Preamble

Working hypothesis in ongoing research:
Key principle in ML architectures: (compositional) sparsity

The thesis of this course — its conceptual backbone — is that compositional sparsity is key in ML. The idea that sparsity is fundamental is new — it is a conjecture which may be wrong. We saw the key role of it first in approximation theory, then in generalization. What about optimization?

Sparsity and Generalization

- Sparsity in the weight matrices implies generalization bounds that are order of magnitude better than for deep networks with dense matrices, see class 22 and 21 (Galanti, Xu, Galanti, Poggio, NeurIps 2023)

- *Thus sparsity seems to be a key principle for approximation and for generalization*

Statistical Learning Theory

and

Applications

2024

Class 15

Deep Learning Theory: Optimization

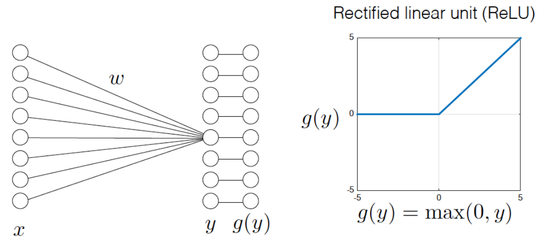
Optimization

Optimization: fitting parameters of the neural network, that approximates the unknown function,
using data. This is also called training.

We do not have a “complete”/coherent theory as yet:
today class will consist of several, not well integrated parts!

Optimization by SGD

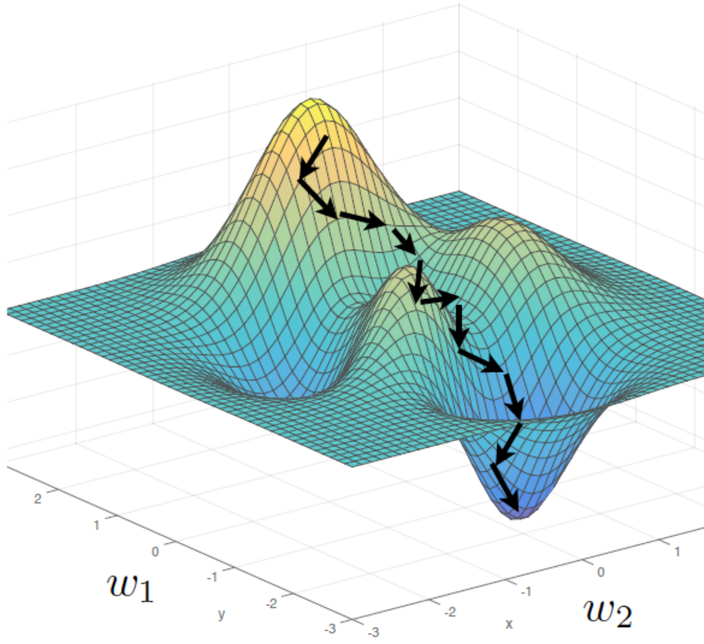
$$\underset{\mathbf{w}}{\operatorname{argmin}} \sum_i \ell(\mathbf{z}_i, f(\mathbf{x}_i; \mathbf{w})) = L(\mathbf{w})$$



One iteration of gradient descent:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta_t \frac{\partial L(\mathbf{w}^t)}{\partial \mathbf{w}}$$

learning rate



A motivation for our ongoing work: what is the role of sparse compositionally in DNNs optimization?

Approximation deals mostly with *existence* results: there exist sparse networks that can approximate any function without curse of dimensionality.

The next obvious question: under which conditions *can optimization —in particular SGD — learn* these sparse networks?

It is clearly impossible always to learn the underlying function (think about a training sets of random names and random telephone numbers).

The guiding *conjecture* in our ongoing work — also in this class — is that sparse compositionally is important not only for approximation but also for optimization (remember the non-exponential requirement on the parameters)

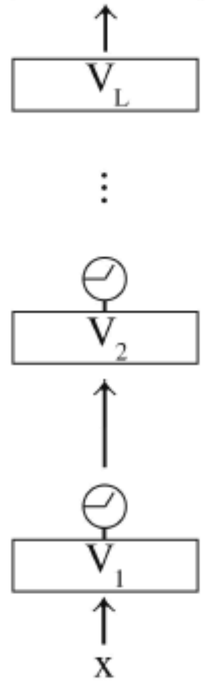
An important hint from compositional sparsity

…the key step in optimization is “selecting” the compositionally sparse decomposition which always exists but is never unique… as in

$$F = f_5 \circ f_4 \circ f_3 \circ f_2 \circ f_1 x = g \circ f_3 \circ f_2 \circ f_1 x = \dots$$

We expect different architectures and different optimization algorithms to bias the choice of the decomposition — for instance, the number of layers — and units in each layer — affects the number of constituent functions.

Non-uniqueness of the decomposition is a key problem in understanding learning with deep networks.



Today's plan

1. Loss landscape
2. Intuition about SGD convergence to global, degenerate minima
3. Dynamics of Optimization/training under the square loss
4. Gradient Descent and SGD
5. Low-rank bias and SGD noise
6. Neural Collapse
7. Architectures
8. Appendix: remarks

Research
A SCIENCE PARTNER JOURNAL




RESEARCH ARTICLE

Dynamics in Deep Classifiers Trained with the Square Loss: Normalization, Low Rank, Neural Collapse, and Generalization Bounds

Mengjia Xu[1,2], Akshay Rangamani[1], Qianli Liao[1], Tomer Galanti[1], and Tomaso Poggio[1*]

[1]Center for Brains, Minds and Machines, Massachusetts Institute of Technology, Cambridge, MA, USA.
[2]Division of Applied Mathematics, Brown University, Providence, RI, USA.

*Address correspondence to: tp@ai.mit.edu

Citation: Xu M, Rangamani A, Liao Q, Galanti T, Poggio T. Dynamics in Deep Classifiers Trained with the Square Loss: Normalization, Low Rank, Neural Collapse, and Generalization Bounds. Research 2023;6:Article 0024. https://doi.org/10.34133/research.0024

Submitted 26 July 2022
Accepted 24 November 2022
Published 8 March 2023

GD optimization induces a gradient dynamical system

$$L = \sum_n^N (y_n - f(W_K, ..., W_1; x_n))^2$$

for regression

Square loss

Deep network with RELUs

$$\dot{W}_k^{i,j} = \sum_n^N (y - f_n) \frac{\partial f_n}{\partial W_k^{i,j}}$$

Commonly used is cross-entropy

Exponential loss
Deep network with RELUs,

$$L = \sum_n^N e^{-y_n f(W_K, ..., W_1; x_n)}$$

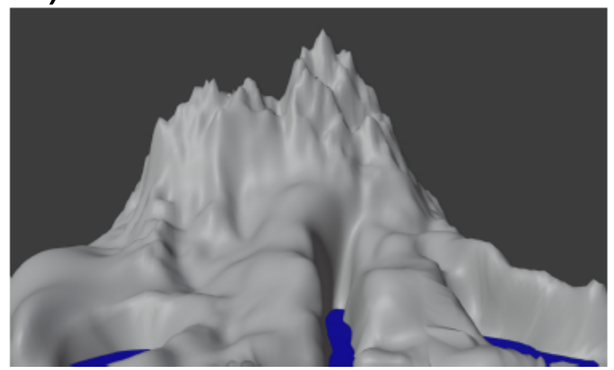
$$\dot{W}_k^{i,j} = \sum_n^N e^{-y_n f(x_n)} \frac{\partial f}{\partial W_k^{i,j}}$$

Loss landscape

Let $g(w)$ be the function given by the neural net with W parameters (and smooth RELU). Let the loss function used in training be

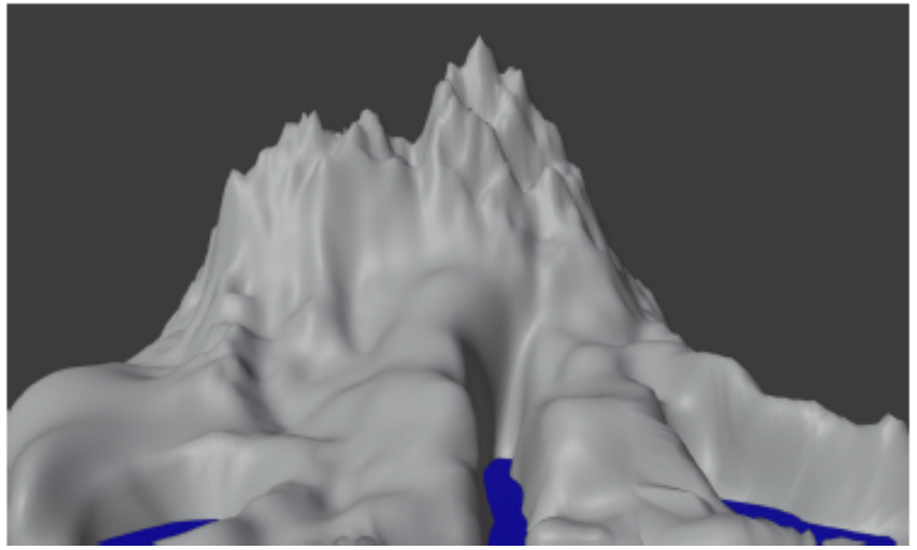
$$L(W) = \sum_{n}^{N} (g(x_n) - y_n)^2$$

Theorem (Y. Cooper, 2018). In the setting above, the global minimum of L is generically (that is, possibly after an arbitrarily small change to the data set) equal to 0, and the set of global minima M is a nonempty smooth $W - N$ dimensional submanifold of $\mathscr{R}^W$(N is the number of training data).



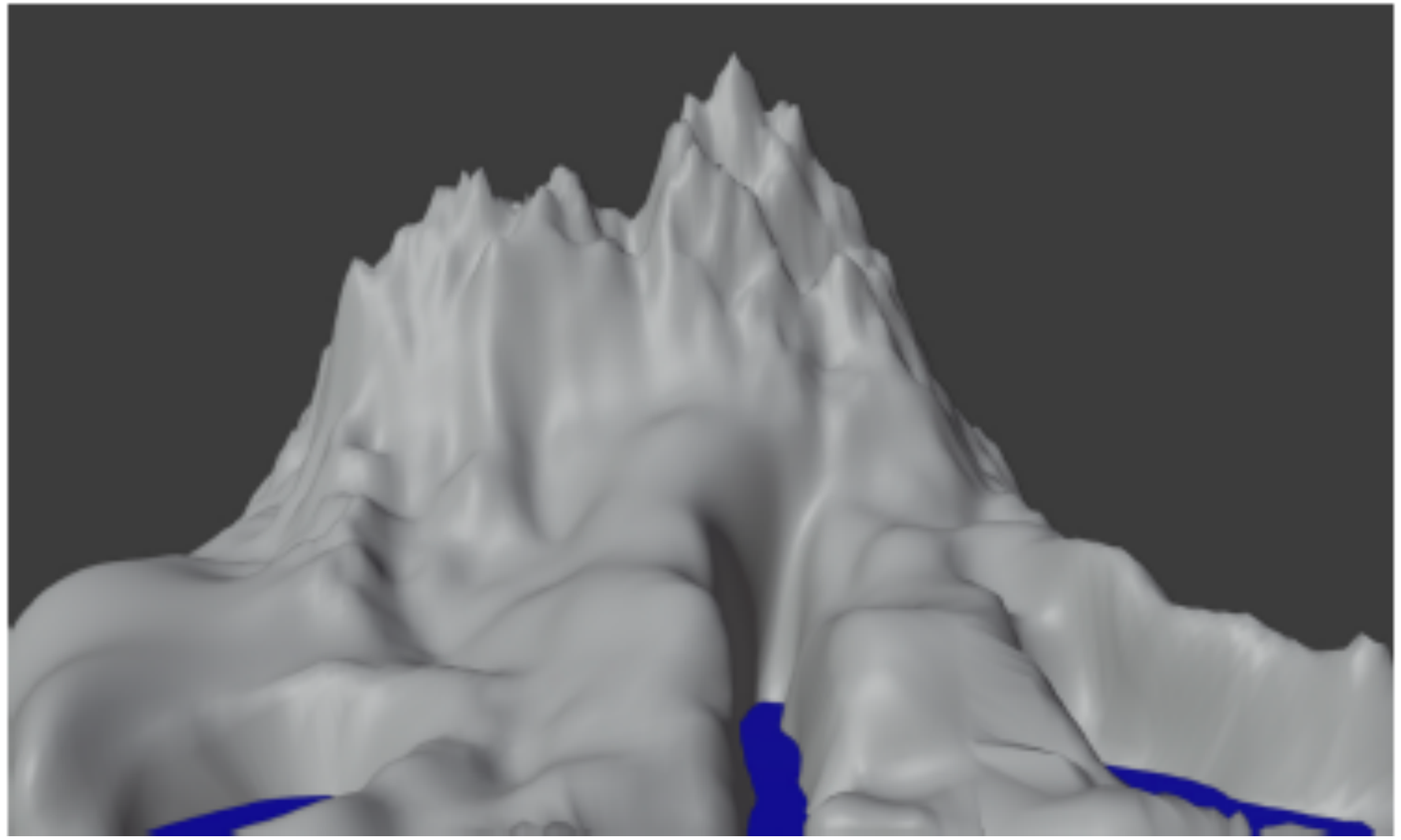
Loss landscape

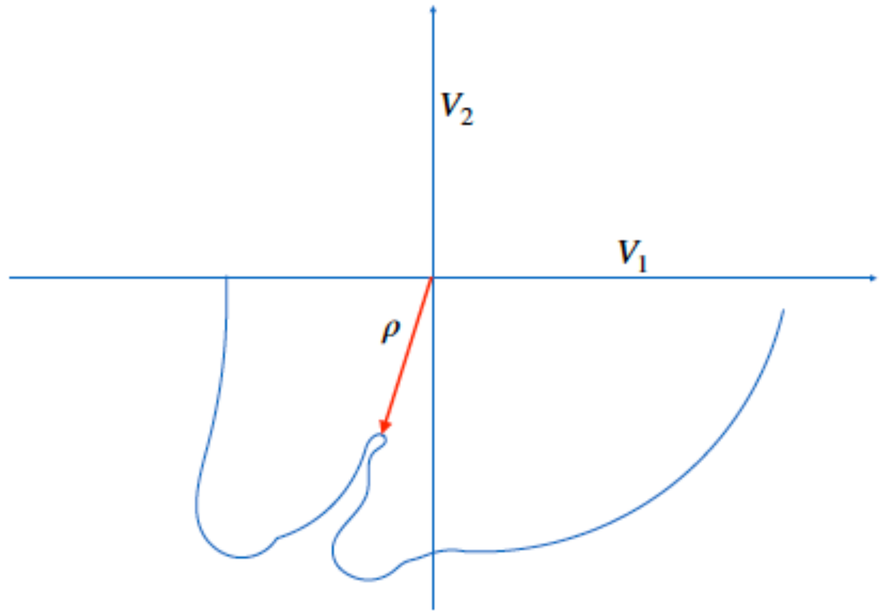
$$L(W) = \sum_{n}^{N} (g(x_n) - y_n)^2$$



Theorem (Nguyen) For the class of deep overparameterized neural nets (where the first hidden layer has twice more neurons than the number of training samples) all of its global minima are connected within a unique and potentially very large global valley.

Landscape of the loss ($\lambda = 0$)





This is for $\lambda = 0$

Landscape theorems (informal)

Theorem (Cooper)

We assume an overparametrized deep network g with smooth RELU activation functions and square loss. Then the minimizers W^* achieve zero loss and are highly degenerate with dimension $d - N$

Theorem (Nguyễn)

If the first layer of the network has at least $2N$ neurons, where N is the number of training data and if the number of neurons in each subsequent layer decreases, then every sublevel set of the loss is connected.

Theorem+conjecture (N. Bottman, Y. Cooper, A. Lerario)

Let L denote the L2 loss function for a feedforward neural network. For a multiplicative regularizer, if the depth ℓ of the neural network is greater than or equal to 4, not only does $L_\epsilon = L + \lambda\rho$ fail to be Morse, but it has a positive-dimensional locus of degenerate critical points. For additive regularizers the conjecture is that the loss function is Morse

Today’s plan

1. Loss landscape ✓
2. Intuition about SGD convergence to global, degenerate minima
3. Dynamics of Optimization/training under the square loss
4. Gradient Descent and SGD
5. Low-rank bias and SGD noise
6. Neural Collapse
7. Architectures
8. Appendix

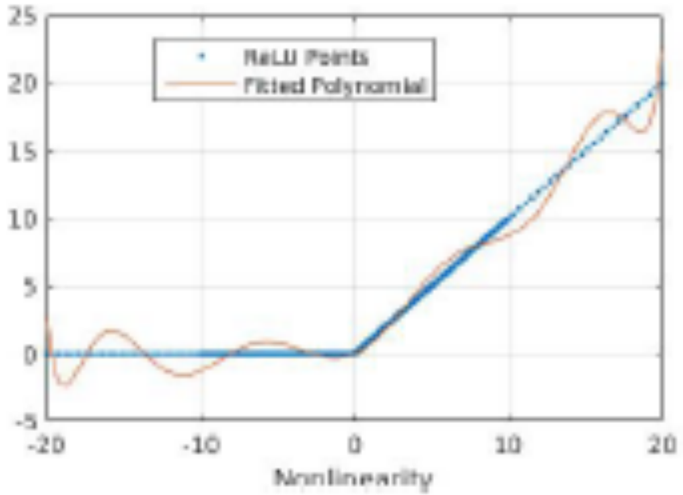
Replacing RELUs with univariate polynomial: the network becomes a polynomial of high degree in its input variables and in the weights

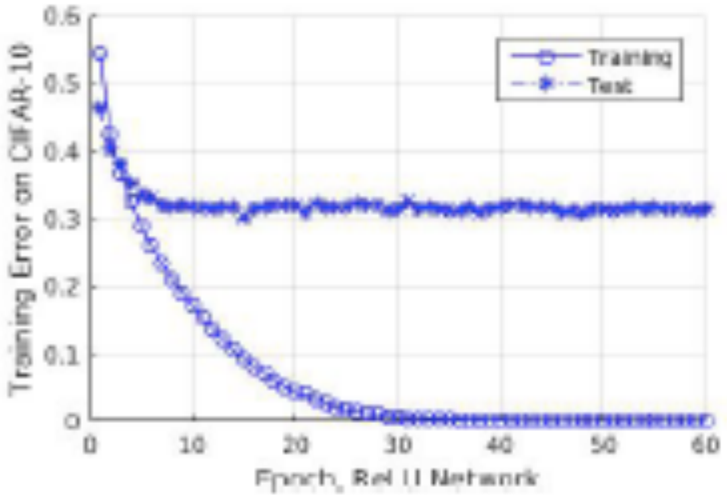
---

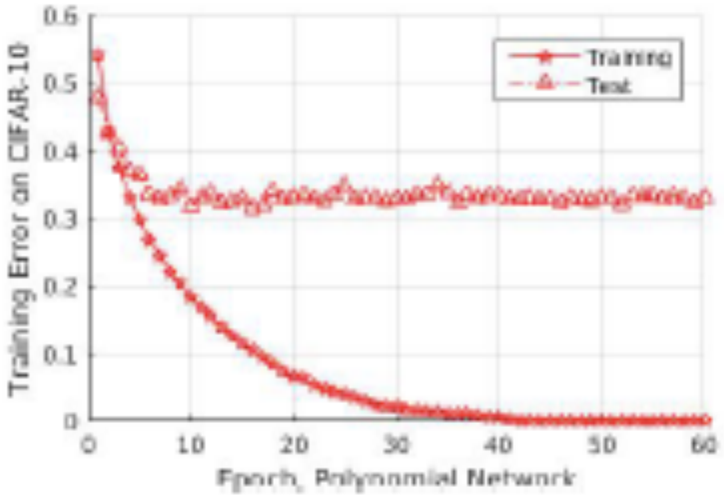
$$g(W;x) = (W_L\sigma(W_{L-1}\cdots\sigma(W_1x))) = W_L\cdots p(W_1x)$$

where $p(z)$ is a univariate polynomial in z

RELU can be approximated (empirical result) by an univariate polynomial







CENTER FOR
Brains
Minds+
Machines

Some intuitions from "soft algebraic geometry": Bezout and critical points of polynomial g

Consider

$$g(x_i) - y_i = 0 \forall i = 1, \cdots, N$$

N equations in W unknowns with $W >> N$

$$\nabla_W \sum_i^N (g(x_i) - y_i)^2 = 0$$

W equations in W unknowns

Bezout theorem: *Global minima are degenerate.*

Conjecture: *Other critical points of the gradient are less degenerate*

SGDL: a version of Langevin equation
(to be technically correct we add tiny amount of “white noise” to GD or SGD)

$$\frac{dg}{dt} = -\eta \nabla_W \mathscr{L} + dB(t)$$ where $\nabla_W \mathscr{L}$ is the full gradient

the Boltzman equation is the asymptotic solution of the SDE above

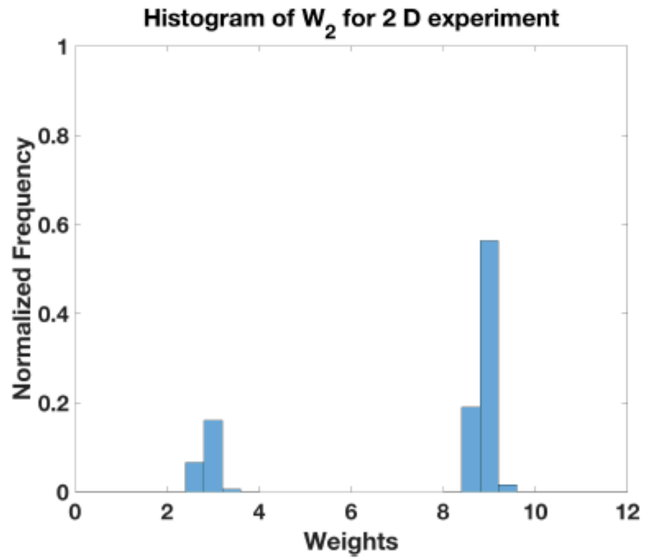
$$P(g) \approx \frac{1}{Z} e^{-\frac{\mathscr{L}}{T}}$$

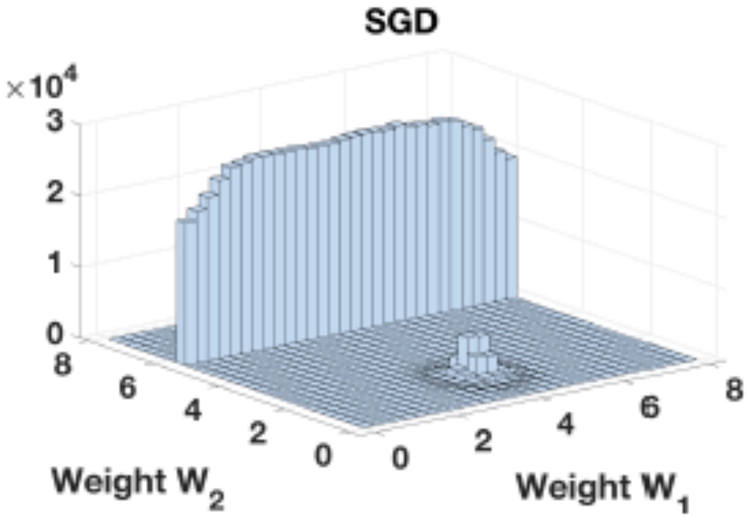
C-BRIC
Center for Brain Inspired Computing

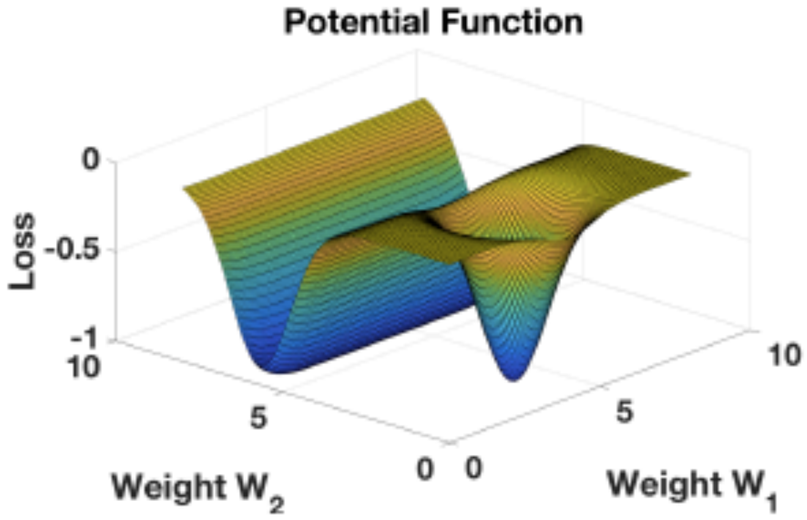
SGDL and SGD: intuition summary

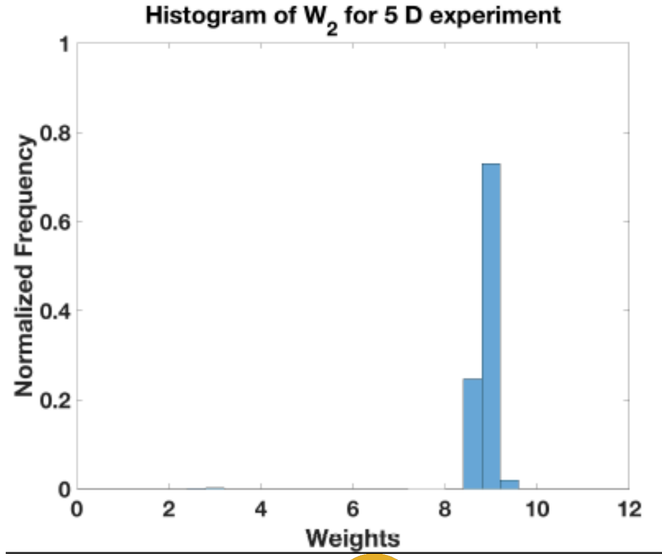
- Over-parametrized deep networks have many global minimizers that are generically degenerate; other critical points of the gradient are generically isolated.

- SGDL *concentrates with high probability* on large volume (degenerate) minima which are zero-minimizers.









C-BRIC
Center for Brain Inspired Computing

Today's plan

1. Loss landscape ✓
2. Intuition about SGD convergence to global, degenerate minima ✓
3. Dynamics of Optimization/training under the square loss
4. Gradient Descent and SGD
5. Low-rank bias and SGD noise
6. Neural Collapse
7. Architectures
8. Appendix

Deep Networks

We assume a deep network with L layers with the usual coordinate-wise scalar activation functions $\sigma(z): \quad \mathbf{R} \rightarrow \mathbf{R}$ as the set of functions

$$g(W;x) = (W_L \sigma(W_{L-1} \cdots \sigma(W_1 x)))$$

where the input is $x \in \mathbf{R}^d$, the weights are given by the matrices W_k, one per layer, with matching dimensions. No bias terms: the bias is instantiated in the input layer by one of the input dimensions being a constant. The activation nonlinearity is a ReLU, given by $\sigma(x) = x_+ = max(0,x)$.

Assumptions in this model

Simplifying assumptions in the first stage of analysis:

- Scalar output of the network $f(x)$; binary classification to measure error;
- Gradient Flow (learning rate is infinitesimally small), not discrete Gradient Descent nor SGD;
- Normalization of the weights is modeled with a Lagrange multiplier; this is equivalent to Weight Normalization (with is similar but not identical to Batch Normalization).
- Overparametrization: zero-loss (interpolation) is achieved.

Weight normalization: A simple reparameterization to accelerate training of deep neural networks
Tim Salimans and Diederik P. Kingm in Advances in Neural Information Processing Systems}, 2016

Square loss classification, overparametrization

- Empirically square loss works well for classification (L. Hui and M. Belkin, 2020)

$$\mathscr{L} = \sum_{n}^{L} (g(x_n) - y_n)^2, \quad y_n = \pm 1$$

- Without regularization there are global minima with zero square loss, thus corresponding to interpolating solutions (in general degenerate). All the interpolating solutions are optimal solutions of the regression problem, corresponding to different margins and to different expected classification performance.

- In other words, zero square loss does not imply by itself neither large margin nor good classification on a test set. When can we expect the solutions of the regression problem to have a large margin and good classification?

Loss function

Consider the loss function

$$\mathscr{L} = \sum_{n}^{L} (g(x_n) - y_n)^2 + \sum_{k=1}^{L-1} \nu_k(||W_k||^2 - 1) + \lambda \sum_{k=1}^{L} ||W_k||^2$$

under the constraint $||W_k||^2 = 1.$

Gradient flow in this model is

$$\dot{W}_k = 2 \sum_{n} (1 - \overline{g}_n)(\frac{\partial \overline{g}_n}{\partial W_k} - W_k \overline{g}_n)$$

$$\dot{W}_L = 2 \sum_{n} (1 - \overline{g}_n) \frac{\partial \overline{g}_n}{\partial W_L} - 2\lambda W_L \quad \text{with } \overline{g}_n = y_n g(x_n) \text{ where } y_n = \pm 1$$

Reparametrization

We reparametrize $f_W(x)$ in terms of normalized weight

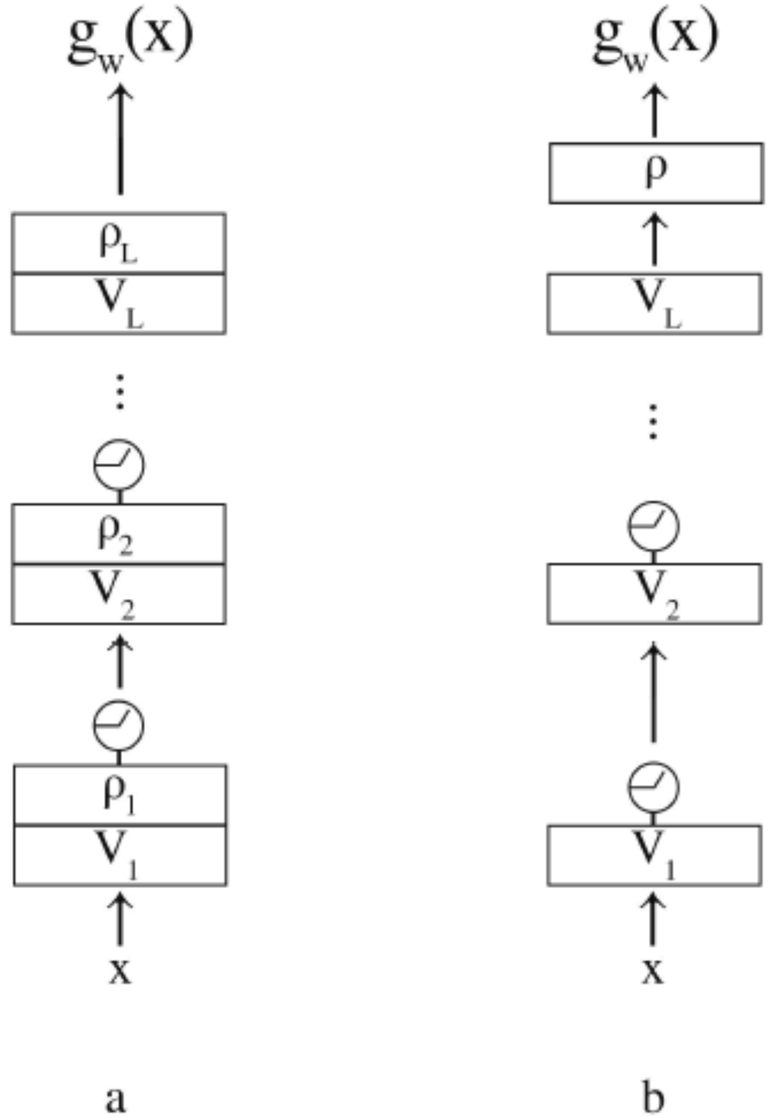
matrices $V_k = \dfrac{W_k}{\|W_k\|}$

defining $\rho_k = \|W_k\|$ and obtaining

$$f_W(x) = \rho_L V_L \sigma\left(\rho_{L-1} \ldots \sigma\left(\rho_1 V_1 x\right) \ldots\right) = \rho f_V(x).$$

Because of homogeneity of the ReLU it is possible to pull out the product of the layer norms as $\rho = \prod_k \rho_k$. Notice

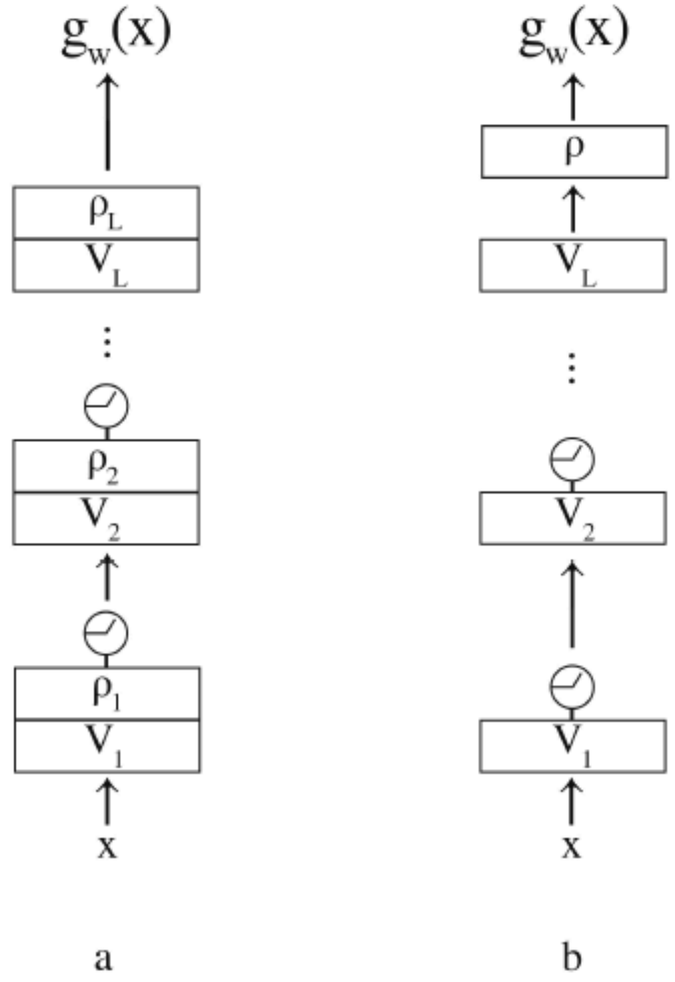
that the two networks -- $g_W(x)$ and $\rho f_V(x)$ -- are equivalent reparameterizations of the same function. We define $f_n := f_V(x_n)$.



Gradient flow dynamics

$$\dot{\rho} = -2[\sum_{S} \rho(f_n)^2 - \sum_{S} f_n y_n] - 2\lambda\rho$$

$$\dot{V}_k = 2\rho \sum_{S} [(\rho f_n - y_n)(V_k f_n - \frac{\partial f_n}{\partial V_k})]$$



$$f(x) = (V_L\sigma(V_{L-1}\cdots\sigma(V_1 x))) = V_L D_{L-1}(x)V_{L-1}\cdots D_1(x)V_1 x$$

$g(x) = \rho f(x)$ ρ is the product of the Frobenius norms of the weight matrices

Assume regularization

We know that

$$\mathscr{L} = \sum_{n}^{N} (g(x_n) - y_n)^2 + \sum_{k=1}^{L} \lambda_k ||\rho_k||^2$$

implies isolated minima but we know that

$$\mathscr{L} = \sum_{n}^{N} (g(x_n) - y_n)^2 + \lambda \sum_{k=1}^{L} ||\rho_k||^2$$

may not (it does not in linear case , may in the nonlinear case). Is this also true for

$$\mathscr{L} = \sum_{n}^{N} (g(x_n) - y_n)^2 + \lambda \Pi_{k=1}^{L} ||\rho_k||^2$$

?

How to derive it

$$\dot{\rho} = -\frac{\partial \mathscr{L}}{\partial \rho} = \frac{2}{N}\sum_n (1-\rho\bar{f}_n)\bar{f}_n - 2\lambda\rho$$

$$\dot{V}_k = -\frac{\partial L}{\partial V_k} = \frac{2}{N}\sum_n (1-\rho\bar{f}_n)\rho\frac{\partial \bar{f}_n}{\partial V_k} - 2\nu_k V_k\,.$$

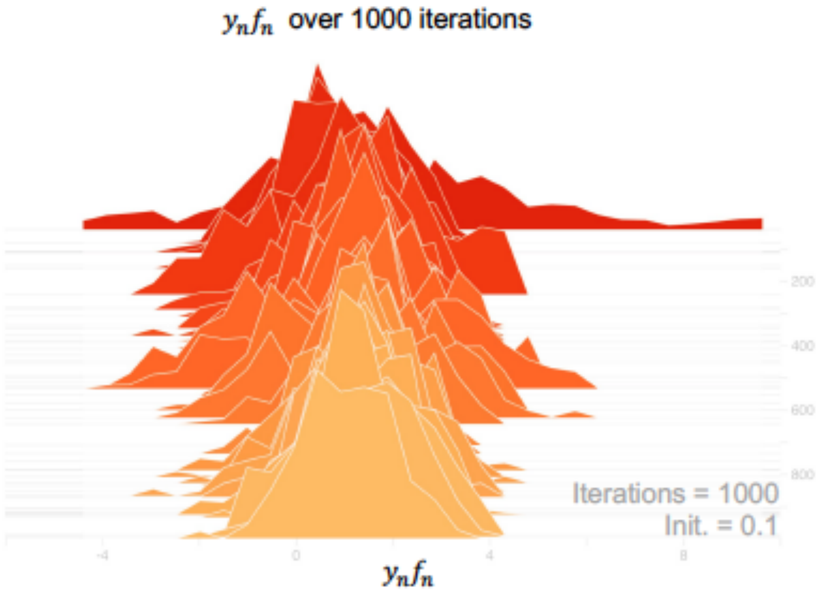
and since $V_k^T\dot{V}_k = 0$, using the property $\sum_{i,j}\frac{\partial f}{\partial V_k^{i,j}}V_k^{i,j} = V_k^T\frac{\partial f(x)}{\partial V_k} = f(x)$

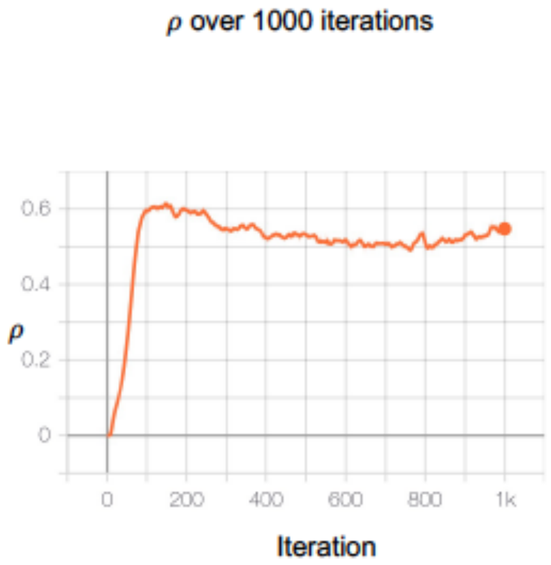
$$\nu_k = \frac{1}{N}\sum_n (\rho\bar{f}_n - \rho^2 f_n^2) = \frac{1}{N}\sum_n \rho\bar{f}_n(1-\rho f_n)$$

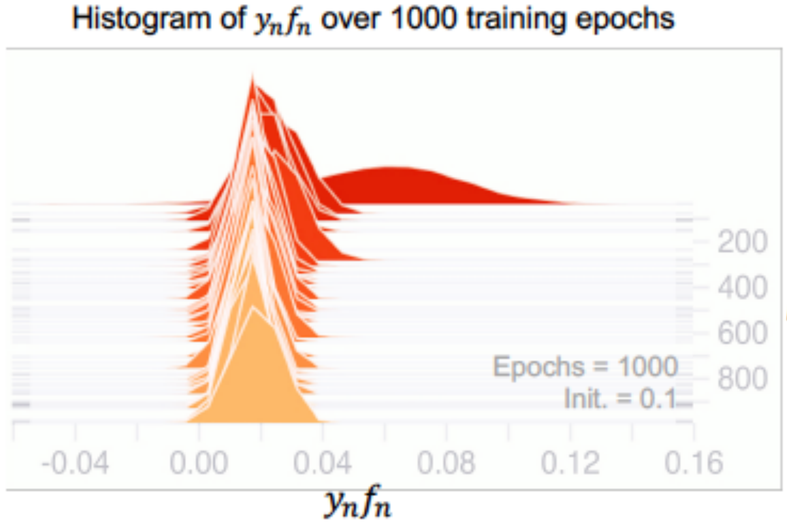
which, once substituted, gives the previous equations

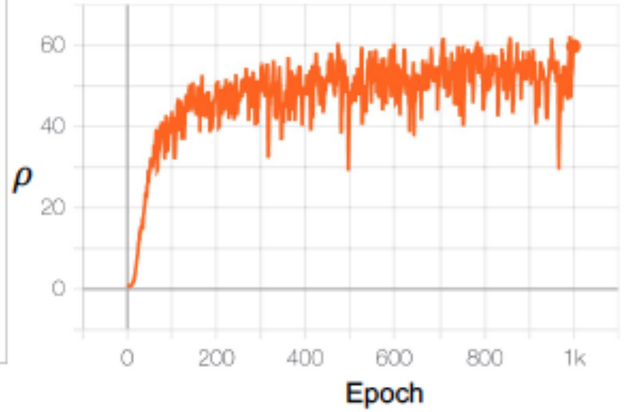
$$\dot{\rho} = -2[\sum_n \rho(f_n)^2 - \sum_n f_n y_n] - 2\lambda\rho \qquad \dot{V}_k = 2\rho\sum_n [(\rho f_n - y_n)(V_k f_n - \frac{\partial f_n}{\partial V_k})]$$

Margins converging to average margin ($\sigma \to 0$)









More in Appendix

Today's plan

1. Loss landscape ✓
2. Intuition about SGD convergence to global, degenerate minima ✓
3. Dynamics of Optimization/training under the square loss ✓
4. Gradient Descent and SGD
5. Low-rank bias and SGD noise
6. Neural Collapse
7. Architectures
8. Appendix

GD and SGD (minibatch size 1) for interpolating regressors

We distinguish two sets of solution to $\nabla L = \nabla \sum_n \ell_n^2 = 0$ with $\ell_n = g(x_n) - y_n$:

- solutions of $\nabla \ell_i = 0, \forall i$
- solutions of $\sum_n (y_n - g(x_n)) \nabla g(x_n) = 0$ that are not solutions of $\nabla \ell_i = 0, \forall i$

These two sets of solutions are critical points for GD; only the first ones are critical points for SGD. SGD is less likely to get stuck in critical points that are not global minima than GD. Furthermore, at global minima SGD converges to solutions that satisfy

$$g(x_i) = y_i \quad and \quad \nabla \ell_i = 0 \quad \forall i$$

See appendix for details on SGD

Today's plan

1. Loss landscape ✓
2. Intuition about SGD convergence to global, degenerate minima ✓
3. Dynamics of Optimization/training under the square loss ✓
4. Gradient Descent and SGD ✓
5. Low-rank bias and SGD noise
6. Neural Collapse
7. Architectures
8. Appendix

SGD: intrinsic noise and low rank

At convergence (zero square loss) with SGD mini batch of size 1

$$0 = (I - V_k V_k^T)\mathscr{C}(x_n)\frac{\partial f(x_n)}{\partial V_k}, \quad \forall n$$

which implies

$$\frac{\partial f(x_n)}{\partial V_k} = V_k f(x_n), \quad \forall n\,.$$

$$\dot{\rho} = -2[\sum_n \rho(f_n)^2 - \sum_n f_n y_n] - 2\lambda\rho$$

$$\dot{V}_k = 2\rho \sum_n [(\rho f_n - y_n)(V_k f_n - \frac{\partial f_n}{\partial V_k})]$$

Low-rank bias and SGD noise

Lemma *At SGD "convergence" defined as* $\dot{\rho} = 0, \quad \dot{V}_k = 0, \quad \forall S$ *(where* S *is a minibatch of size 1 and* $\lambda > 0$*), the matrices* V_k *should have rank 1 (if the the network* $g(x)$ *has scalar output) .*

The lemma follows assuming that $\dot{V}_k = 0$ for all minibatches at convergence. Since

$$\dot{V}_k = 2\rho \sum_n [(\rho f_n - y_n)(V_k f_n - \frac{\partial f_n}{\partial V_k})]$$ and $(\rho f_n - y_n) \neq 0$ then

$$V_k f_n = \frac{\partial f_n}{\partial V_k}, \quad \forall k = 1, \cdots, L, \quad \forall n$$

Constraint equations

To see this consider

$$f(x) = V_L\sigma\left(V_{L-1}\cdots\sigma\left(V_1x\right)\right) = f\left(x_j\right) = V_L D_{L-1}\left(x_j\right) V_{L-1}\cdots\cdots V_{k+1}D_k\left(x_j\right) V_k\cdots D_1\left(x_j\right) V_1x_j$$

Call $V_L D_{L-1}(x)V_{L-1}\cdots V_{k+1}D_k(x) = a^T$ and $D_{k-1}(x)V_{k-1}D_{k-2}(x)\cdots D_1(x)V_1x = b$.

Then $f(x) = a^T V_k b$ and

$\frac{\partial f}{\partial V_k} = ab^T$. Then $V_k f = \frac{\partial f_n}{\partial V_k}$ yields $V_k f = ab^T$ that is V_k is a rank one matrix.

Furthermore

$V_k f = \left[V_L D_{L-1}(x)V_{L-1}\cdots V_{k+1}D_k(x)\right]^T D_{k-1}(x)V_{k-1}D_{k-2}(x)\cdots D_1(x)V_1x$ that is

Convergence across layers: here is intertwining of SGD noise, rank minimization, minimum ρ and generalization, Neural Collapse,

$$\dot{\rho} = -\frac{2}{B}[\sum_{n\in B} \rho(f_n)^2 - \sum_{n\in B} f_n y_n] - 2\lambda\rho$$

$$\dot{V}_k = \frac{2\rho}{B}\sum_{n\in B}[(\rho f_n - y_n)(V_k f_n - \frac{\partial f_n}{\partial V_k})]$$

- Initially there is large error $(\rho f_n - y_n)$ driving V_k to change; at the same time there is rank reduction by SGD (rank reduction is limited by approximation error becoming bad if rank is too small).

- At convergence at the top layer there is quasi-interpolation, that is $(\rho f_n - y_n) \leq \epsilon > 0$ implying in all layers $\|\dot{V}_k\| \leq \frac{2\rho\epsilon}{B^2}\|\sum_{n\in B}(V_k f_n - \frac{\partial f_n}{\partial V_k})\|$;

- Assume that at some layer $k = k^*$, the term $\|\sum_{n\in B}(V_k f_n - \frac{\partial f_n}{\partial V_k})\|$ is small. Then $\|\dot{V}_k\|$ is close to zero, V_{k^*} has small rank (less than B) *but not too small (otherwise error* $(\rho f_n - y_n)$ *increases too much)*.

Predictions

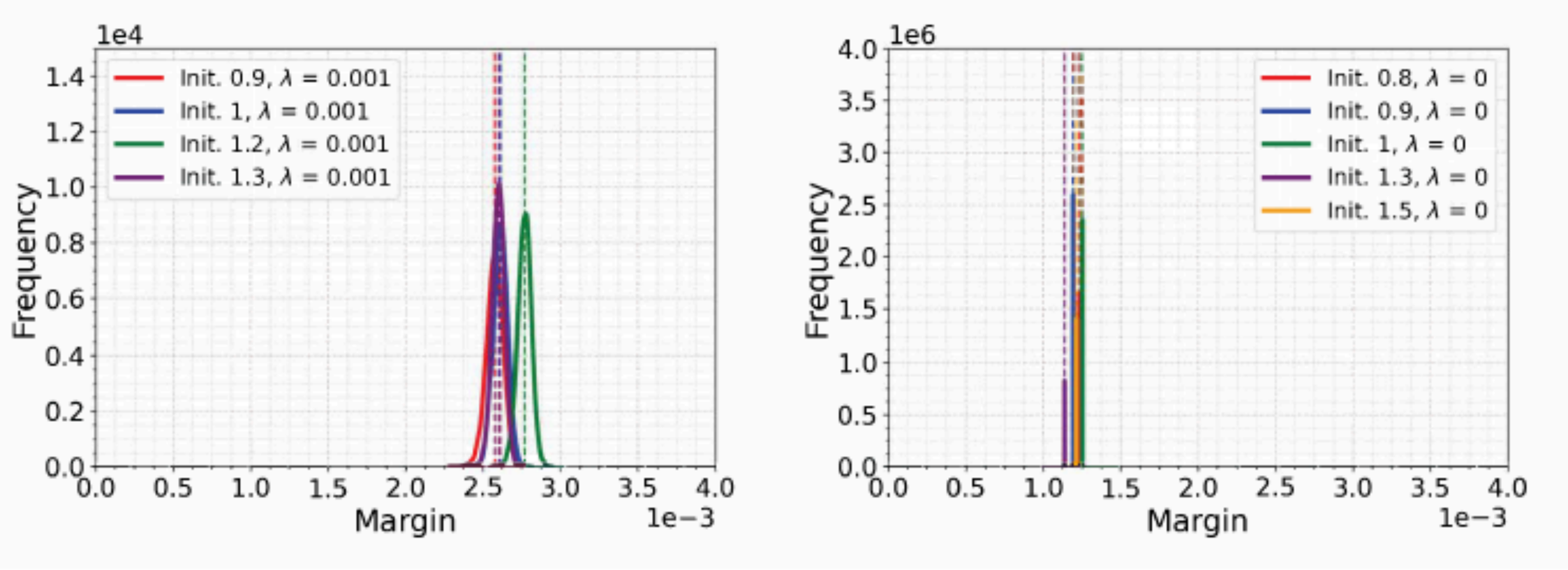
- Fluctuations in $\bar{f}_n$ during training should be minimal for $\lambda = 0$ -- just due to the finite learning rate of gradient descent -- and increase for increasing λ . Separately, μ (average margin) should increase with increasing λ because $\mu = (M + \lambda)\rho$.

- According to our equations there should be no SGD specific noise (that is $\|\dot{V}_k\| \approx 0$) when the minibatch size is equal to the training dataset size -- that when SGD becomes GD -- and no dependence of these fluctuations on λ.

- According to our equations the size of $\dot{V}_k$ depends on λ, because λ ensures $(1 - \rho f_n) > 0$. It should be minimal at the top layer, assuming that the top layer is close to converging to Neural Collapse, since the rank of the top layer is small.

- In the case of exponential-type loss functions such as the logistic loss, the presence of the SGD-specific noise is expected even when $\lambda = 0$, because of $\dot{W}_k^{i,j} = \sum_n^N e^{-y_n f(x_n)} \frac{\partial f}{\partial W_k^{i,j}}$

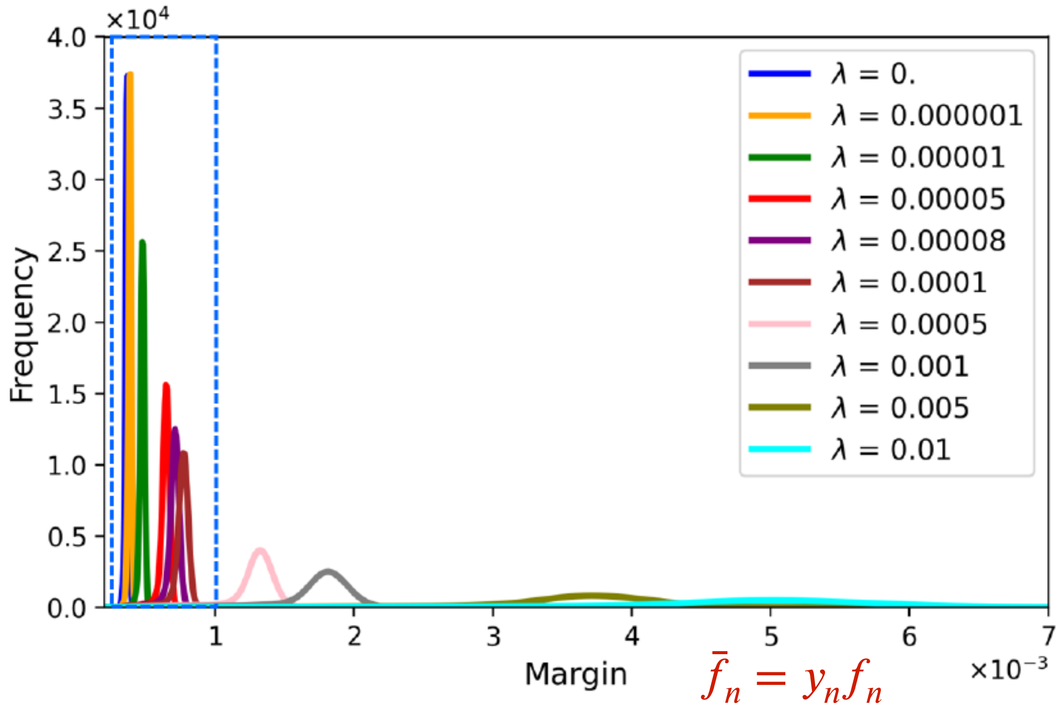
SGD noise


1e4
Frequency
Margin
1e−3
Init. 0.9, λ = 0.001
Init. 1, λ = 0.001
Init. 1.2, λ = 0.001
Init. 1.3, λ = 0.001
1e6
Frequency
Margin
1e−3
Init. 0.8, λ = 0
Init. 0.9, λ = 0
Init. 1, λ = 0
Init. 1.3, λ = 0
Init. 1.5, λ = 0

SGD noise in Margins

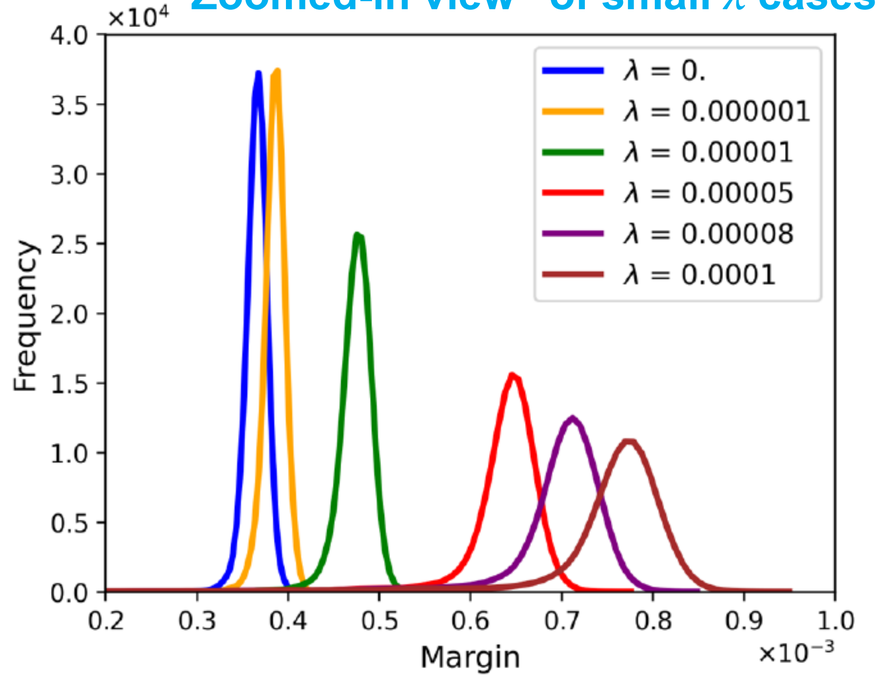
- Margin distributions over all training data trained with *SGD, square loss, different weight decay parameters (λ)*

Six-layer model (4 Conv + 2FC), CIFAR10, $N = 10K,\ B = 128,\ \eta = 0.03.$



$\bar{f}_n = y_n f_n$

"Zoomed-in view" of small λ cases



❖ As predicted, the variance of the fluctuations is small with $\lambda = 0$ and grows with increasing λ.

Xu M., Galanti, T. and Poggio, T., Intrinsic fluctuations of SGD, *submitted to NeurIPS*, May 2023

Predictions

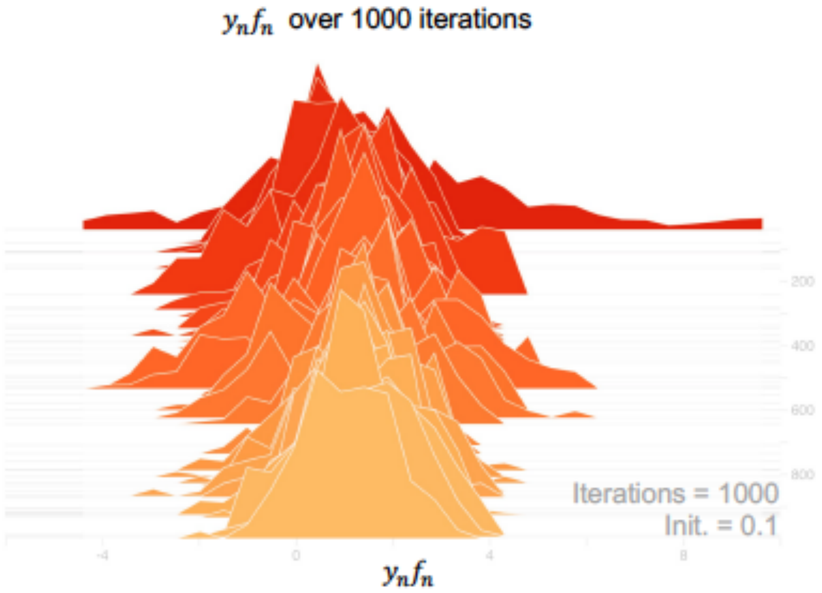
- Fluctuations in $\bar{f}_n$ during training should be minimal for $\lambda = 0$ -- just due to the finite learning rate of gradient descent -- and increase for increasing λ . Separately, μ (average margin) should increase with increasing λ because $\mu = (M + \lambda)\rho$.

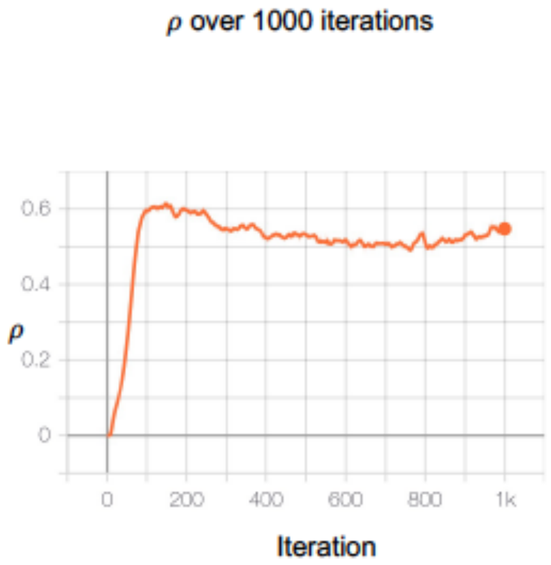
- According to our equations there should be no SGD specific noise (that is $\|\dot{V}_k\| \approx 0$) when the minibatch size is equal to the training dataset size -- that when SGD becomes GD -- and no dependence of these fluctuations on λ.

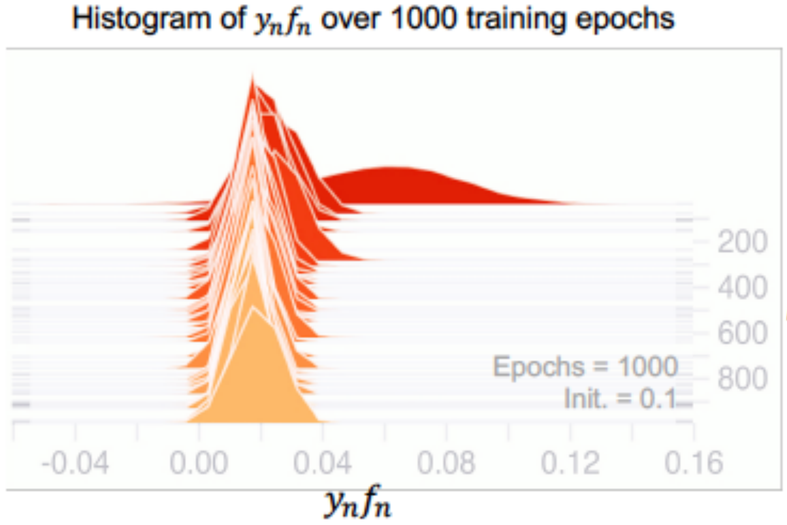
- According to our equations the size of $\dot{V}_k$ depends on λ, because λ ensures $(1 - \rho f_n) > 0$. It should be minimal at the top layer, assuming that the top layer is close to converging to Neural Collapse, since the rank of the top layer is small.

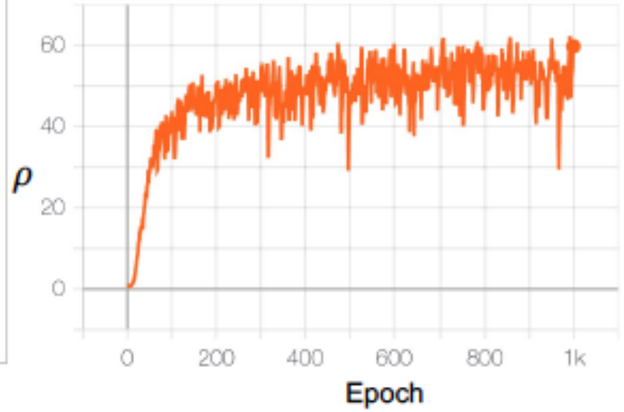
- In the case of exponential-type loss functions such as the logistic loss, the presence of the SGD-specific noise is expected even when $\lambda = 0$, because of $\dot{W}_k^{i,j} = \sum_n^N e^{-y_n f(x_n)} \frac{\partial f}{\partial W_k^{i,j}}$

Margins converging to average margin ($\sigma \to 0$)

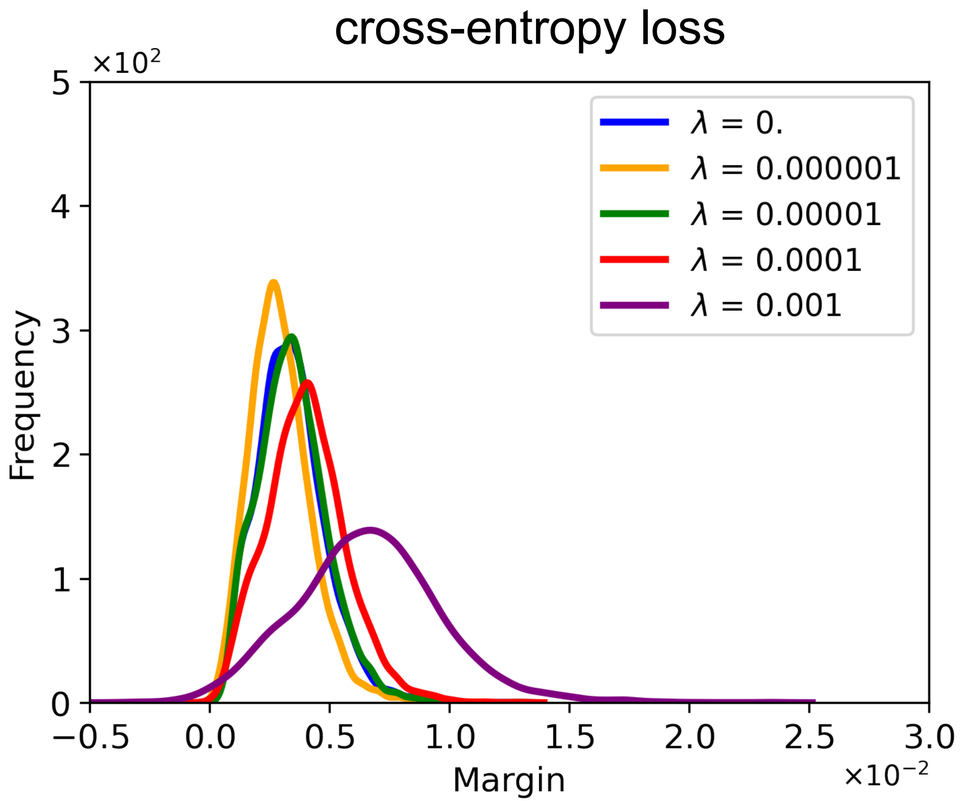


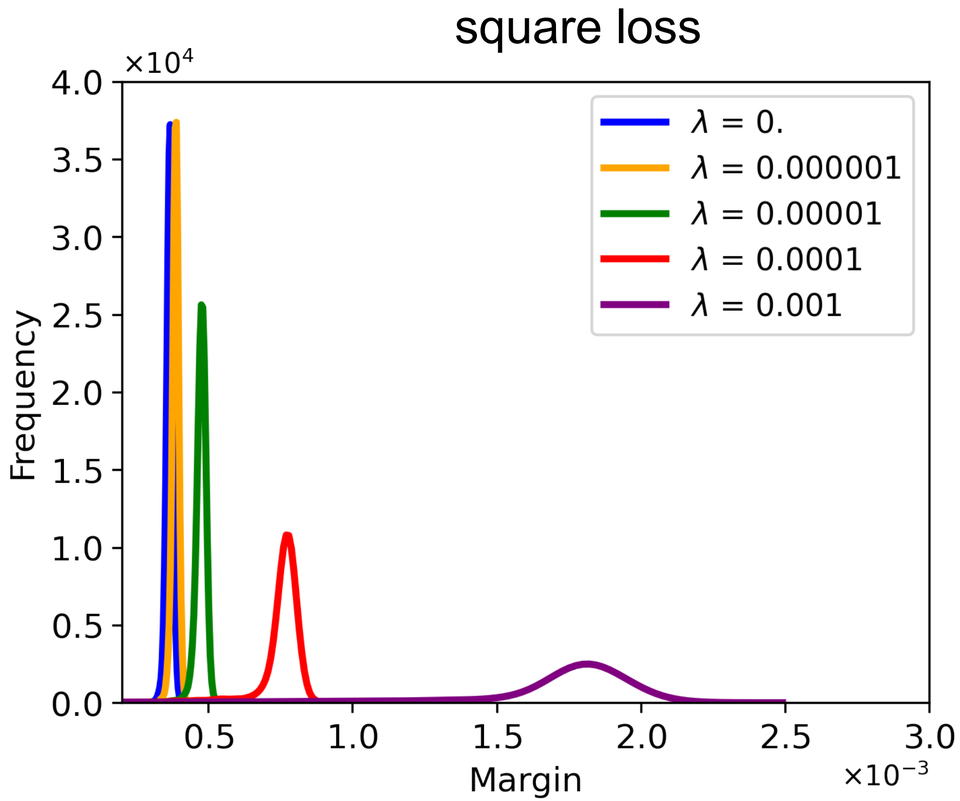






SGD noise in Margins (exponential loss vs. square loss)





❖ *With cross-entropy loss,* the presence of the SGD-specific noise is expected to be significant even when $\lambda = 0$, unlike the square loss case.

Xu M., Galanti, T. and Poggio, T., Intrinsic fluctuations of SGD, *submitted to NeurIPS*, May 2023

Predictions

- Fluctuations in $\bar{f}_n$ during training should be minimal for $\lambda = 0$ -- just due to the finite learning rate of gradient descent -- and increase for increasing λ . Separately, μ (average margin) should increase with increasing λ because $\mu = (M + \lambda)\rho$.

- According to our equations there should be no SGD specific noise (that is $\|\dot{V}_k\| \approx 0$) when the minibatch size is equal to the training dataset size -- that when SGD becomes GD -- and no dependence of these fluctuations on λ.

- According to our equations the size of $\dot{V}_k$ depends on λ, because λ ensures $(1 - \rho f_n) > 0$. It should be minimal at the top layer, assuming that the top layer is close to converging to Neural Collapse, since the rank of the top layer is small.

- In the case of exponential-type loss functions such as the logistic loss, the presence of the SGD-specific noise is expected even when $\lambda = 0$, because of $\dot{W}_k^{i,j} = \sum_n^N e^{-y_n f(x_n)} \frac{\partial f}{\partial W_k^{i,j}}$

Intrinsic fluctuations of SGD

All gradient descent methods try to converge to points in the parameter space that has zero or small gradient, *i.e.,* try to minimize $\| \dot{V}_k \|, \forall k \in [L]$ iteratively, over different minibatches (batch size is *B*).

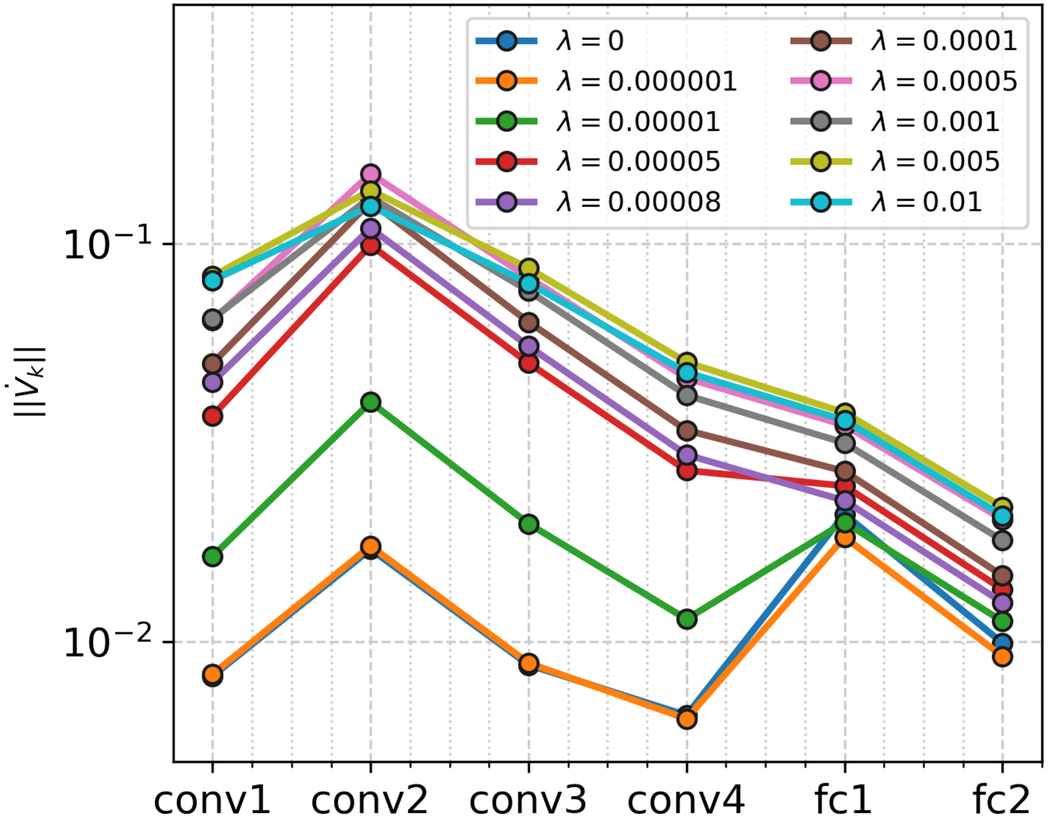
$$\dot{V}_k = \frac{2}{B}\rho \sum_{n \in B} \left[\underbrace{(1 - \rho \bar{f}_n)}_{\text{error}} \left(-V_k \bar{f}_n + \frac{\partial \bar{f}_n}{\partial V_k} \right) \right]$$

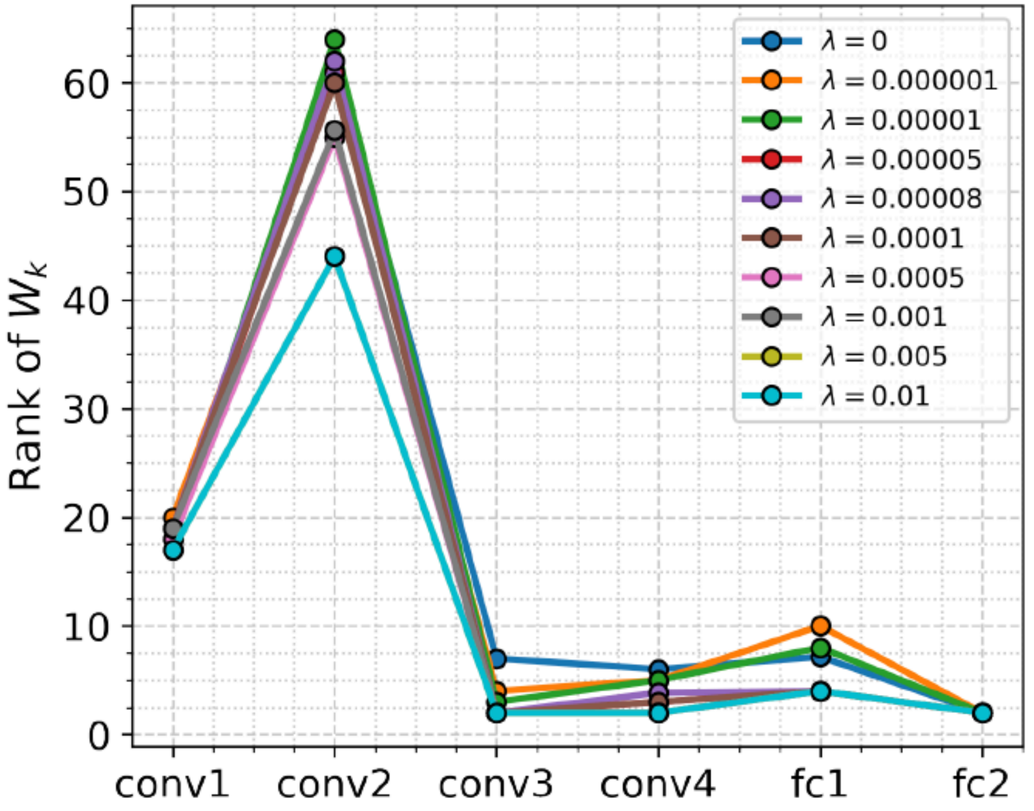
Then, $\|\dot{V}_k\| = \| \frac{2\rho}{B} \sum_{n \in B} \ell_n (\frac{\partial \bar{f}_n}{\partial V_k} - f_n V_k) \|$

Our results in the following slide suggest that the fluctuations are due to a specific SGD dynamics, arising from a tradeoff between *minimizing the error* and *minimizing the rank*. The dynamics can also be described as switching between slightly different V_k, each one trying to fit just one mini batch.

SGD noise in weight matrices (V_k)

❑ **Layer-wise $||\dot{V}_k||(\forall k \in [L])$ in logarithmic scale by 10 different λ at convergence**. The SGD noise across layers, as measured by $||\dot{V}_k||$, is small when λ is set to 0 or 1e-06. An increase of λ from 1e-05 to 0.01 generates larger fluctuations, especially for the first four convolution layers. The last two fully connected layers tend to consist of low-rank matrices, corresponding to smaller $||\dot{V}_k||$.





Today's plan

1. Loss landscape ✓
2. Intuition about SGD convergence to global, degenerate minima ✓
3. Dynamics of Optimization/training under the square loss ✓
4. Gradient Descent and SGD ✓
5. Low-rank bias and SGD noise ✓
6. Neural Collapse
7. Architectures
8. Appendix

Neural Collapse *(Papyan, Han, Donoho, PNAS, 2020)*

- (NC1) Cross-example within-class variability of last-layer training activations collapses to zero, eg. *all margins* $f(x_n)$ *converge to the same value within a very small error < 1%*

- (NC2) The class means collapse to the vertices of a simplex equiangular tight frame (ETF)

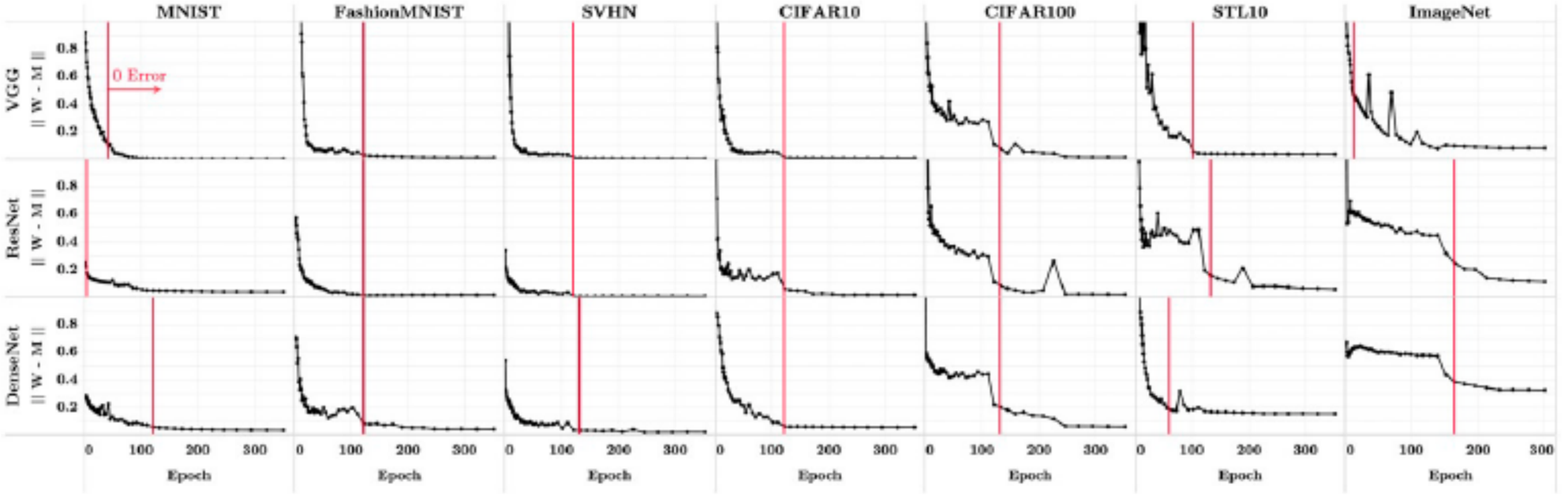
- (NC3) Up to rescaling, the last-layer classifiers collapse to the class means or in other words, to the simplex ETF (i.e., to a self-dual configuration), eg the *last layer activations are proportional to the classifier weights*

- (NC4) For a given activation, the classifier decision collapses to simply choosing whichever class has the closest train class mean (i.e., the nearest class center decision rule).

Prevalence of neural collapse during the terminal phase of deep learning training

PNAS, 2020

Vardan Papyan[a,1], X. Y. Han[b,1], and David L. Donoho[a,2]



Constraint equations for $D_k = I, \quad \forall k > k^*$

- The constraint equations will be satisfied if $V_k, k \geq k^*$ are projection matrices
- This implies $V_k, \quad \forall k > k^*$ show NC
- We can prove (because SGD with $B < N$ is "equivalent to SGD with mini batch 1; or Tomer which is the same) that for $k > k^*$ once there is NC there will be NC (and appropriate gates are open);

- This cannot go too far down layers because at some point the interpolation error will be too large (you need a few layers to approximate a compositional function)!

- There are a few PROJECTs here

Neural Collapse: other predictions/conjectures

- SGD is required for NC1 for crossentropy; for the square loss NC1 is expected not only for SGD but also for GD;

- Weight decay is needed to yield neural collapse (NC1 to NC4) for square loss but not for cross entropy;

- NC1 to NC4 should take place for any interpolating solutions (in the square loss case), including solutions that do not have the minimum ρ.

- This means that NC is not directly related to generalization.

Neural Collapse for binary classification is predicted from our analysis

The key observation is that $h_{i,c} = \frac{\partial f(x_{i,c})}{\partial V_L}$. Then since $V_k f_c = \frac{\partial f_c}{\partial V_k}$ at convergence it follows that $h^T(x_{i,c}) = V_L f_c$ at convergence.

- (NC1) Observe that $V_L f_c$ does not depend on i implying that the standard deviation of $h_{i,c}$ wrt i converges to zero;
- (NC2) in the binary case $h_{+1} \rightarrow V_L^T$ and $h_{-1} \rightarrow -V_L^T$. Thus $\mu_1 = V^T$ and $\mu_2 = -V^T$. This is a special case of NC2;
- (NC3) since $h(x_{i,c}) = V_L^T f_c$ and f_c is a number, at convergence h is proportional to V_L^T;
- (NC4) is implied by NC1 and NC2 (Papyan et al., 2020).

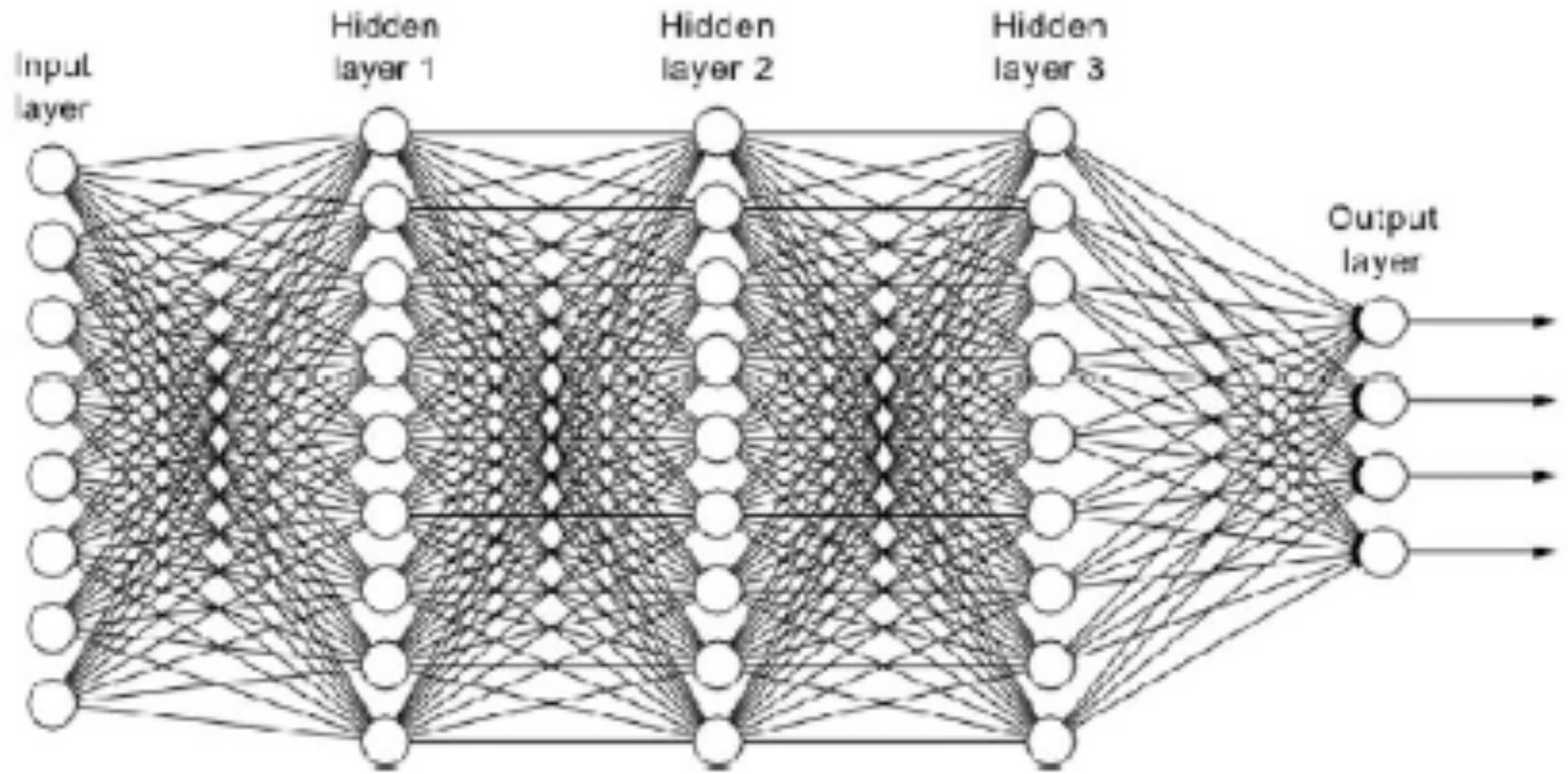
Today's plan

1. Loss landscape ✓
2. Intuition about SGD convergence to global, degenerate minima ✓
3. Dynamics of Optimization/training under the square loss ✓
4. Gradient Descent and SGD ✓
5. Low-rank bias and SGD noise ✓
6. Neural Collapse ✓
7. Architectures and prior bias on sparse decomposition
8. Appendix

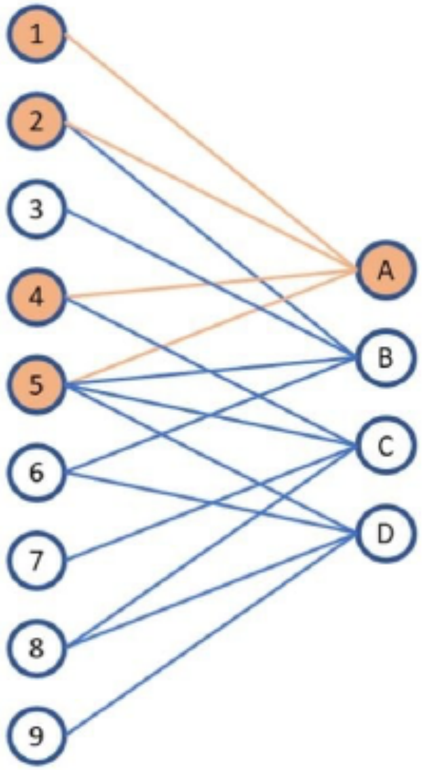
Neural network architectures:

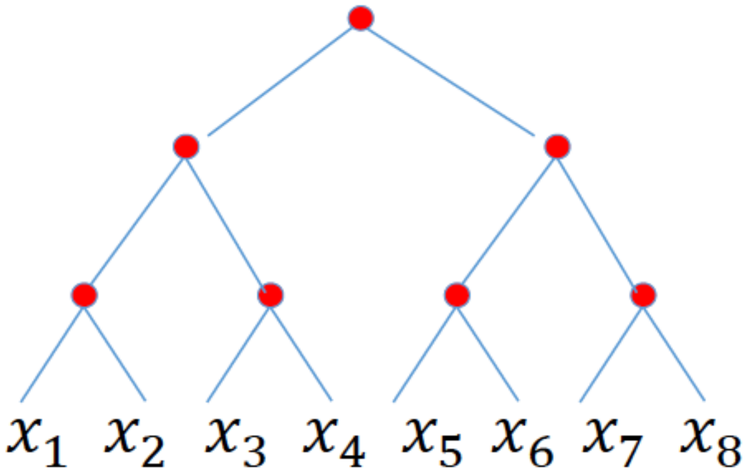
dense, convolutional and residual architectures

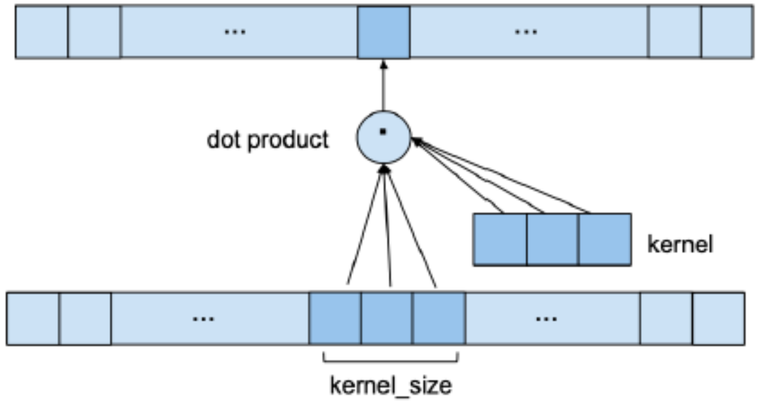
Dense architecture



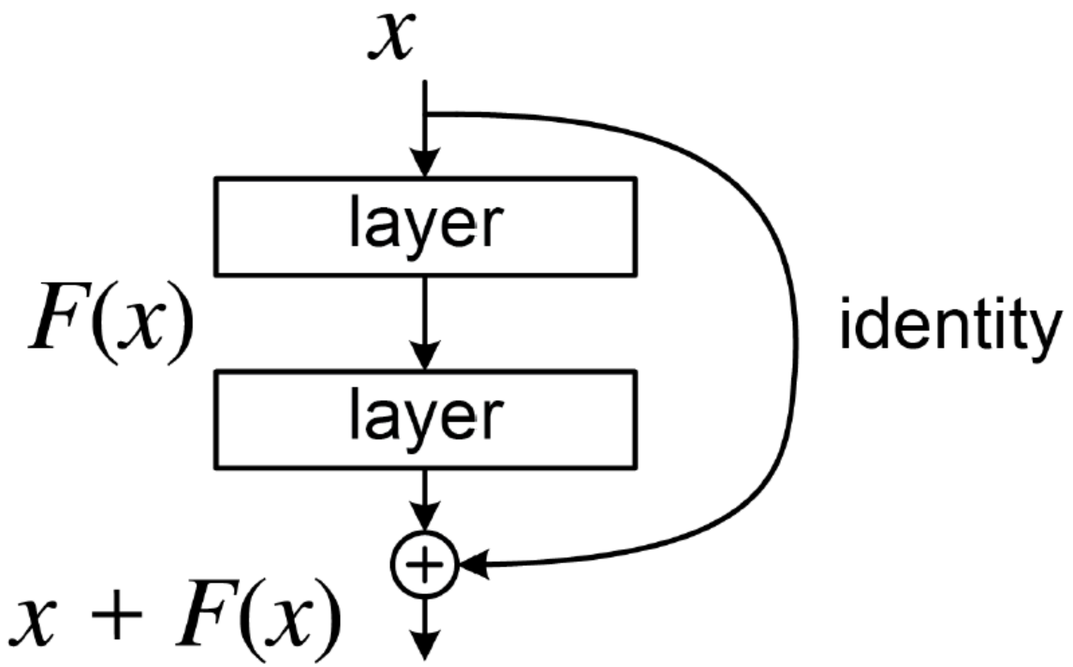
Sparse (“convolutional” is standard term) architecture







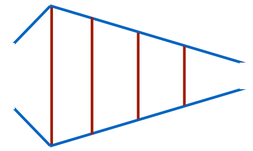
Residual architecture



Deep Residual Learning for Image Recognition

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

Wing architecture



For a class of monotonic activation functions including GeLU, neural networks having a pyramidal structure (no layer has more hidden units than the input dimension), produce necessarily connected decision regions. This implies that a sufficiently wide hidden layer (at least $> d + 1$) is necessary to guarantee that the network can produce disconnected decision regions.

Theorem(Nguyen et al., 2018) Let the width of the layers of the feedforward network network satisfy $d = n_0 \geq n_1 \geq \ldots \geq n_{L-1}$and let $\$\sigma_l : \mathbb{R} \rightarrow \mathbb{R}\$$ be a continuous, strictly monotonically increasing function with $\sigma_l(\mathbb{R}) = \mathbb{R}$ for every layer $1 \leq l \leq L - 1$ and all the weight matrices $\left(W_l\right)_{l=1}^{L-1}$have full rank. Then every decision region C_j is an open connected subset of $\mathbb{R}^d$ for every $1 \leq j \leq m$.

Remark:
Full rank of the weight matrices is necessary to get connected decision regions when the other conditions are satisfied.

Loss landscape

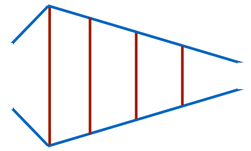
Theorem (Nguyen) Let the width of the layers of the feedforward network network satisfy $d = n_0 \geq n_1 \geq \ldots \geq n_{L-1}$ and let $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuous, strictly monotonically increasing function with $\sigma(\mathbb{R}) = \mathbb{R}$ for every layer $1 \leq l \leq L-1$ and all the weight matrices $\left(W_l\right)_{l=1}^{L-1}$ have full rank. Then every decision region C_j is an open connected subset of $\mathbb{R}^d$ for every $1 \leq j \leq m$.

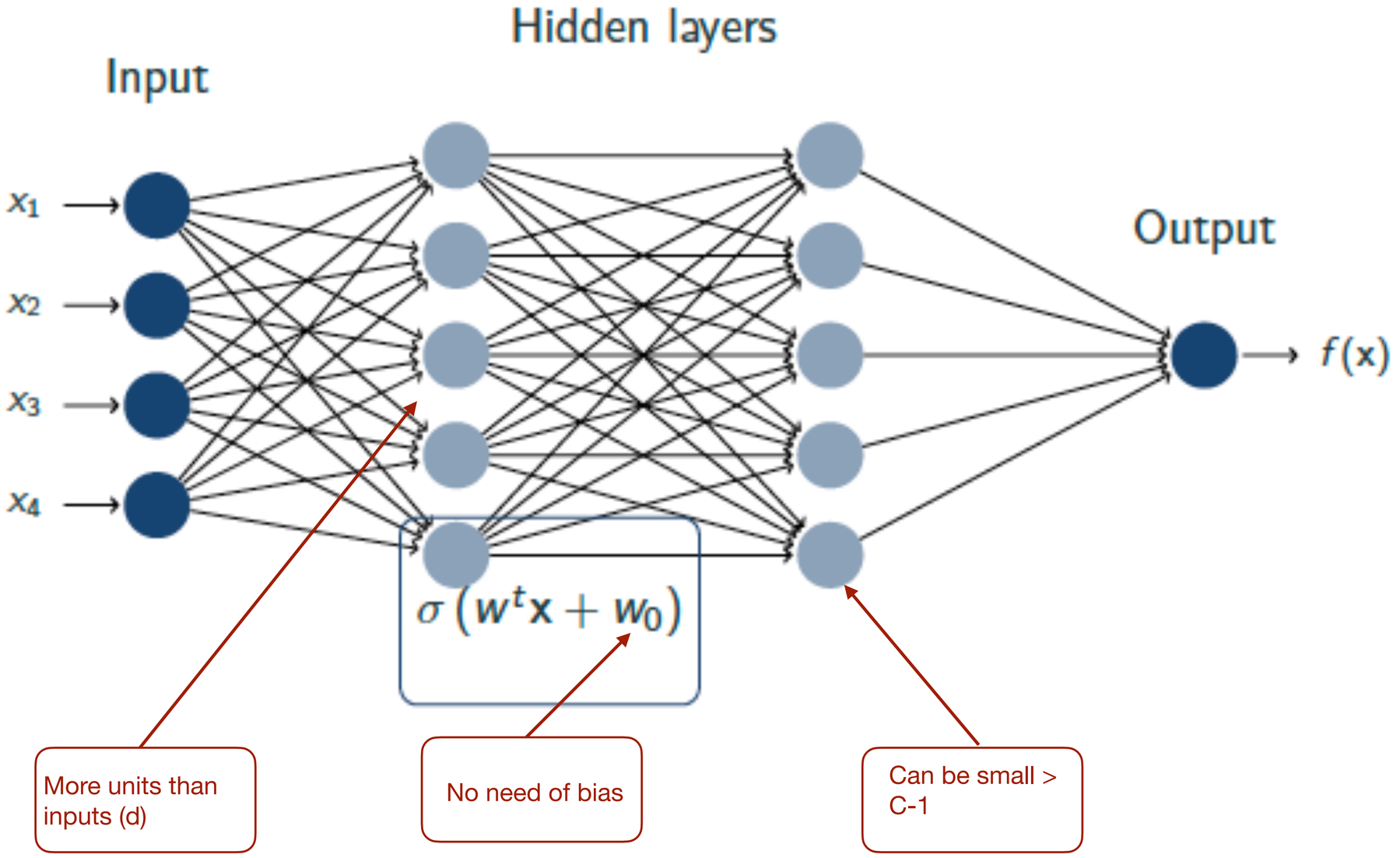
- A general guiding principle for the construction of hidden layers in neural networks one should use, at least for the first hidden layer, more units than the input dimension.

- If the weight matrices of the hidden layers are not full rank while the other conditions are still satisfied, then the decision regions can be disconnected.

Neural Networks Should Be Wide Enough to Learn Disconnected Decision Regions
Quynh Nguyen [1] Mahesh Chandra Mukkamala [1] Matthias Hein
Proceedings of the 35 th International Conference on Machine, Learning, Stockholm, Sweden, PMLR 80, 2018.

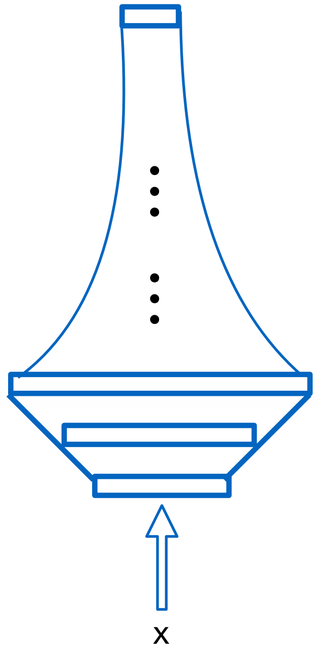
Wing architecture





CENTER FOR
Brains
Minds+
Machines

Wing architecture



Open question:
can GD learn the relevant hierarchical architecture?

We have shown that hierarchical networks can avoid the curse of dimensionality for functions with a compositional representation. This is exponentially better than shallow networks.

But…can a dense network learn a target function is hierarchical compositional? This is strictly outside approximation theory, involves learning/optimization

Let us look at learning sparsity in the linear case

$$W(t+1) - W(t) = -2\frac{\eta}{B}\left(W(t) - W^*\right)XX^T - 2\eta\lambda W(t)$$

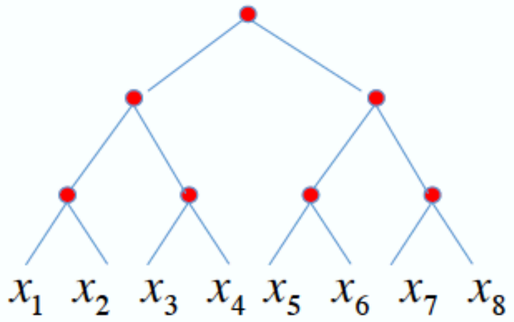
SGD vs GD: linear regression

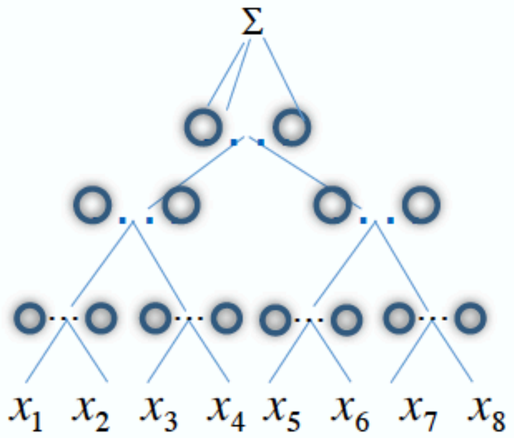
Theorem Consider the linear regression problem $WX = Y$. Assume that $W \in \mathbf{R}^{m,d}$ is found by SGD with minibatch of size B or by GD, both with learning rate η and regularization parameter λ. Assume overparametrization, that is the data matrix is $X \in \mathbf{R}^{d,N}$ with d > N. Let π be the projection on the span of the data (the columns of X), $W^{\|} = W\pi$ the weight matrix restricted to the data span, and $W^{\perp} = W(I - \pi)$ the weight matrix W restricted to the null space of X, so that $W = W^{\|} + W^{\perp}$. Then for each epoch assuming minibatches of size 1

- if W at initialization is the zero matrix both SGD and GD will converge to the same regularized solution;

- if W at initialization is a non-zero matrix, such as a random matrix, SGD and GD will also converge to the regularized solution. However, the convergence per epoch of $\$W^{\perp}\$$ (that is, components of $\mathscr{W}$ that are in the null space of XX^T) to zero is much faster for SGD than GD. The decrease is $(1 - \eta\lambda)^N$ for SGD and $(1 - \eta\lambda)$ for GD.

Can SGD learn
the relevant hierarchical architecture?





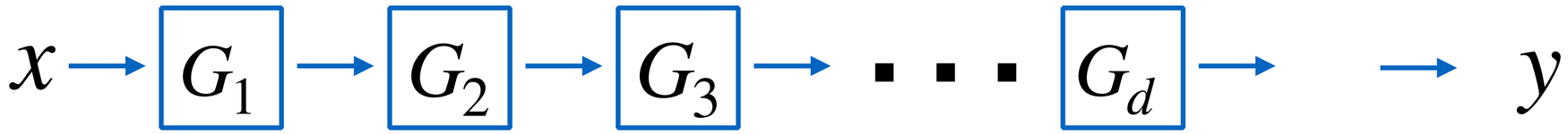
It turns out that in general a CNN architecture "cannot" be learned by a dense network from examples *at least for classification*.

A recent theorem (Malach and Shalev-Schwartz, 2021) shows a class of problems that can be efficiently solved using convolutional networks trained with gradient-descent; the class of problems is computationally hard to learn using a polynomial-size fully-connected network. Another one (Arora) shows results for sample complexity in similar situation.

```
Eran Malach and Shai Shalev{-Shwartz},
Computational Separation Between Convolutional and Fully-Connected
                Networks}  2020
```

Why Are Convolutional Nets More Sample-Efficient than Fully-Connected Nets?},
author={Zhiyuan Li and Yi Zhang and Sanjeev Arora},
year={2021}

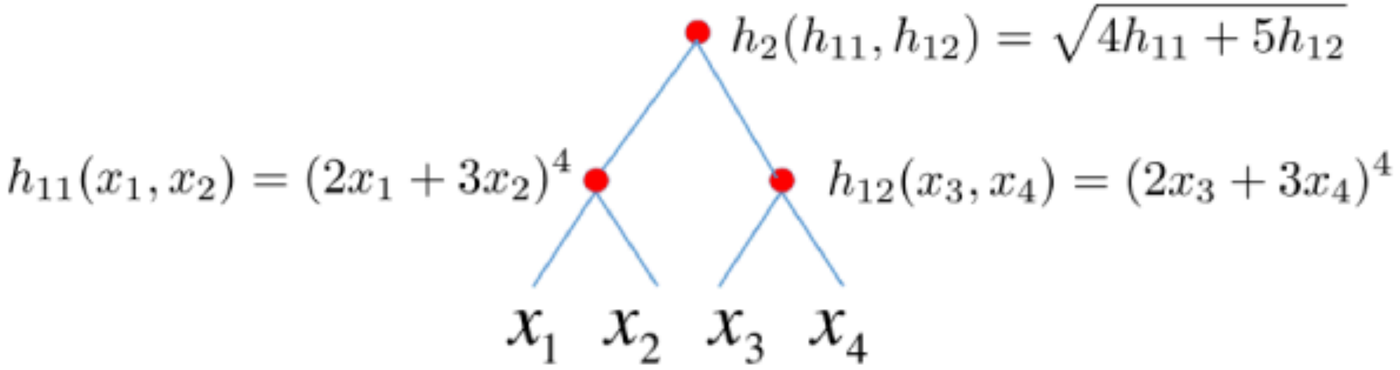
Theorems and Conjectures for overparametrization

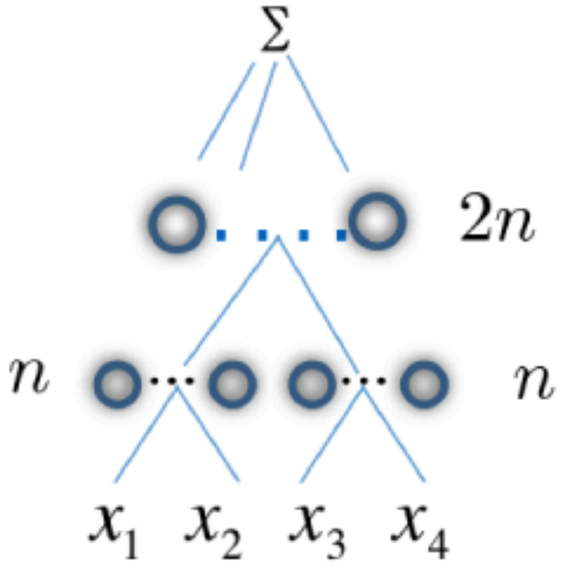


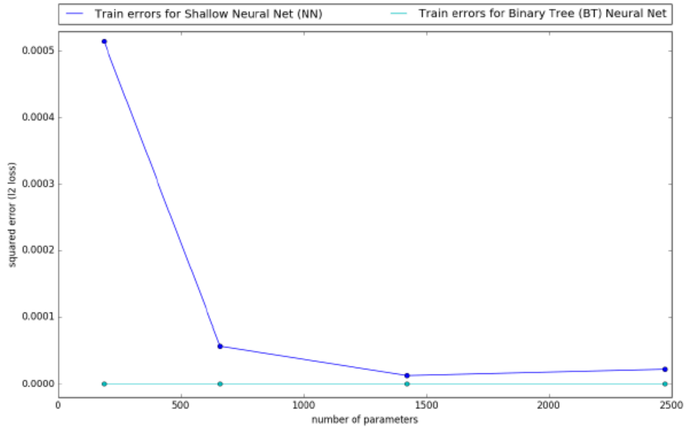
1. Compositionally sparse target functions can be learned by SGD if the graph of the function is known and the network reflects the graph.

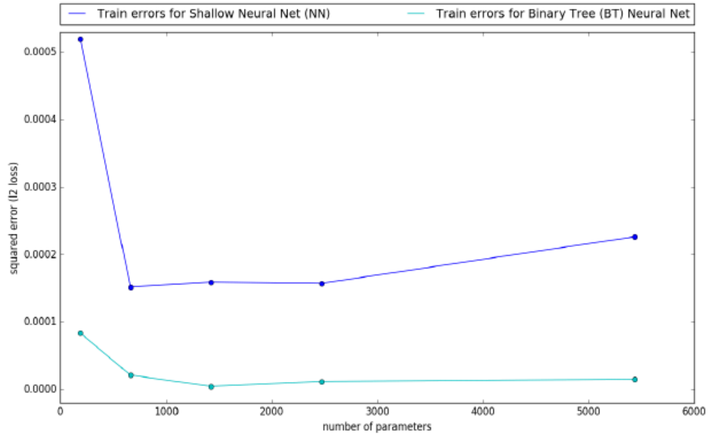
3. If the graph is unknown, sparse constraints may allow optimization to find sparse unknown graph. Pruning. Rather open question.

5. If the graph is unknown but the target function is staircase boolean (that is compositional function, sparse in x) then it can be learned by simple algorithm and empirically by CNN

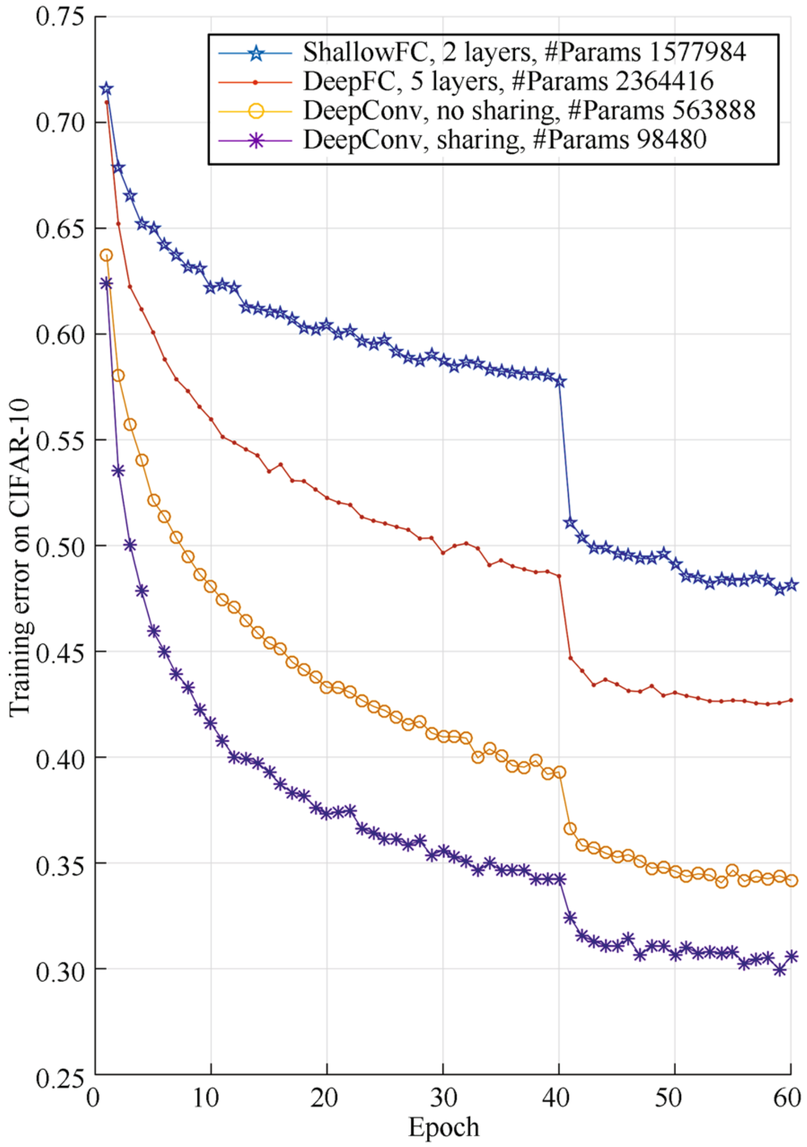


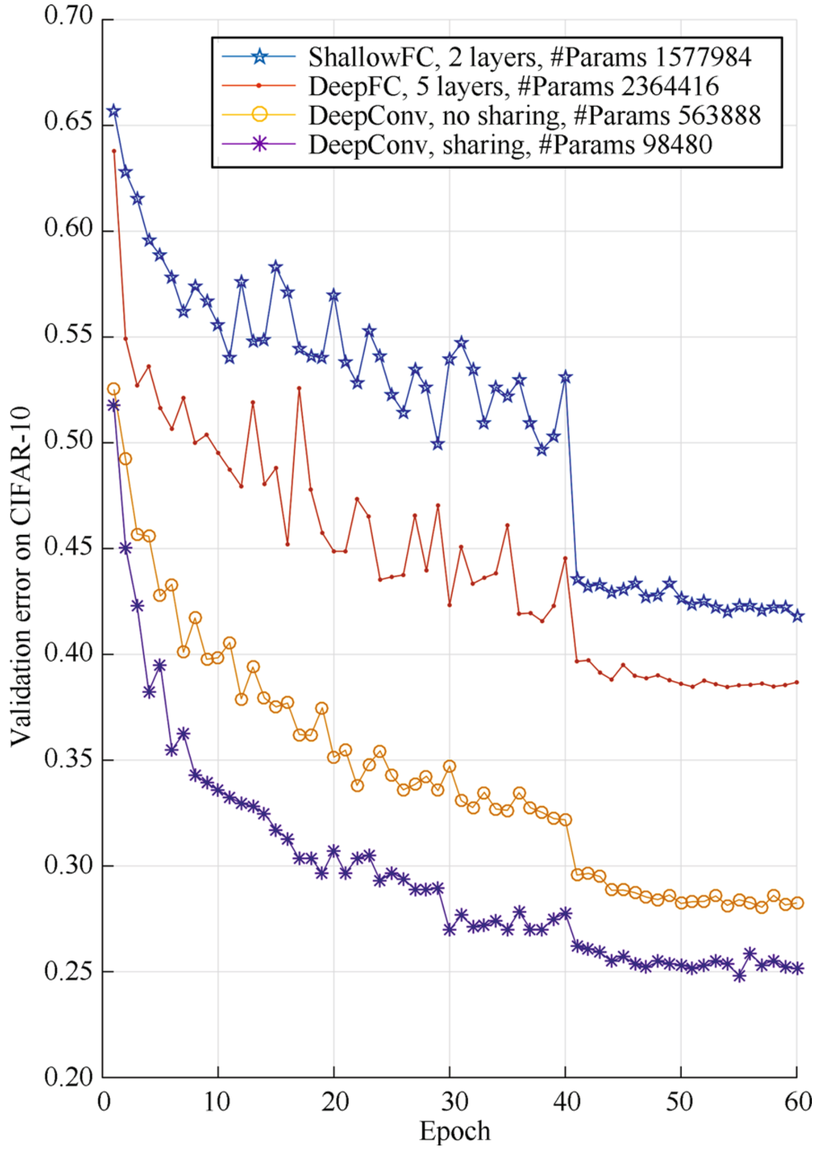






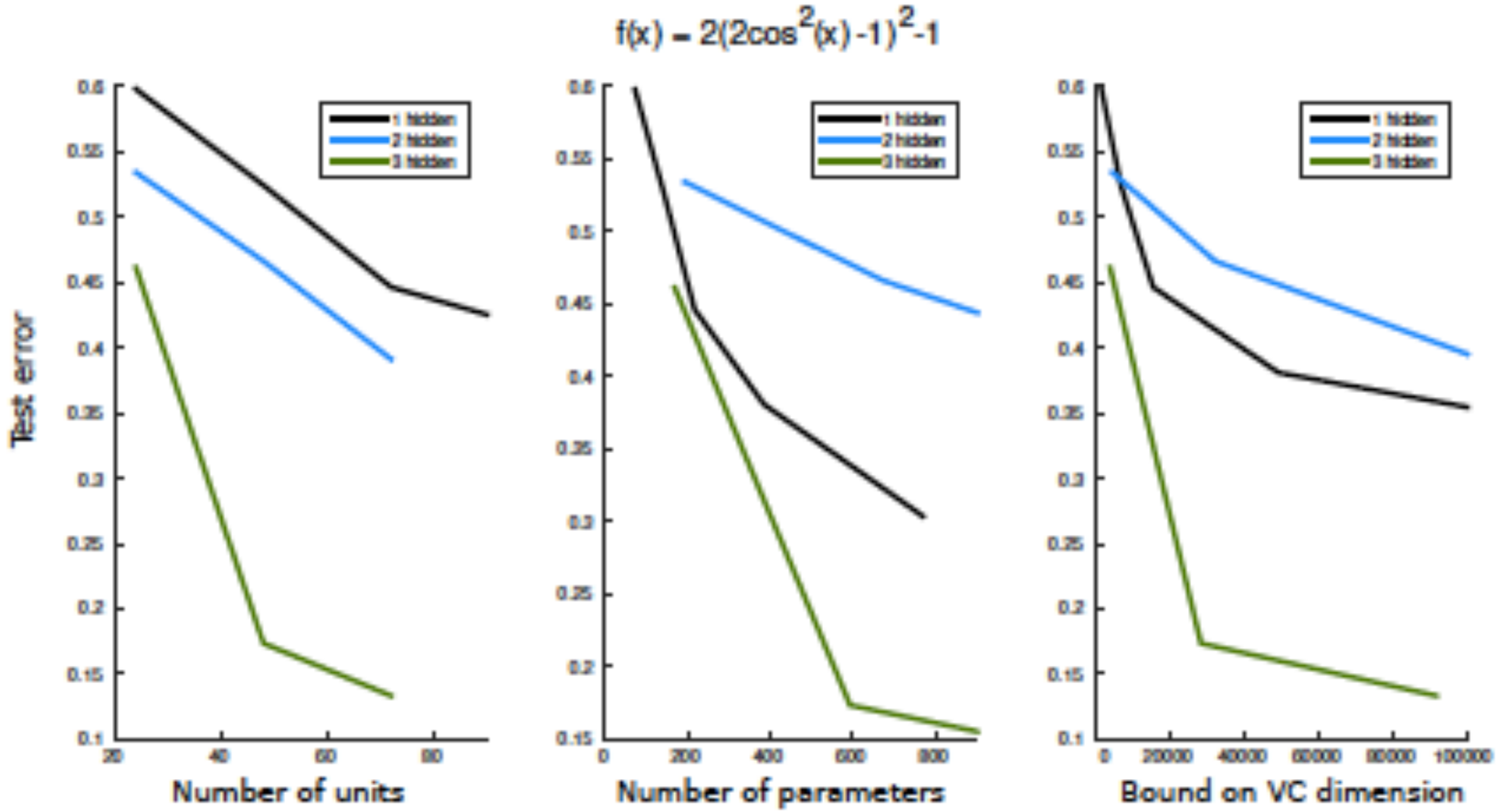
Poggio, T., Mhaskar, H., Rosasco, L. *et al.* Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *Int. J. Autom. Comput.* **14**, 503–519 (2017). https://doi.org/10.1007/s11633-017-1054-2





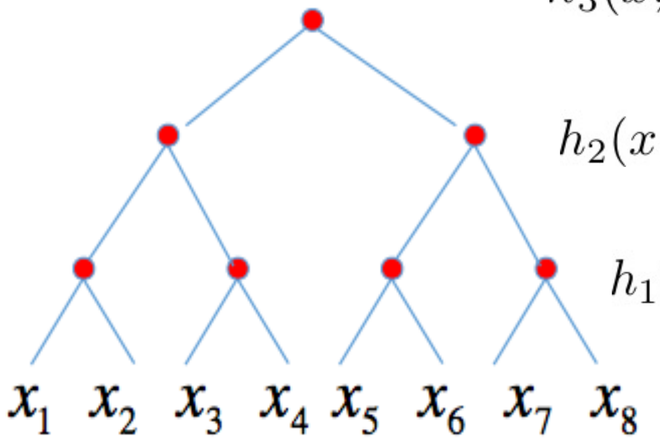
Poggio, T., Mhaskar, H., Rosasco, L. *et al.* Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *Int. J. Autom. Comput.* **14**, 503–519 (2017). https://doi.org/10.1007/s11633-017-1054-2

Sparse (compositional) functions



Mhaskar, Poggio, Liao, 2016

Function and Network architectures



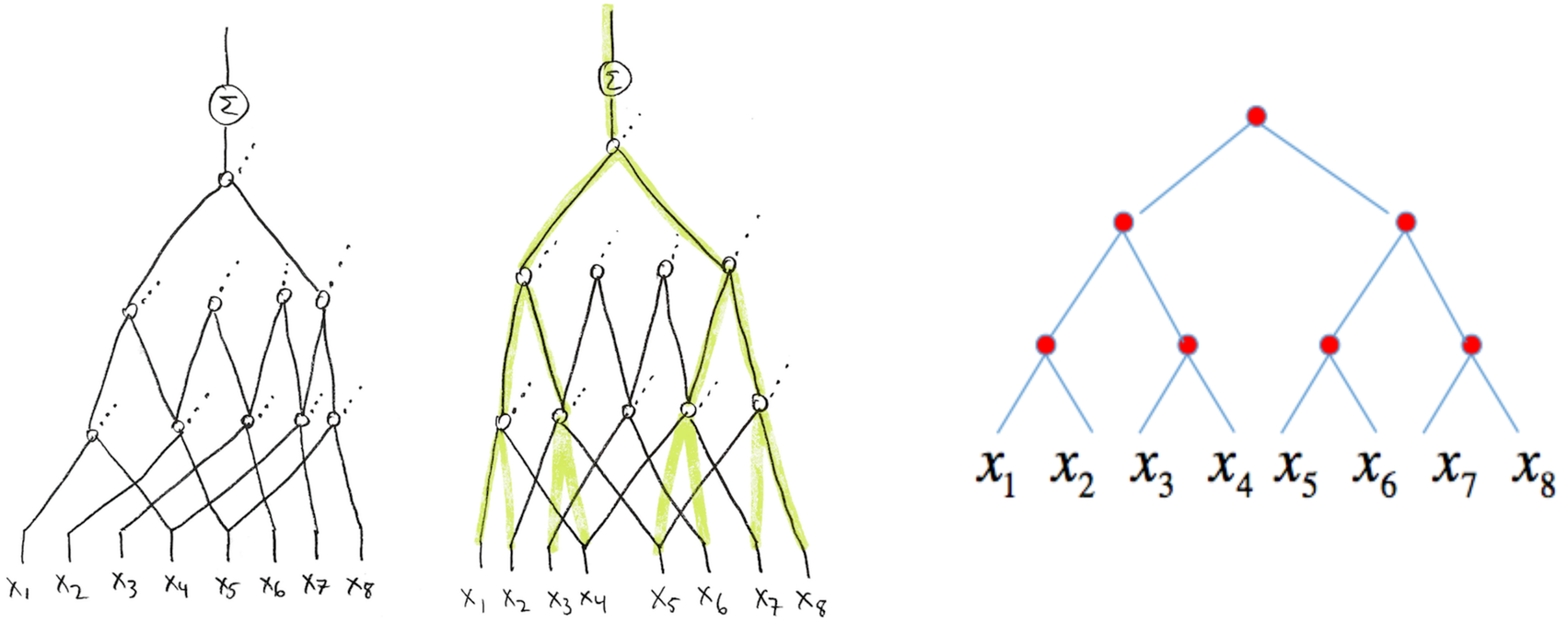
$$h_3(x,y) = 1.3\sqrt{1.2x + 1.3y}$$

$$h_2(x,y) = 0.6(1.1x + 1.9y)^3$$

$$h_1(x,y) = 0.7(x + 2y)^2$$

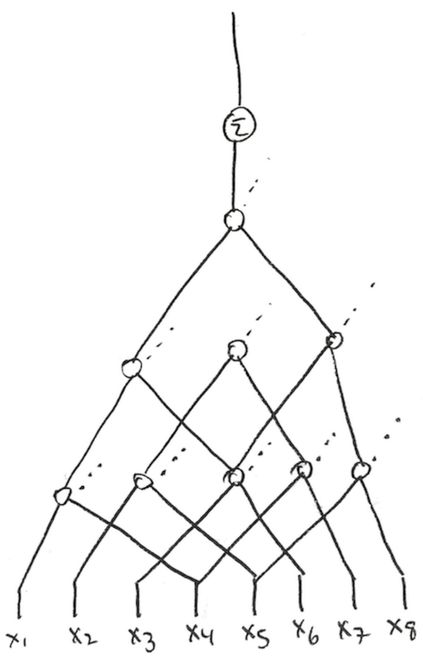
Poggio, T., Mhaskar, H., Rosasco, L. *et al.* Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *Int. J. Autom. Comput.* **14**, 503–519 (2017). https://doi.org/10.1007/s11633-017-1054-2

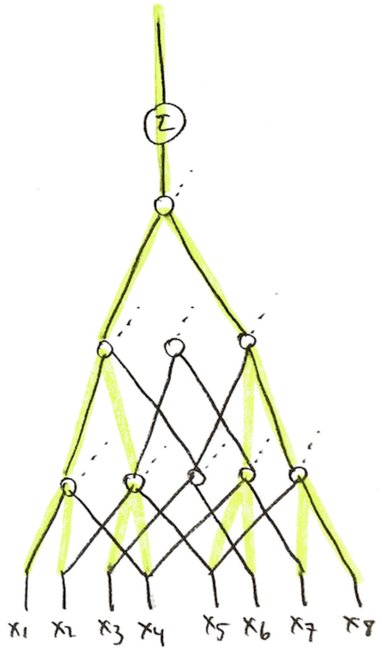
Subgraph #4

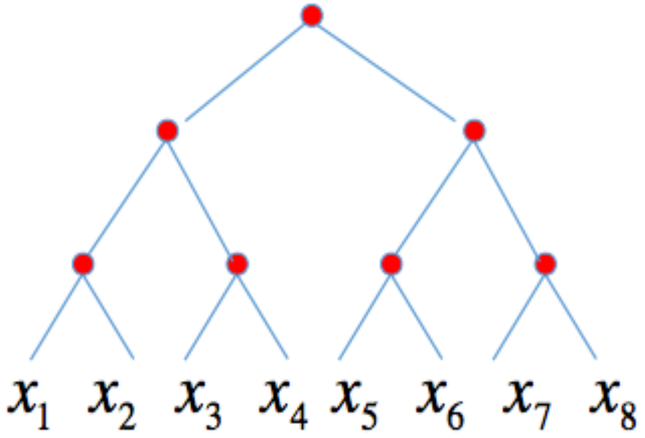


Poggio, T., Mhaskar, H., Rosasco, L. *et al.* Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *Int. J. Autom. Comput.* **14**, 503–519 (2017). https://doi.org/10.1007/s11633-017-1054-2

Subgraph #5

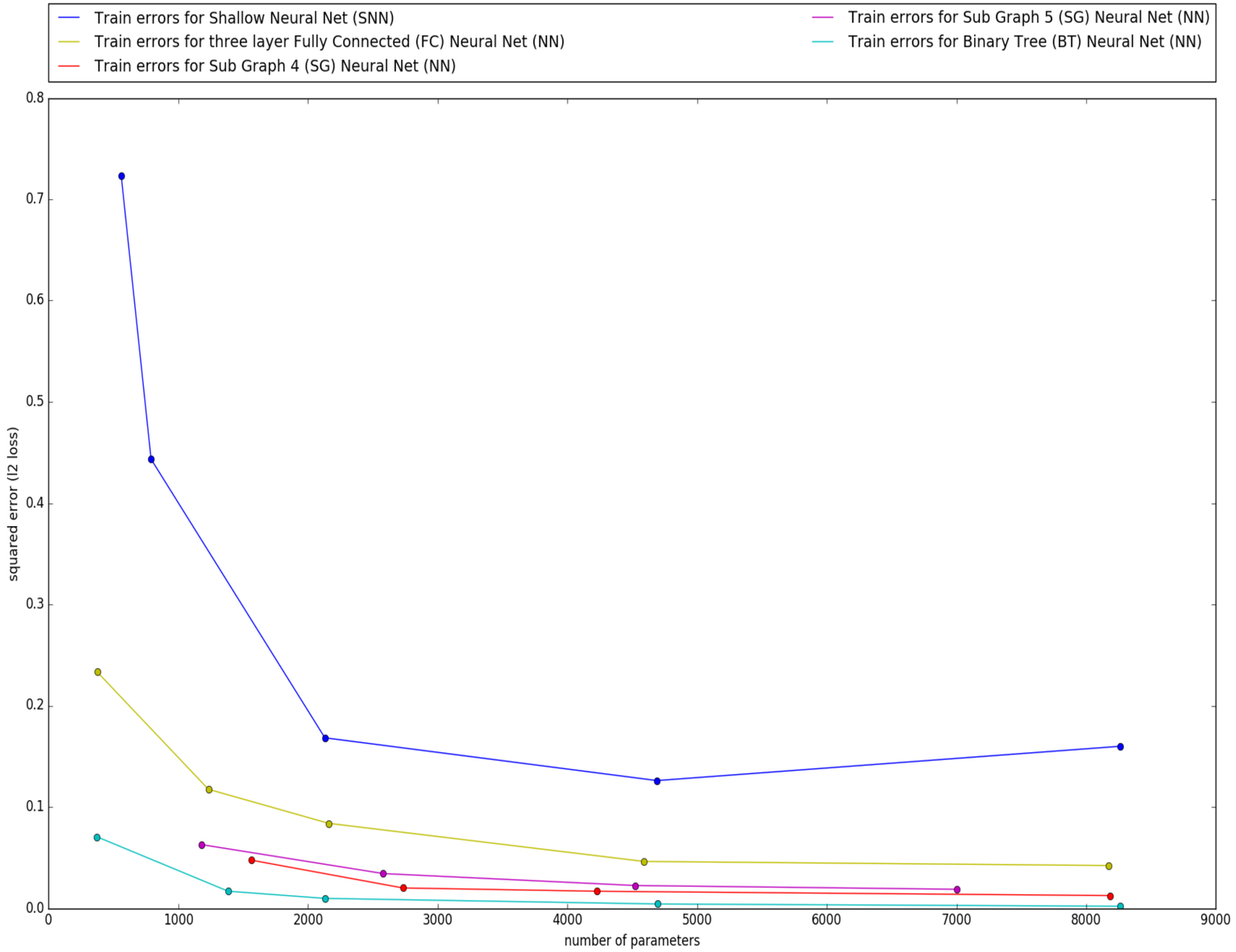






Poggio, T., Mhaskar, H., Rosasco, L. *et al.* Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *Int. J. Autom. Comput.* **14**, 503–519 (2017). https://doi.org/10.1007/s11633-017-1054-2

When the function is a subgraph of the Network



Poggio, T., Mhaskar, H., Rosasco, L. *et al.* Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *Int. J. Autom. Comput.* **14**, 503–519 (2017). https://doi.org/10.1007/s11633-017-1054-2

Learning a shallow function

Poggio, T., Mhaskar, H., Rosasco, L. *et al.* Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *Int. J. Autom. Comput.* **14**, 503–519 (2017). https://doi.org/10.1007/s11633-017-1054-2

Boolean functions: the staircase property

Definition 1.1. *A function* $g : \{-1,1\}^n \to \mathbb{R}$ *satisfies the* $[M_1, M_2]$*-staircase property over the unbiased binary hypercube if:*

- *for all* $S \subset [n]$*, if* $\hat{g}(S) \neq 0$ *then* $|\hat{g}(S)| \in [M_1, M_2]$.
- *for all* $S \subset [n]$*, if* $\hat{g}(S) \neq 0$ *and* $|S| \geq 2$*, there is* $S' \subset S$ *such that* $|S \setminus S'| = 1$ *and* $\hat{g}(S') \neq 0$.

Furthermore, g *is said to be an* s*-sparse polynomial if* $|\{S : \hat{g}(S) \neq 0\}| \leq s$.

Example

$$S_k(x) = x_1 + x_1x_2 + x_1x_2x_3 + x_1x_2x_3x_4 + \cdots + \chi_{1:k}\,.$$

The staircase property:
How hierarchical structure can guide deep learning

Emmanuel Abbe
EPFL
emmanuel.abbe@epfl.ch

Enric Boix-Adsera
MIT
eboix@mit.edu

Matthew Brennan
MIT

Guy Bresler
MIT
guy@mit.edu

Dheeraj Nagaraj
MIT
dheeraj@mit.edu

Decision trees can be staircase

In a smoothed complexity setting, where the input distribution is drawn from the biased binary hypercube, and the bias of each variable is randomly chosen, the class of log(n)-depth decision trees satisfies the general staircase structure. This is because Lemma 3 in [37] implies that with high probability there is a poly(n=)-sparse polynomial that -approximates the decision tree and satisfies the staircase property over the biased binary hypercube. Thus, the extension of our result to the case of biased binary inputs would imply that regular neural networks learn decision trees in the smoothed complexity model.

The staircase property of polynomials

Lemmas

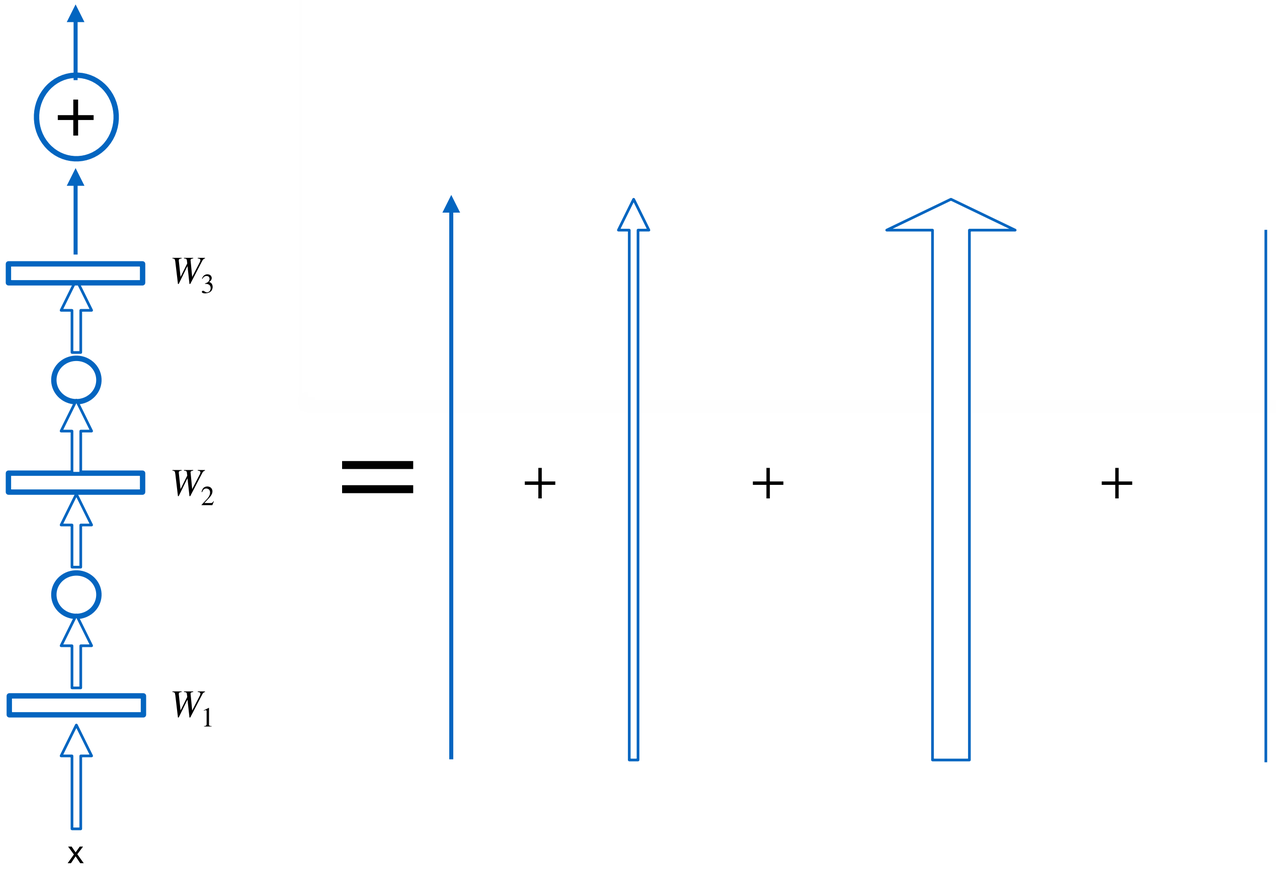
- *The composition of staircase functions is staircase*
- *Consider a DAG with first layer constituent functions being staircase. Then the compositional function associated with the DAG is staircase*
- *A sparse polynomial has a sparse compositional representation*

Target function

Assume that the target function is staircase

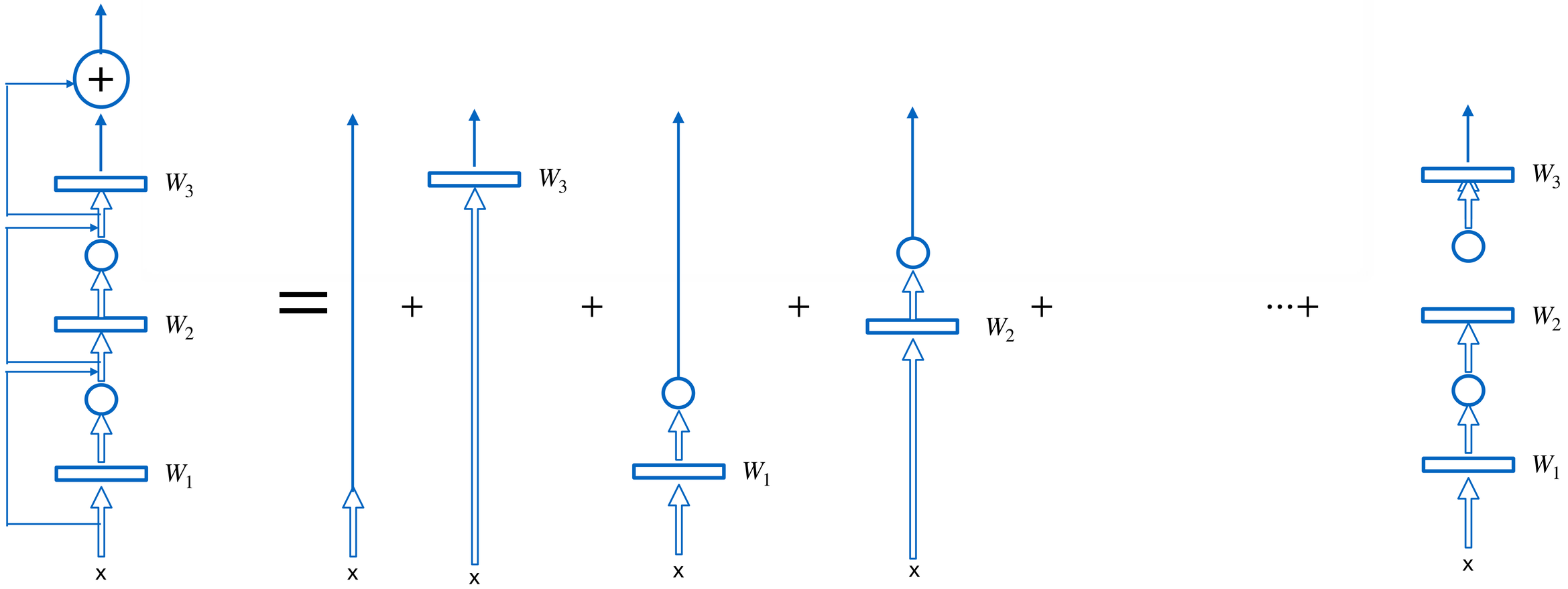
Approximating network

If a network has an activation function *containing* $1 + z + z^2$ all the polynomials generated by the network are staircase and thus the network is equivalent to the sum of the networks on the left with the property that each quadratic term contains at least one of the linear terms and so on to guarantee that the overall function generated by the network is staircase.



Approximating network

If a residual network has an activation function *containing* the polynomials generated by the network are staircase even without restrictions on the activity function



SGD on staircase network: conjecture

SGD first fits the linear terms (using support selection theorem (SST) on the linear network of the figure). The associated weights in the first layer remain non-zero even in the presence of regularization. Thus nonzero second order monomials will be preserved in the second layer; some of them will be pruned out (use the SST on the order 2 terms after the first activation layer). The same will happen after the second activation layer wrt monomials of order 3 and 4

End-to-end learning vs autoregressive framework

Suppose $f = g_1 \circ g_2 \cdots \circ g_d$

Two learning frameworks:

- I have training sets for each g_i, that is x_i, y_i, where $g_i(x_i) = y_i$ (autoregressive framework, see classes on transformers, especially Malach's)

- I have a training set for f that consists of (x_i, y_i) where $f(x_i) = y_i$ (standard supervised framework/end-to-end)

Remarks

The conjecture in guiding future work is that useful learning algorithms (that learn well) exploit sparsity. A case in point is staircase functions. Another one is decision trees which can be approximated by sparse Boolean functions, where the number of Fourier coefficients is polynomial in the size of the tree. Notice that every Boolean function of n variables can be represented by a tree of $\leq 2^n$ nodes and of depth $\leq n$.

Claim 5.3 (Mansour, 1994) Let T be a decision tree with m nodes. There exists a sparse function h such that all the Fourier coefficients of h on sets larger than t are 0 , where $t = \log m/\varepsilon$, and $E\left[(T-h)^2\right] \leq \varepsilon$.

Today's plan

1. Loss landscape ✓
2. Intuition about SGD convergence to global, degenerate minima ✓
3. Dynamics of Optimization/training under the square loss ✓
4. Gradient Descent and SGD ✓
5. Low-rank bias and SGD noise ✓
6. Neural Collapse ✓
7. Architectures and prior bias on sparse decomposition ✓
8. Appendix