \nabla f(w^k) \\ w^{k+1} &= w^k - \alpha z^{k+1} \end{aligned} \tag{49}$$

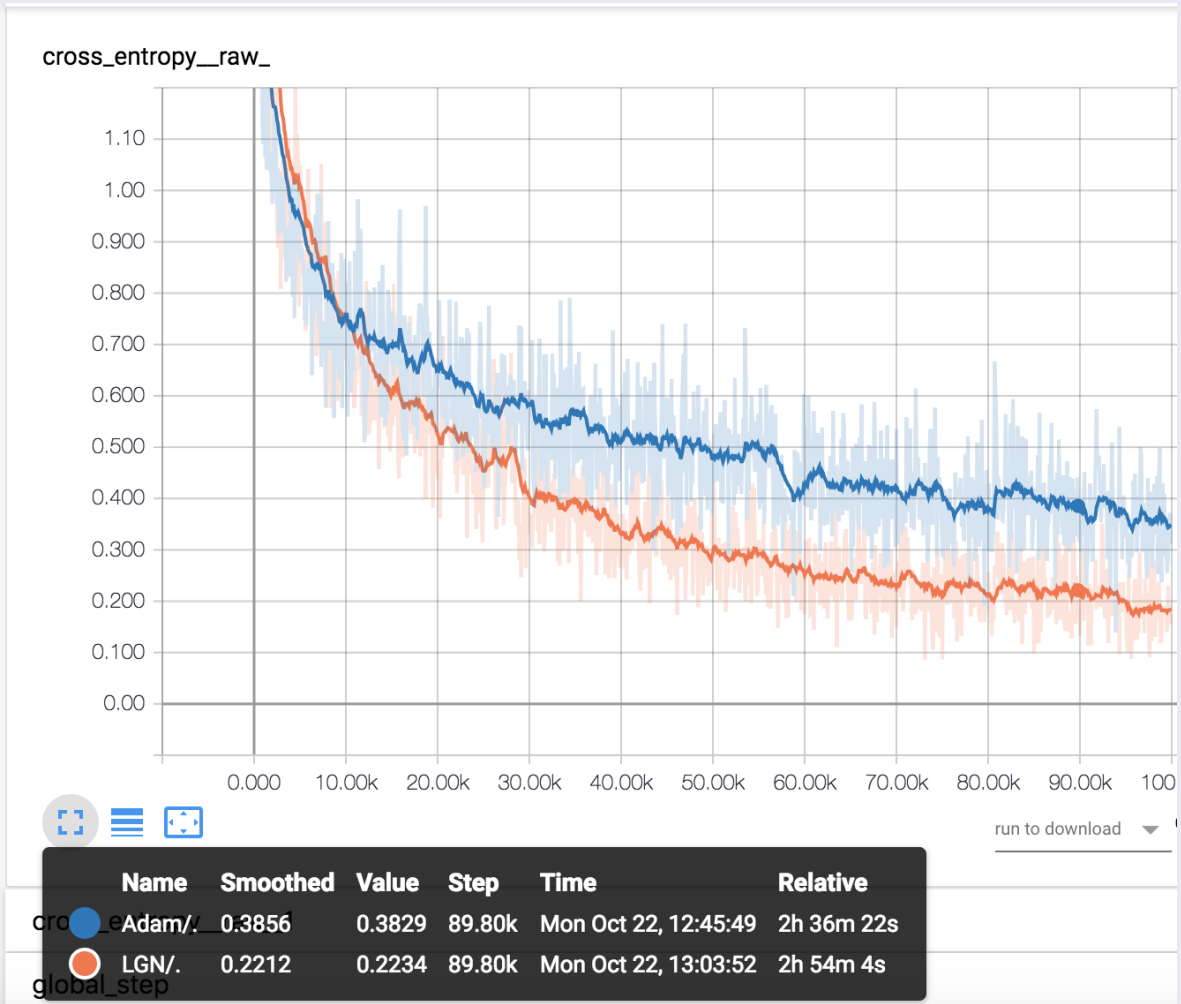
$$R^k \begin{bmatrix} y_i^0 \\ x_i^0 \end{bmatrix} = \begin{bmatrix} y_i^k \\ x_i^k \end{bmatrix} \tag{50}$$

$$R = \begin{bmatrix} \beta & \lambda_i \\ \alpha\beta & 1 - \alpha\lambda_i \end{bmatrix} \tag{51}$$

$$\alpha = \left(\frac{2}{\sqrt{\lambda_1} + \sqrt{\lambda_{bulk}}} \right)^2, \beta = \left(\frac{\sqrt{\lambda_1} - \sqrt{\lambda_{bulk}}}{\sqrt{\lambda_1} + \sqrt{\lambda_{bulk}}} \right)^2 \tag{52}$$

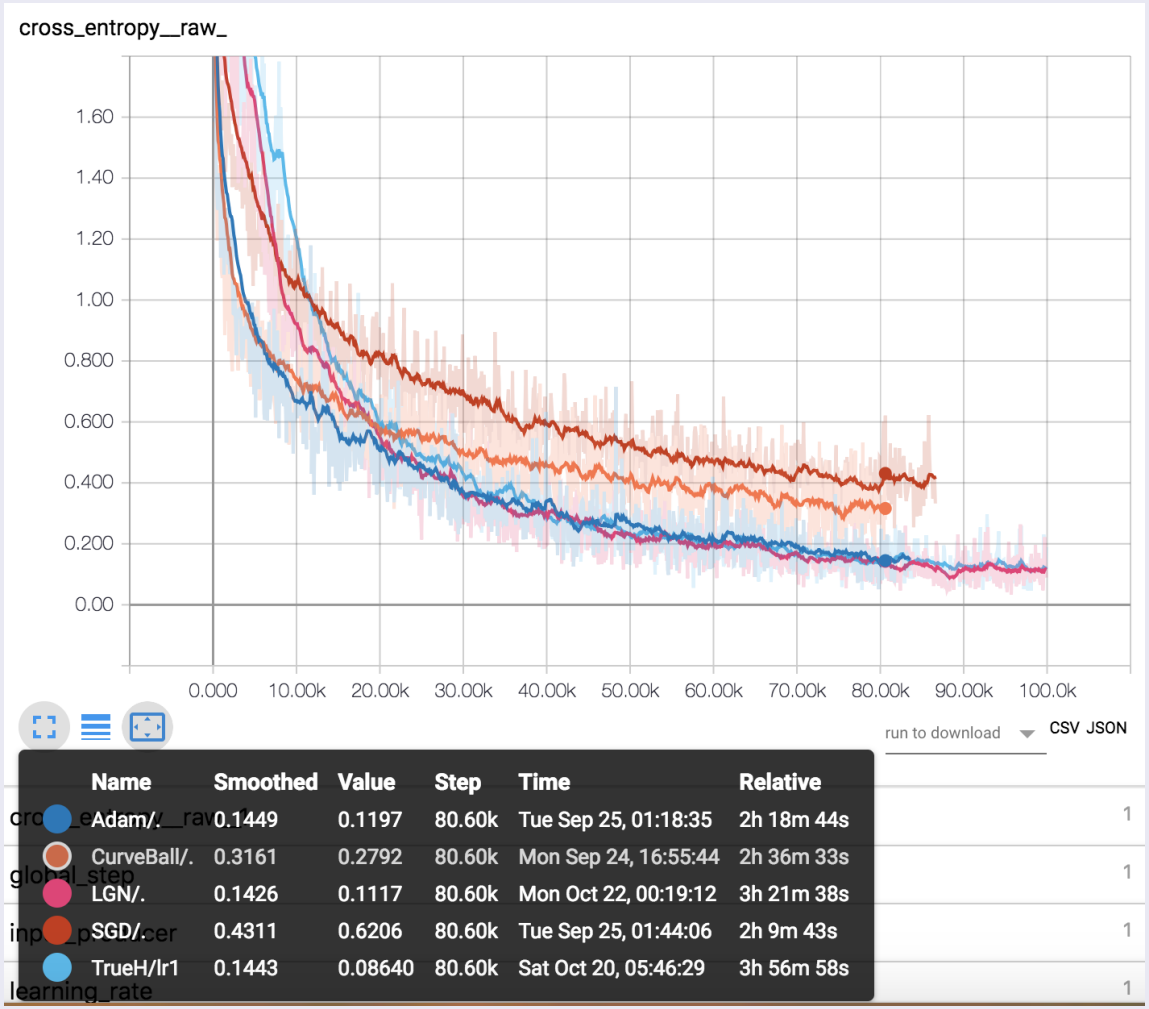
Leaping Stochastic Gradient Descent

Resnet No batch Norm



LSGD - Future MLRG go to optimiser

Resnet batch norm



LSGD - Future MLRG go to optimiser

Basicnet batch norm

End of the Talk

Q & A

Thank you for your time:) diego@robots.ox.ac.uk