The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies

Ronen Basri[1] **David Jacobs**[2] **Yoni Kasten**[1] **Shira Kritchman**[1]

[1]Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel
[2]Department of Computer Science, University of Maryland, College Park, MD

What makes DNs so successful?

Common deep networks seem to defy basic machine learning principles:

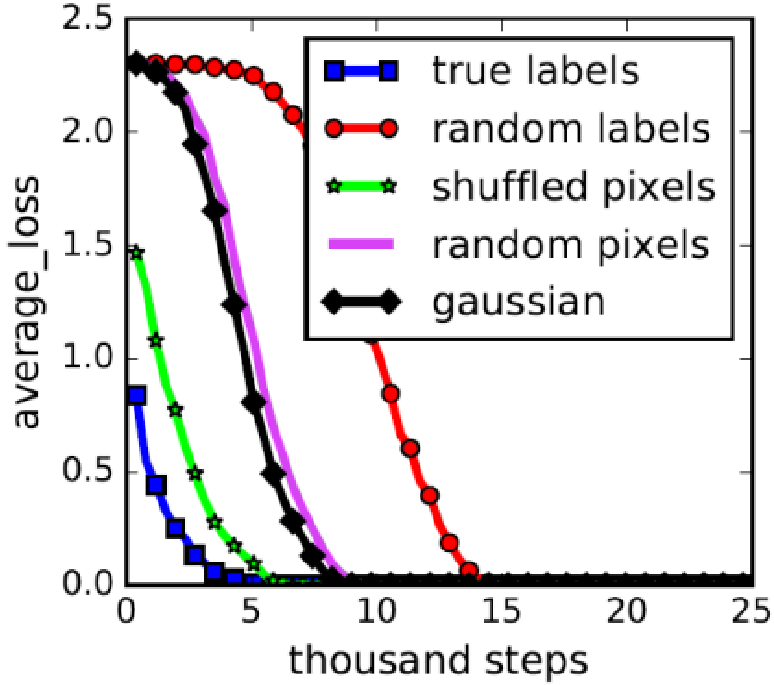
How come over-parameterized networks do not overfit?

- Resnet has 60M learnable parameters (VGG has 140M)
- But ImageNet includes only 1.2M training images

Random relabeling

(Zhang et al., ICLR 2017)

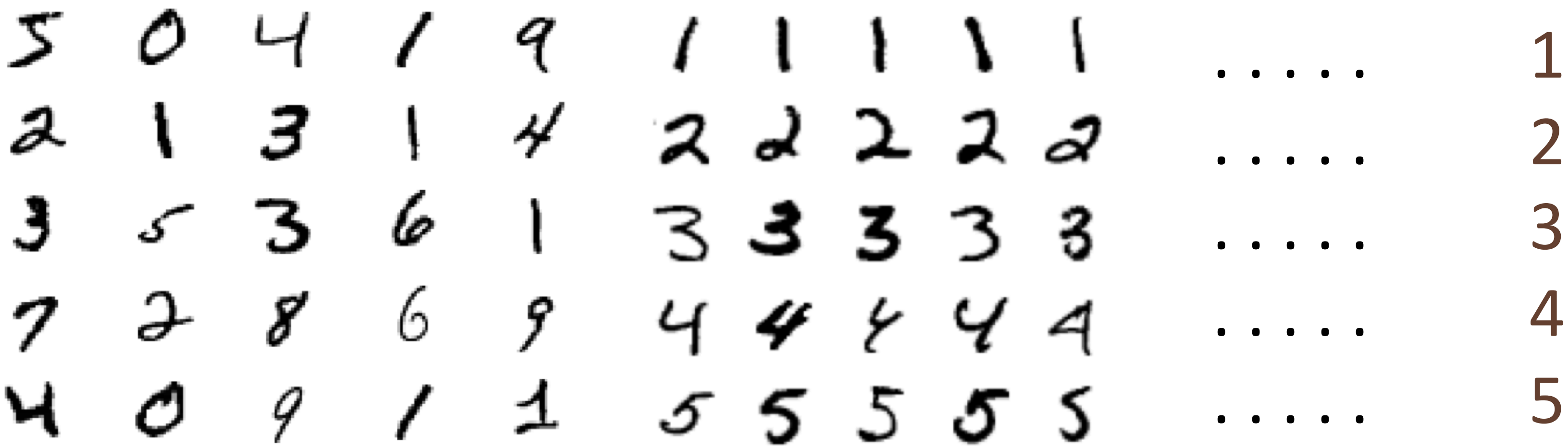
Understanding deep learning requires rethinking generalization



Loss on training (CIFAR10)

Zhang et al., ICLR 2017

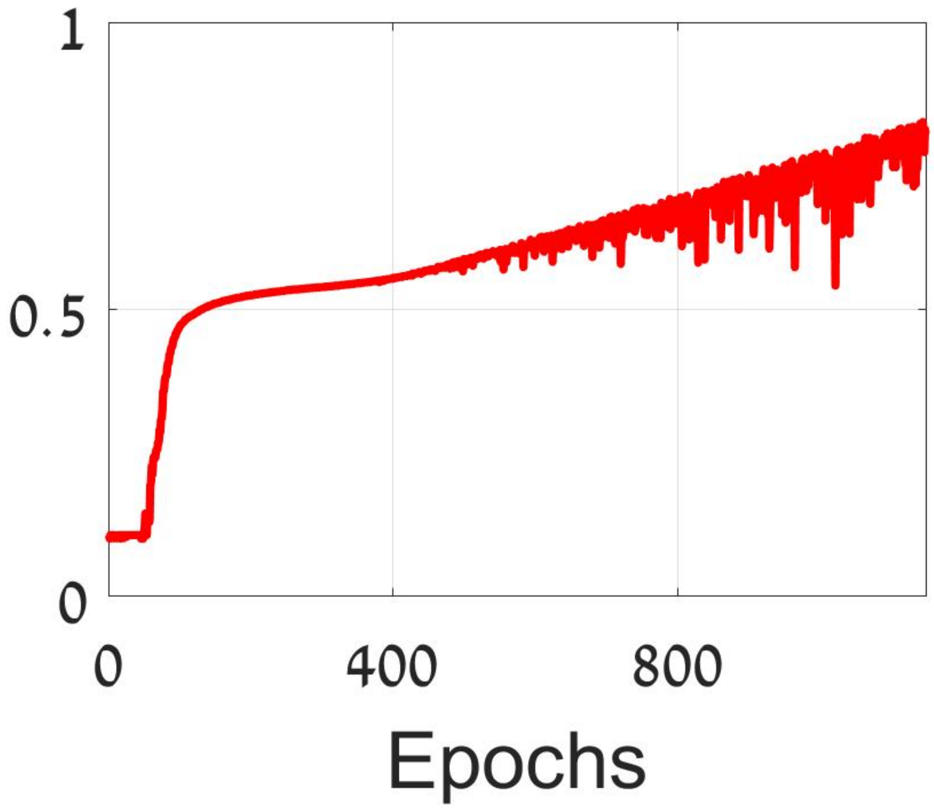
Partial random relabeling



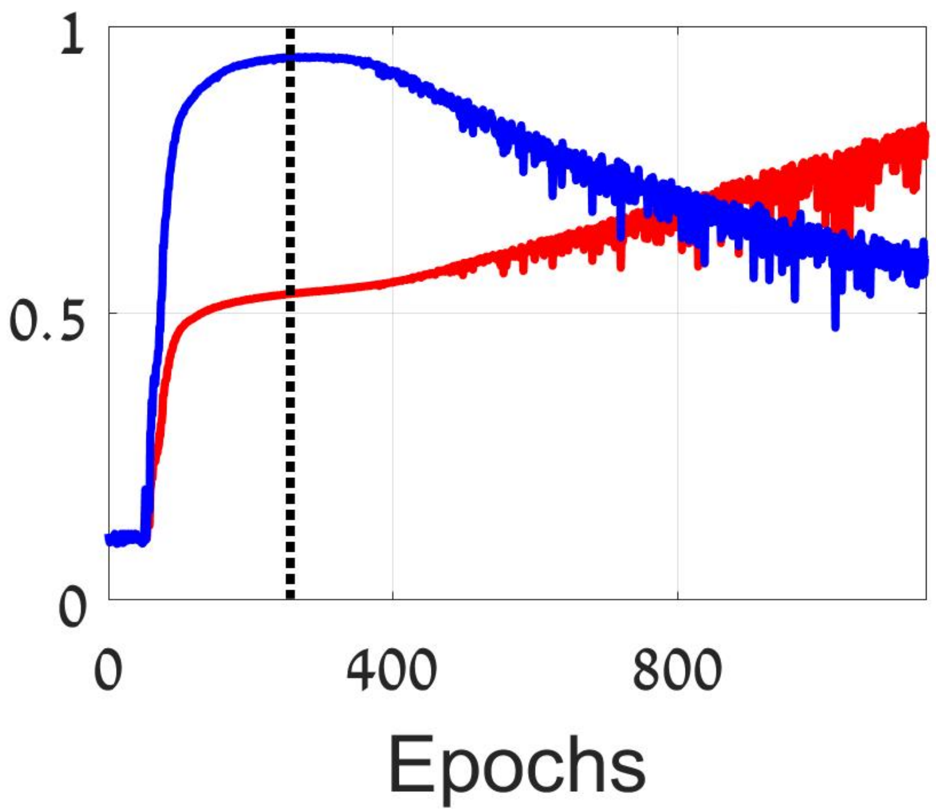
MNIST images

Labels

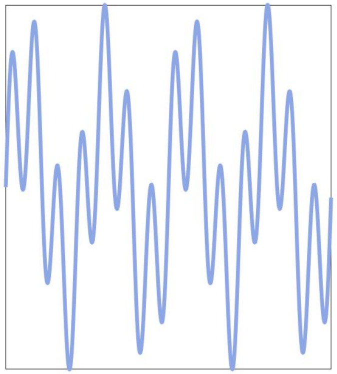
Partial relabeling: training



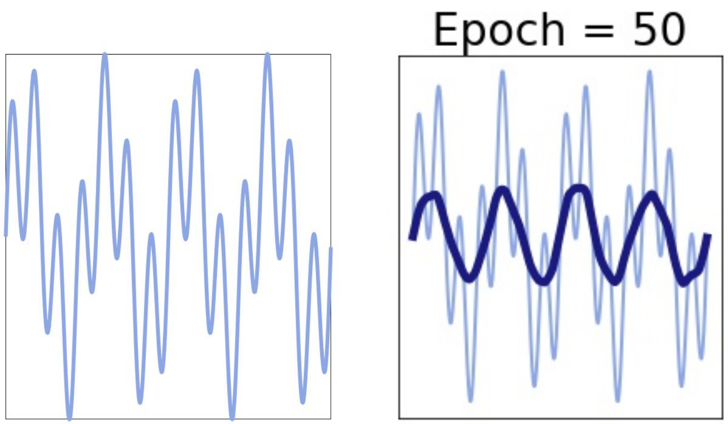
Partial relabeling: test against true labels



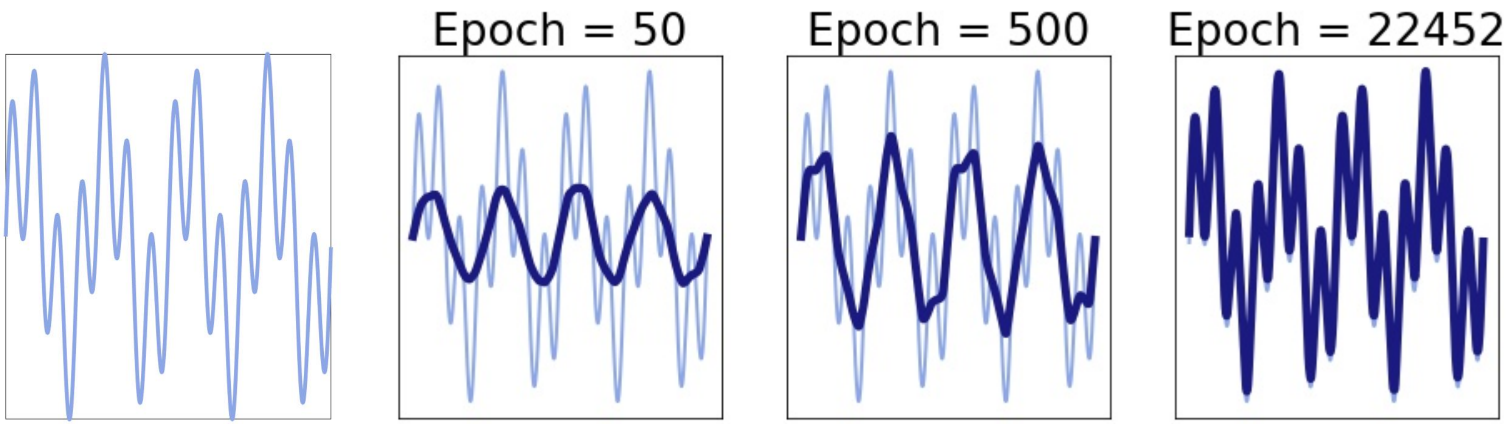
Frequency bias



Frequency bias



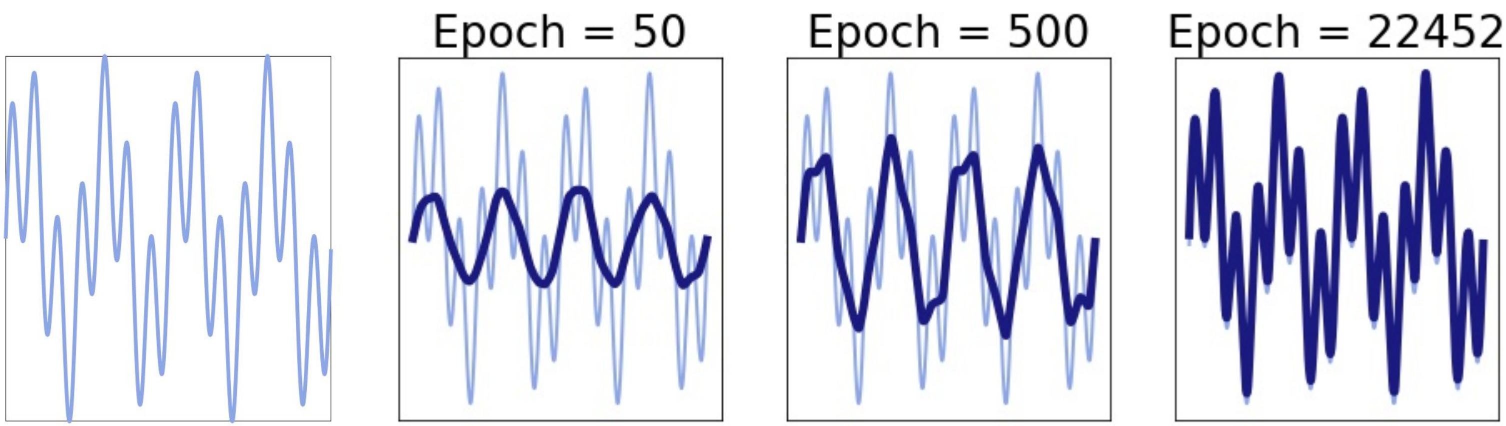
Frequency bias



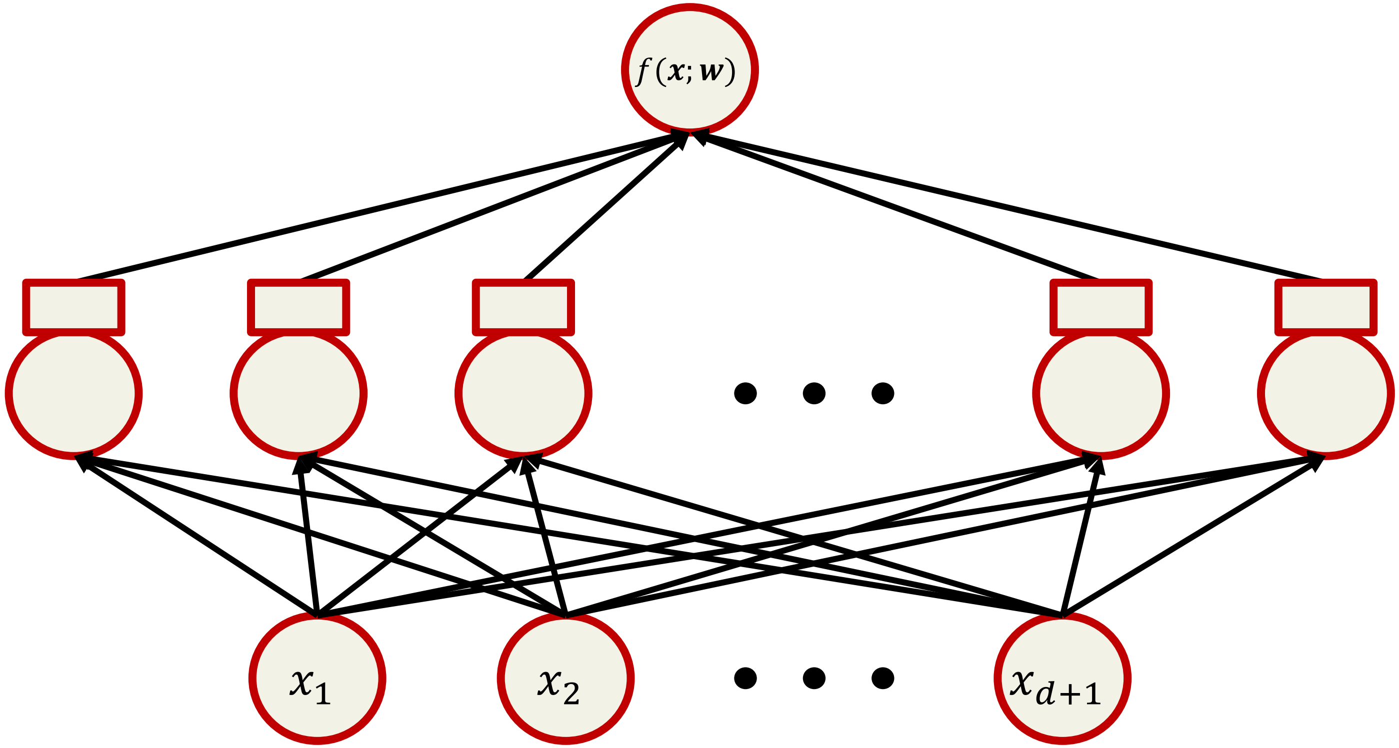
Frequency bias

Can we explain this frequency bias?

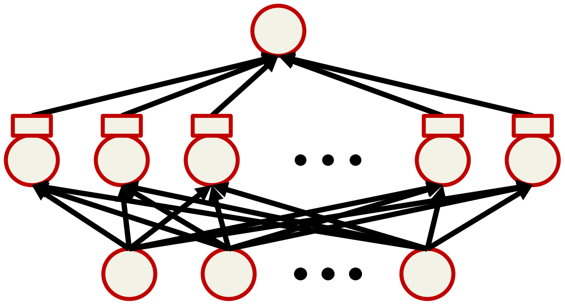
How long should it take to learn a single frequency?



Two-layer network



Two-layer network



$$f(\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma(\boldsymbol{w}_r^T \boldsymbol{x}), \qquad \sigma(x) = \max(x, 0)$$

MSE Loss: $L(\boldsymbol{w}) = \frac{1}{2} \sum_{i=1}^{n} (\boldsymbol{y_i} - f(\boldsymbol{x_i}, \boldsymbol{w}))^2$

Network predictions as a "linear" system

Write predictions for training data as a linear operation:

$$\boldsymbol{u}(t) = \begin{pmatrix} u_1 = f(\boldsymbol{x_1}, \boldsymbol{w}) \\ \vdots \\ u_n = f(\boldsymbol{x_n}, \boldsymbol{w}) \end{pmatrix} = Z^T \boldsymbol{w}$$

where we define $Z = Z(t)$ as

$$Z^T = \frac{1}{\sqrt{m}} \begin{pmatrix} a_1 \mathbb{I}_{11} \boldsymbol{x_1} & \cdots & a_m \mathbb{I}_{m1} \boldsymbol{x_1} \\ \vdots & & \vdots \\ a_1 \mathbb{I}_{1n} \boldsymbol{x_n} & \cdots & a_m \mathbb{I}_{mn} \boldsymbol{x_n} \end{pmatrix}$$

Back-prop minimizes $\frac{1}{2} \sum_{i=1}^{n} (\boldsymbol{y_i} - Z^T(t) \boldsymbol{w})^2$

Eg., Du et al., ICLR 2019

GD for linear systems

Suppose we want to minimize $\frac{1}{2}\|\boldsymbol{y} - Z\boldsymbol{w}\|^2$ using gradient descent with $\boldsymbol{w}^{(0)} = 0$

$$\boldsymbol{u}^{(1)} = -\eta Z^T Z \boldsymbol{y}$$

$$\boldsymbol{u}^{(2)} = -2\eta Z^T Z \boldsymbol{y} - \eta^2 (Z^T Z)^2 \boldsymbol{y}$$

$$\boldsymbol{u}^{(3)} = -3\eta Z^T Z \boldsymbol{y} - 3\eta^2 (Z^T Z)^2 \boldsymbol{y} - \eta^3 (Z^T Z)^3 \boldsymbol{y}$$

…

The kernel

Define

$$H(t) = Z^T Z$$

Du et al. 2018's observation:
when the network is massively over-parameterized
$H(t) \sim H^\infty$, where

$$H_{ij}^\infty = \mathbb{E}_{\boldsymbol{w} \sim \mathcal{N}(0,\kappa^2)} H_{ij} = \frac{1}{2\pi} \boldsymbol{x}_i^T \boldsymbol{x}_j (\pi - \cos^{-1}(\boldsymbol{x}_i^T \boldsymbol{x}_j))$$

What are the eigenvectors?

If the training data is distributed uniformly on the hyper-sphere then H^{∞} represents a convolution

$$K * f(\boldsymbol{u}) = \int_{S^d} K(\boldsymbol{u}^T \boldsymbol{v}) f(\boldsymbol{v}) d\boldsymbol{v}$$

Therefore, eigenvectors are spherical harmonics

Recall that

$$H_{ij}^{\infty} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{2\pi} \boldsymbol{x}_i^T \boldsymbol{x}_j (\pi - \cos^{-1}(\boldsymbol{x}_i^T \boldsymbol{x}_j))$$

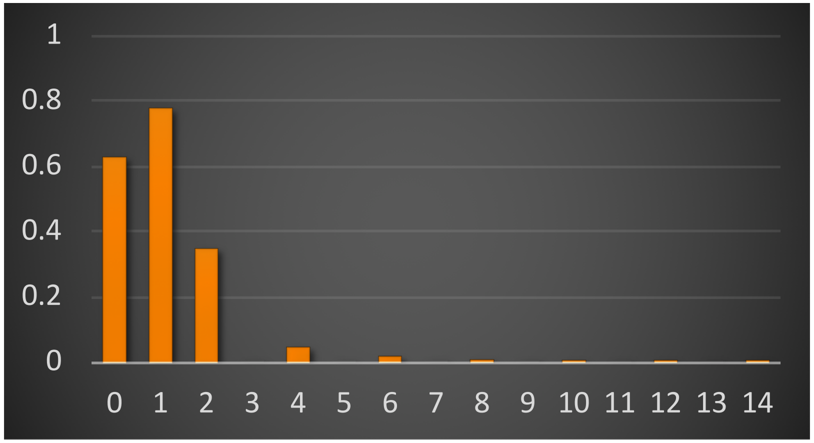
(See also Xie et al., 2017)

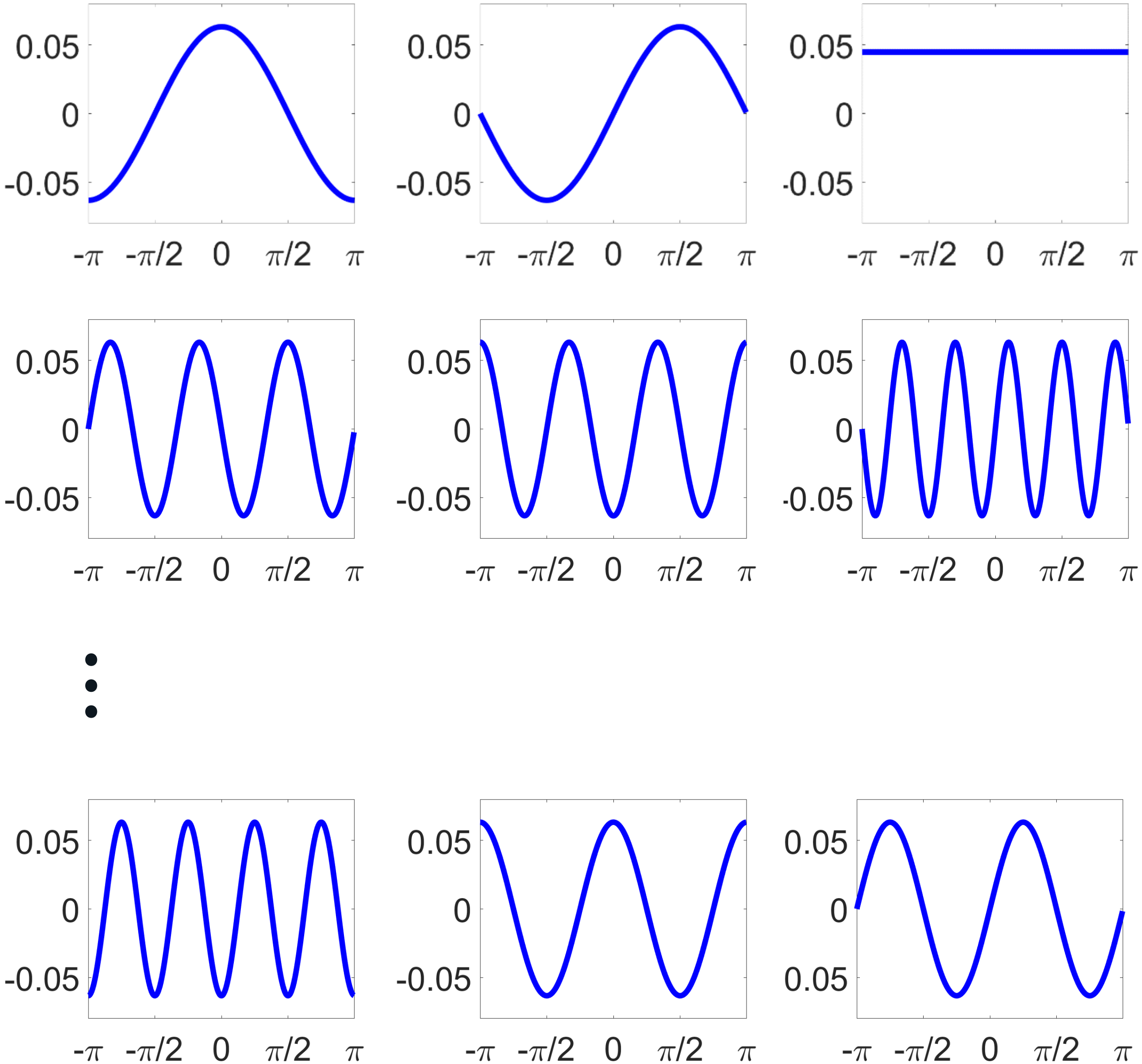
Eigenvalues (d=1)

Eigenvectors are the Fourier series

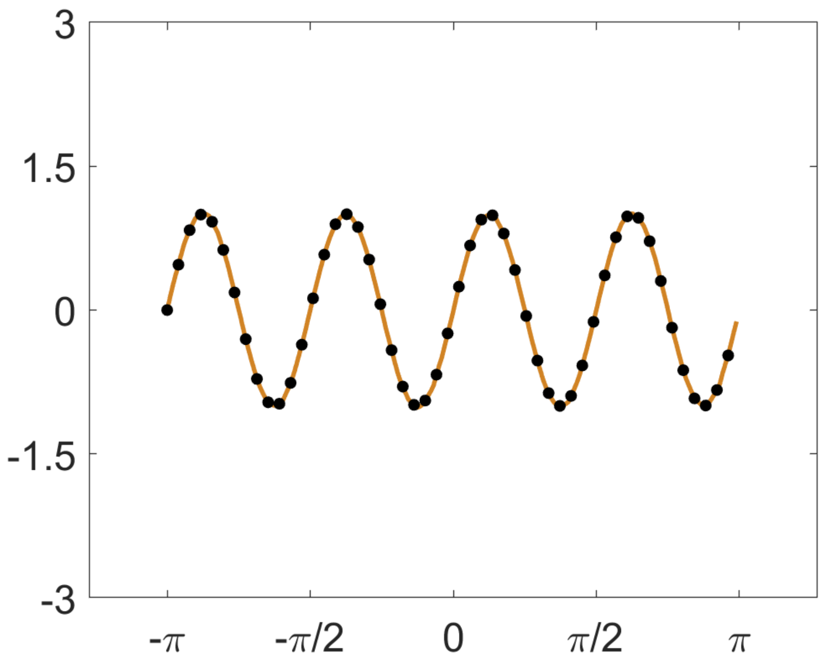
$$a_k = \begin{cases} 2/\pi & k = 0 \\ \pi/4 & k = 1 \\ \frac{2(k^2+1)}{\pi(k^2-1)^2} & k \geq 2, \text{even} \\ 0 & k \geq 2, \text{odd} \end{cases}$$

Odd frequencies vanish!!

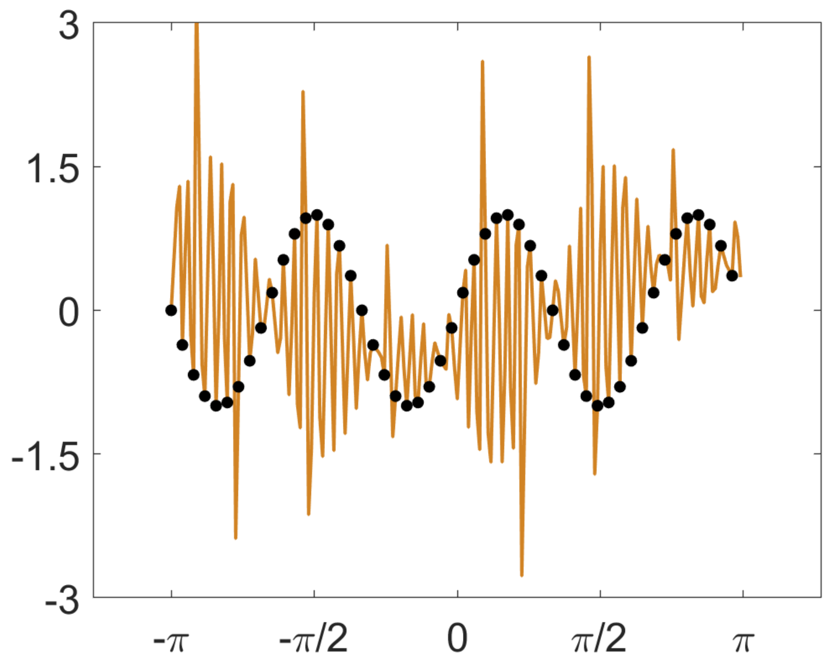



0.05
0
-0.05
-π
-π/2
0
π/2
π

Fitting to pure frequency



k=4



k=3

Convergence times

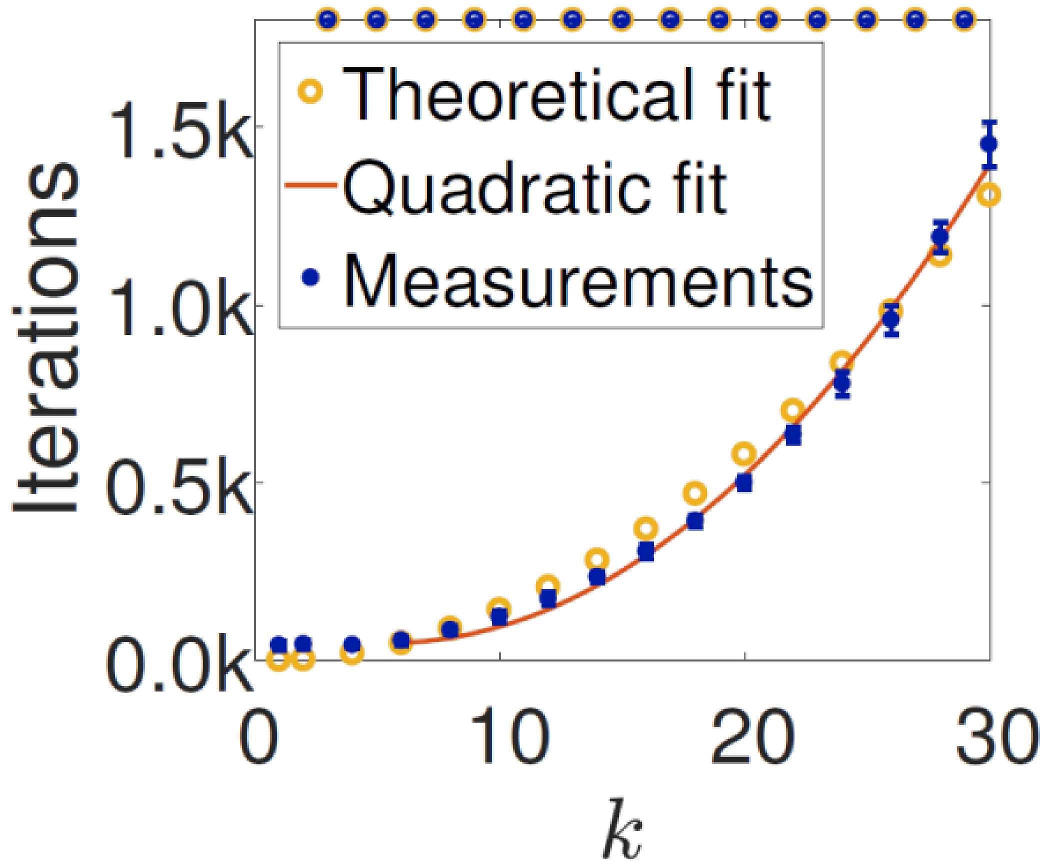
Relying on Arora et al., 2019, the number of iterations required to achieve accuracy δ:

$$t_i > \frac{-\log(\delta + \varepsilon)}{\eta\lambda_i} = O\left(\frac{1}{\lambda_i}\right)$$

Even frequencies: $t_i \gtrsim \frac{\pi\left(k^2-1\right)^2}{2(k^2+1)} = O(k^2)$

Odd frequencies: $t_i \to \infty$

Convergence times

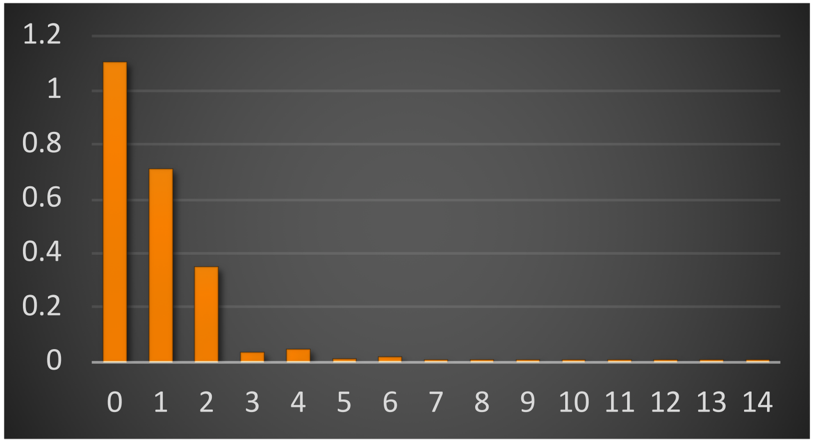


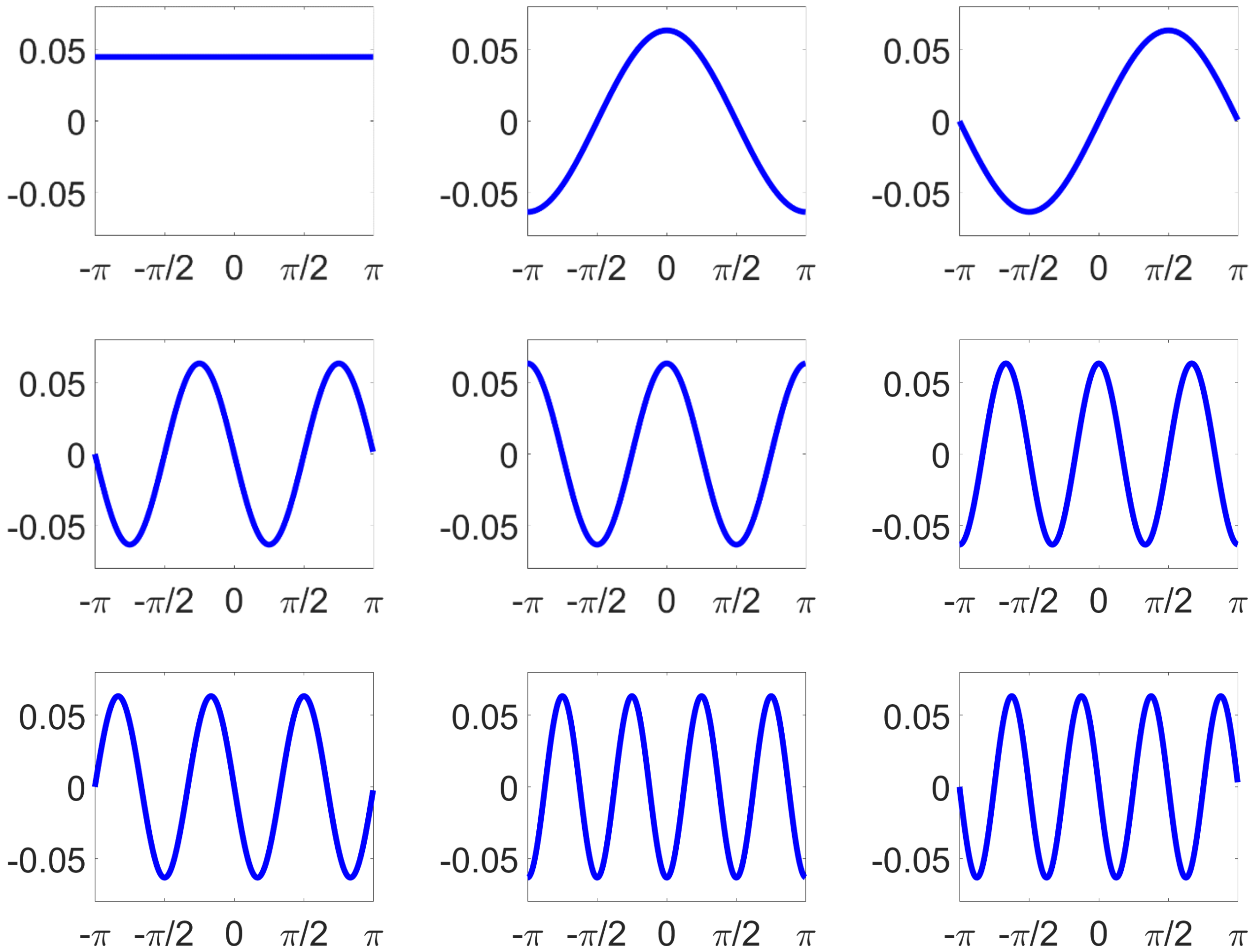
Adding bias

Adding bias rectifies the problem

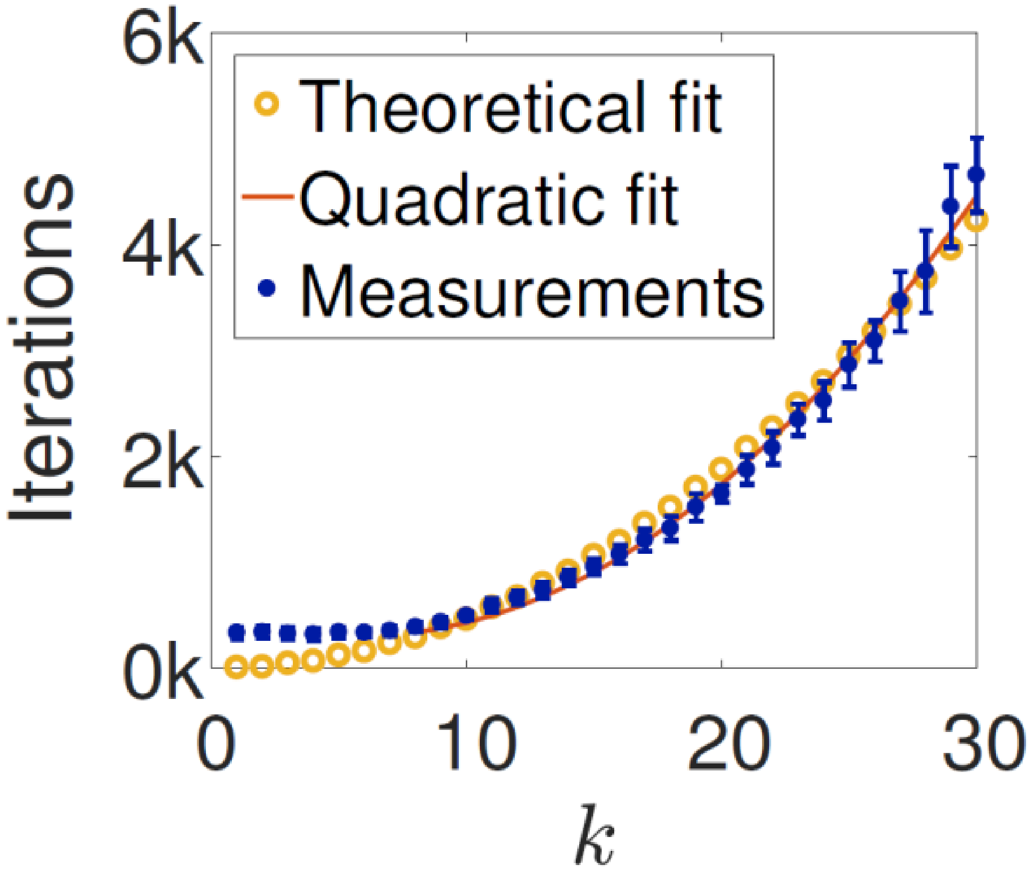
$$\bar{H}^{\infty}_{ij} = \frac{1}{4\pi}(\boldsymbol{x}_i^T \boldsymbol{x}_j + 1)(\pi - \cos^{-1}(\boldsymbol{x}_i^T \boldsymbol{x}_j))$$

$$\bar{a}_k = \begin{cases} 1/\pi + \pi/4 & k = 0 \\ 1/\pi + \pi/8 & k = 1 \\ \dfrac{2(k^2+1)}{\pi(k^2-1)^2} & k \geq 2, \text{even} \\ \dfrac{1}{\pi k^2} & k \geq 2, \text{odd} \end{cases}$$

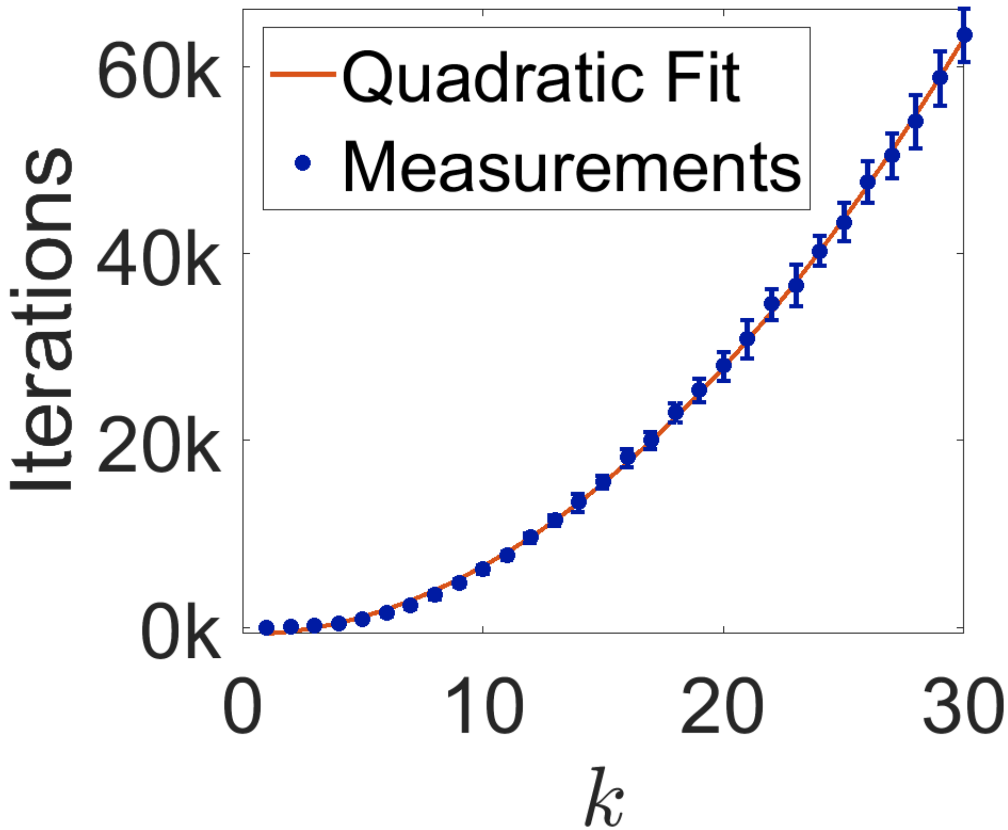



0.05
0
-0.05
-π
-π/2
0
π/2
π

Convergence times (with bias)



Convergence
(Resnet-10, 1D data embedded in $\mathcal{R}^{30}$)



Higher dimension

Eigenvectors are spherical harmonics

Eigenvalues can be derived using the Gegenbauer polynomials

With no bias odd frequencies $k \geq 2$ vanish

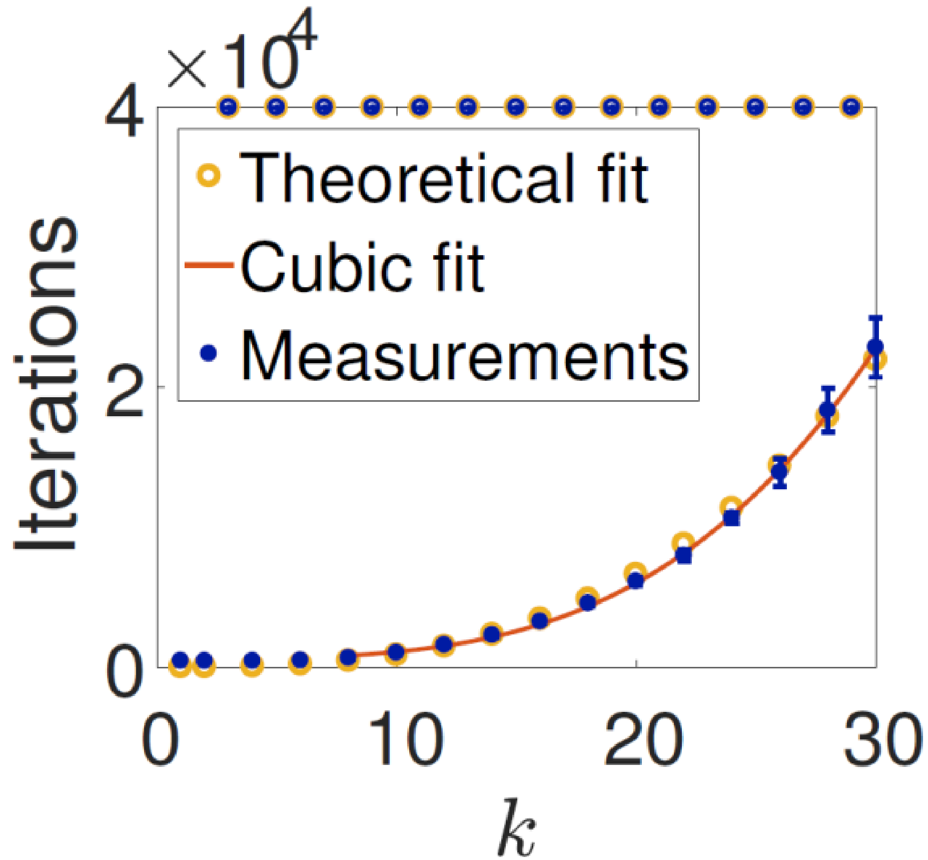
Convergence time for high frequencies increase exponentially in the dimension

Eigenvalues for higher dim.

$$\bar{a}_k = \begin{cases} \frac{c}{2}\left(\frac{1}{d2^{d+1}}\binom{d}{\frac{d}{2}} + \frac{2^{d-1}}{d\binom{d-1}{\frac{d}{2}}} - \frac{1}{2}\sum_{q=0}^{\frac{d-2}{2}}(-1)^q\binom{\frac{d-2}{2}}{q}\frac{1}{2q+1}\right) & k=0 \\ \frac{c}{2}\sum_{q=\lceil k/2\rceil}^{k+\frac{d-2}{2}} b_q\left(\frac{1}{4q+2}+\frac{1}{4q}\left(1-\frac{1}{2^{2q}}\binom{2q}{q}\right)\right) & k=1 \\ \frac{c}{2}\sum_{q=\lceil k/2\rceil}^{k+\frac{d-2}{2}} b_q\left(\frac{-1}{4q-2k+2}+\frac{1}{4q-2k+4}\left(1-\frac{1}{2^{2q-k+2}}\binom{2q-k+2}{\frac{2q-k+2}{2}}\right)\right) & k\geq 2, \text{even} \\ \frac{c}{2}\sum_{q=\lceil k/2\rceil}^{k+\frac{d-2}{2}} b_q\left(\frac{1}{4q-2k+2}\left(1-\frac{1}{2^{2q-k+1}}\binom{2q-k+1}{\frac{2q-k+1}{2}}\right)\right) & k\geq 2, \text{odd} \end{cases}$$

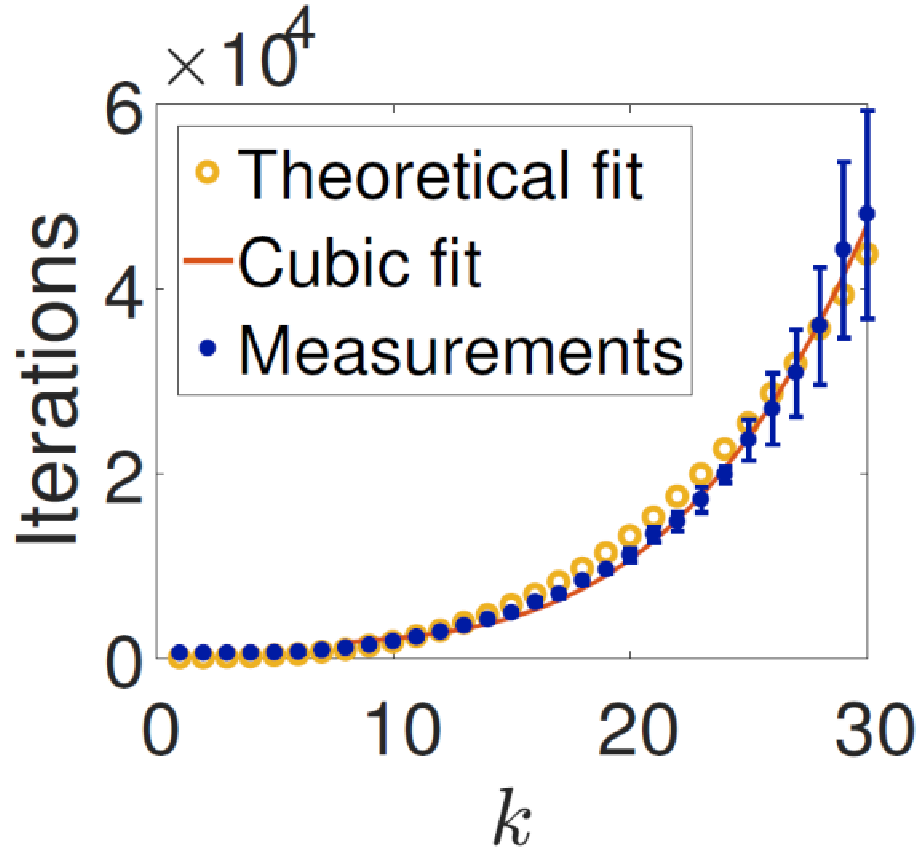
$c = \frac{(-1)^k 2\pi^{d/2}}{2^k\Gamma\left(k+\frac{d}{2}\right)d}$ and $b_q = (-1)^q\binom{k+\frac{d-2}{2}}{q}\frac{(2q)!}{k!}$

2D, no bias



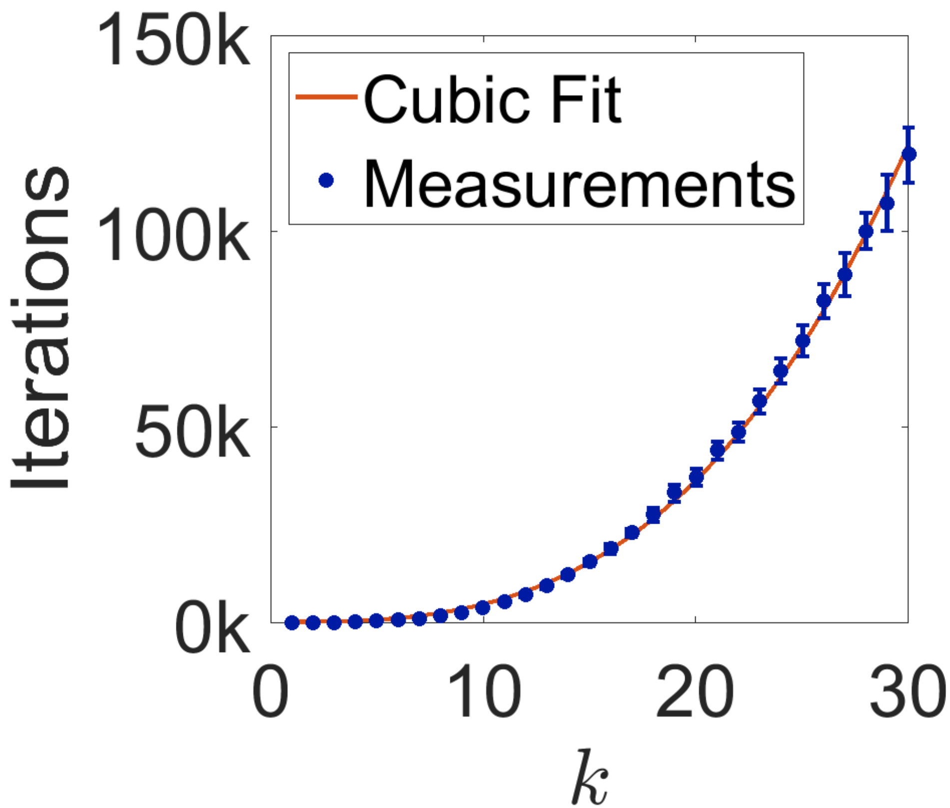
×10^4
4
2
0
Theoretical fit
Cubic fit
Measurements
Iterations
0
10
20
30
k

2D, with bias



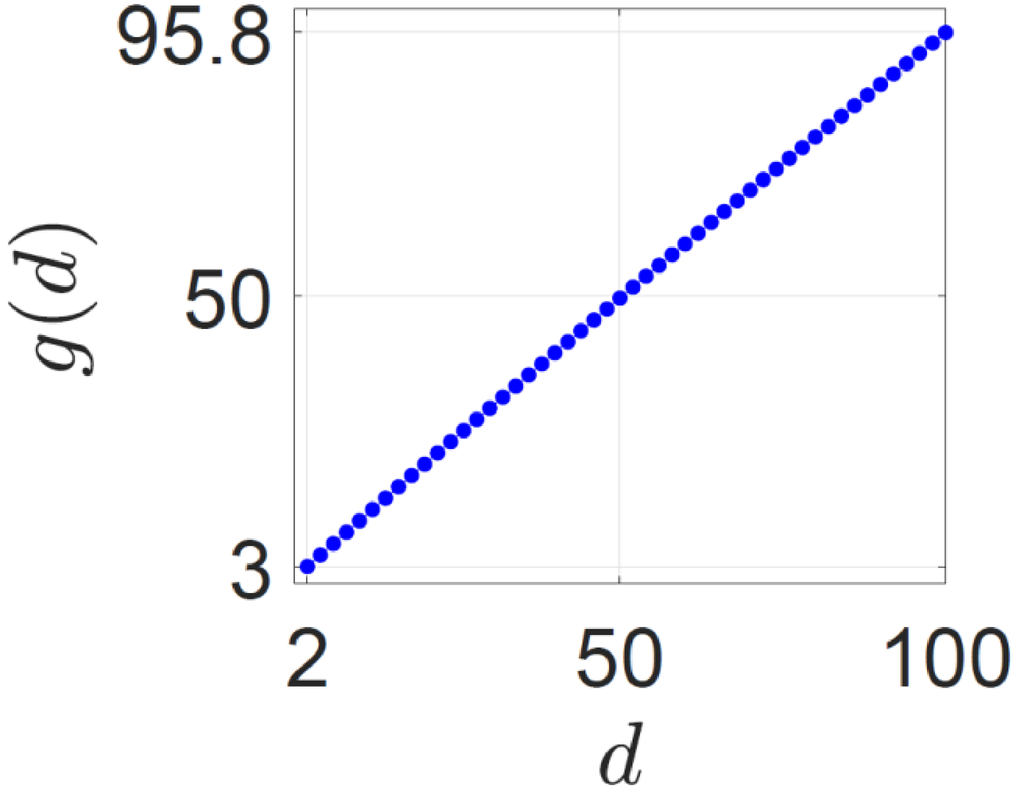
×10^4
6
4
2
0
Iterations
0
10
20
30
k
Theoretical fit
Cubic fit
Measurements

2D, deep

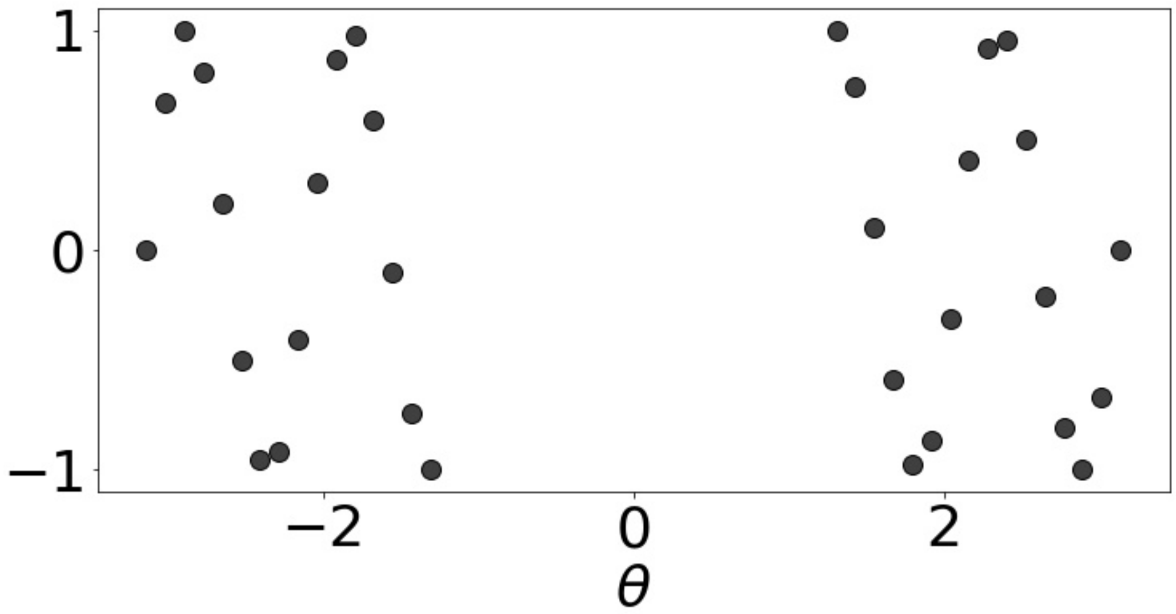


150k
100k
50k
0k
Iterations
0
10
20
30
k
Cubic Fit
Measurements

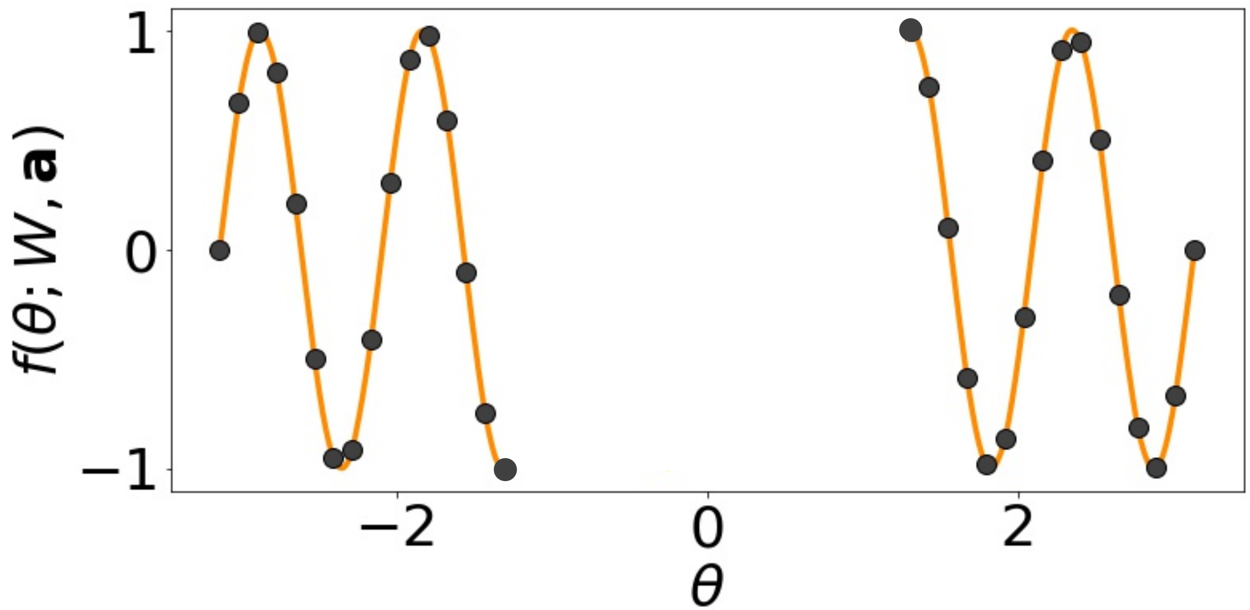
Higher dimension



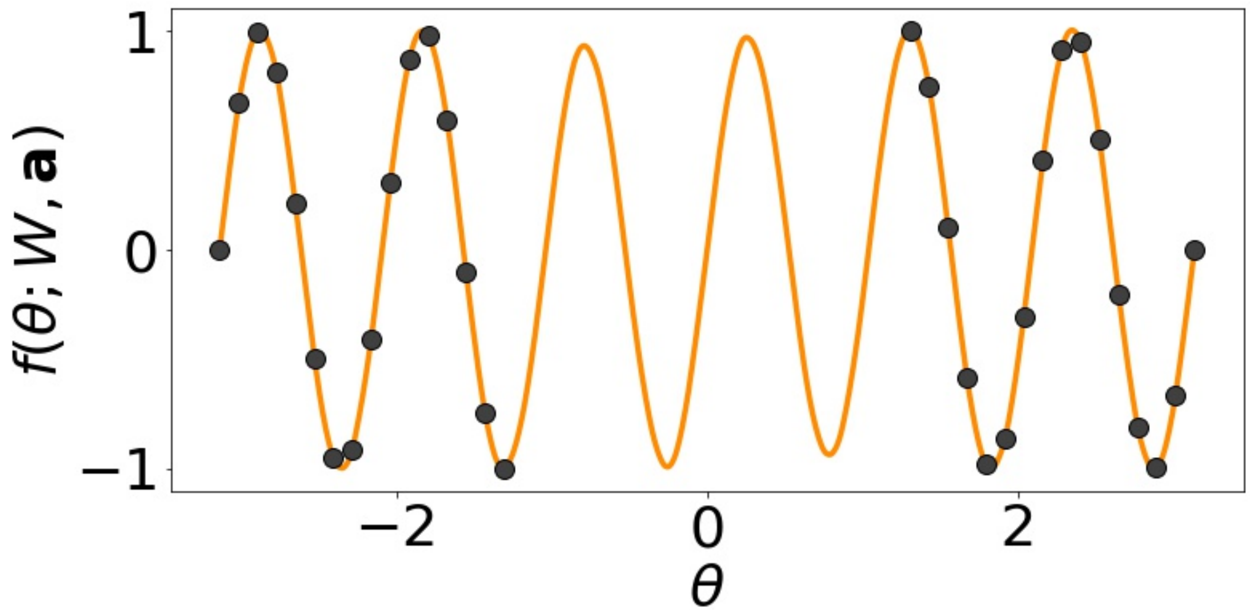
What will the network learn?



What will the network learn?



What will the network learn?



Conclusion

Deep networks show a frequency bias –
low frequencies appear to be learned faster than high frequencies

Our work determines the rate of learning analytically, as a function of frequency, for over-parameterized, two-layer network

It further points out that two-layer, bias free networks are non-universal, and cannot represent odd frequencies