The Hidden Convex Optimization Landscape of Deep Neural Networks

One World Optimization Seminar

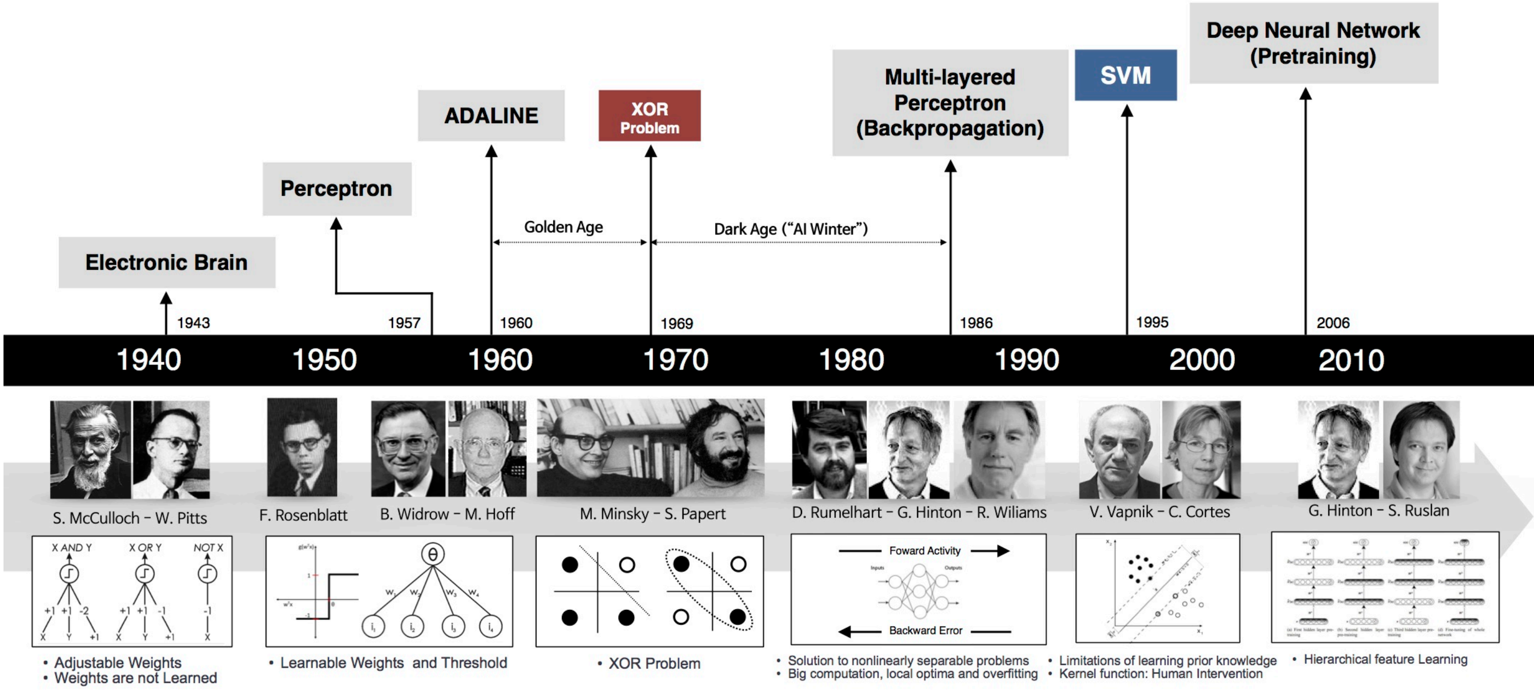
Mert Pilanci

Electrical Engineering

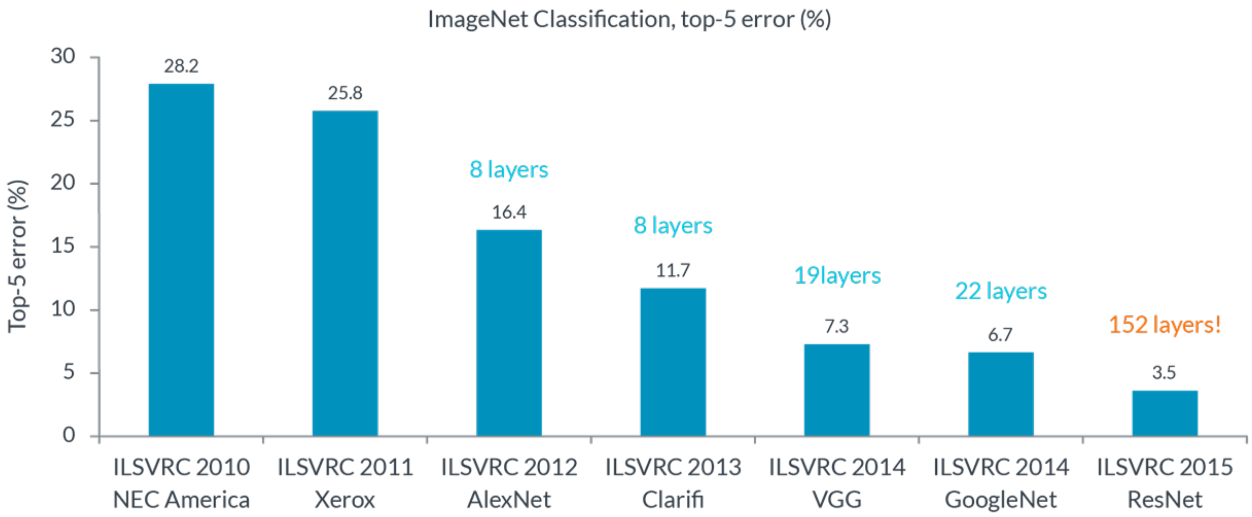
Stanford University

joint work with Tolga Ergen, Jonathan Lacotte and Yifei Wang

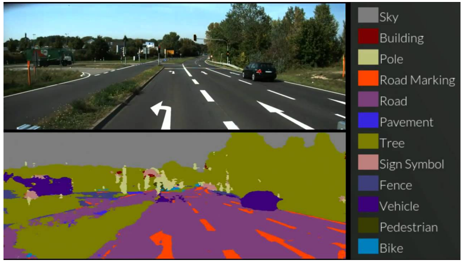
History of Artificial Neural Networks

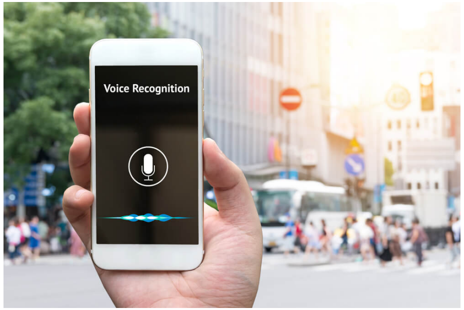


Deep learning revolution



The Impact of Deep Learning





Y. LeCun, Y. Bengio, G. Hinton (2015)

The Impact of Deep Learning

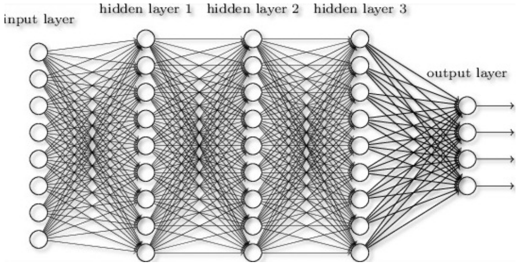
these are not real people

- Generative Adversarial Networks, Goodfellow et al. (2014), Karras et al. (2018)

Outline

- Challenges in neural networks
- ReLU neural networks are convex models
- Role of the architecture
- Generative Adversarial Networks
- Deeper ReLU networks

Deep Neural Networks



- non-convex (stochastic) gradient descent
- extremely high-dimensional problems

  152 layer ResNet-152: 60.2 Million parameters (2015)

  GPT[1]-3 language model: 175 Billion parameters (May 2020)

  BAAI[2] multi-modal model: 1.75 Trillion parameters (June 2021)

[1] OpenAI General Purpose Transformer

[2] The Beijing Academy of Artificial Intelligence

deep learning models

- often provide the best performance due to their large capacity

  $\rightarrow$ **challenging to train**

  GPT-3 is estimated to cost $12 Million for a single training run

  requires large non-public datasets

deep learning models

- often provide the best performance due to their large capacity
  $\rightarrow$ **challenging to train**
- are complex black-box systems based on non-convex optimization
  $\rightarrow$ **hard to interpret what the model is actually learning**

deep learning models

- often provide the best performance due to their large capacity
  $\rightarrow$ **challenging to train**
- are complex black-box systems based on non-convex optimization
  $\rightarrow$ **hard to interpret what the model is actually learning**

nature

Letter | Published: 29 August 2018

Deep learning of aftershock patterns following large earthquakes

Phoebe M. R. DeVries, Fernanda Viégas, Martin Wattenberg & Brendan J. Meade

Nature **560**, 632–634(2018) | Cite this article

deep learning models

- often provide the best performance due to their large capacity
  $\rightarrow$ **challenging to train**
- are complex black-box systems based on non-convex optimization
  $\rightarrow$ **hard to interpret what the model is actually learning**

one year later, another paper

logistic regression performs just as good as the 6 layer NN

nature

Matters Arising | Published: 02 October 2019

One neuron versus deep learning in aftershock prediction

Arnaud Mignan & Marco Broccardo

Nature **574**, E1–E3(2019) | Cite this article

Interpretability is important

Example: Deep networks for MR image reconstruction (FastMRI Challenge, 2020)

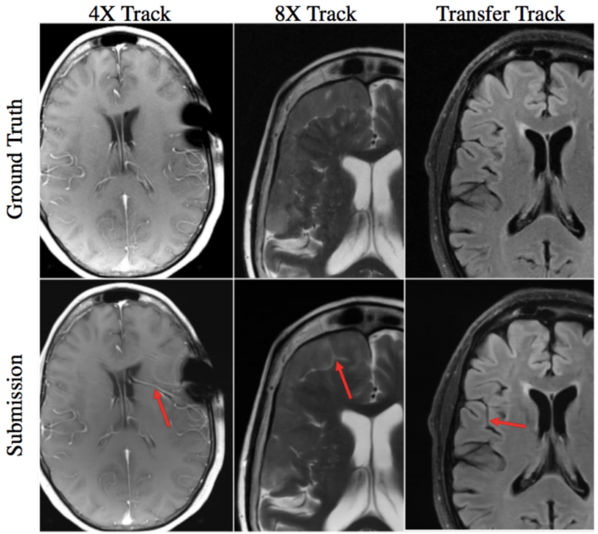
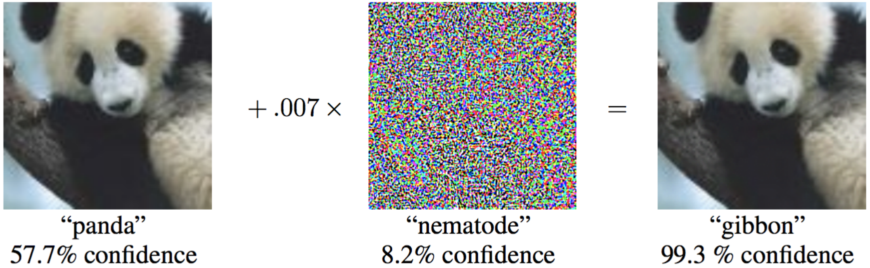


Figure 7: Examples of reconstruction hallucinations among challenge submissions. (*left*) A 4X submission from Neurospin generated a false vessel, possibly related to susceptibilities introduced by surgical staples. (*center*) An 8X submission from ATB introduced a linear bright signal mimicking a cleft of cerebrospinal fluid, as well as blurring of the boundaries of the extra-axial mass. (*right*) A submission from ResoNNance introduced a false sulcus or prominent vessel.

Adversarial examples



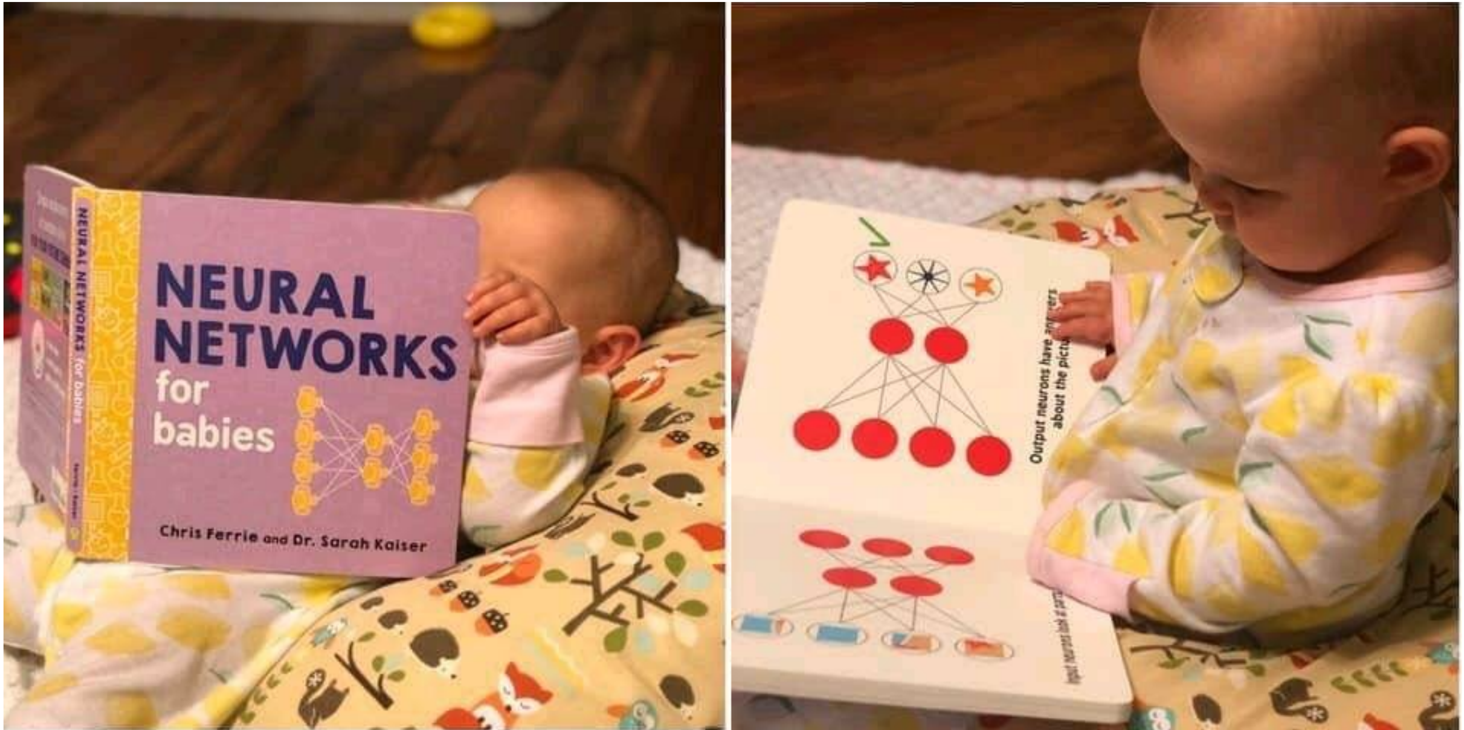
- adversarial examples, Szegedy et al., 2014, Goodfellow et al., 2015
- stop sign recognized as speed limit sign, Evtimov et al, 2017

Questions

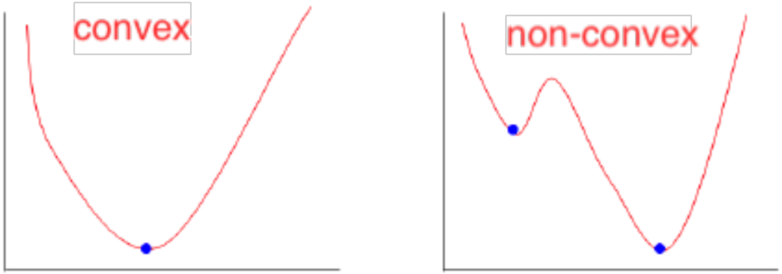
- **What are neural networks actually doing?**
- **Are they automatically finding the 'best' features?**
- **Is it possible to establish optimality?**
- **Is there a more efficient way?**

  deep convnet (2012), transformer (2017), fully connected mixer (May 2021), ...?

How neural networks work?



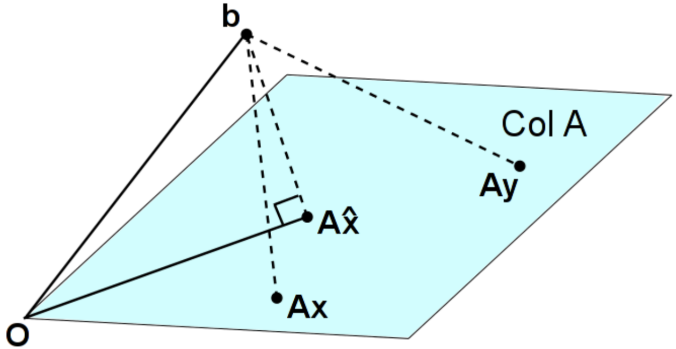
How neural networks work?



- least-squares, logistic regression, support vector machines etc. are understood extremely well
- the choice of the solver does not matter
- insightful theorems for neural networks?

Least Squares

$$\min_x \|Ax - b\|_2^2$$



convex optimality condition: $A^T Ax = A^T b$

efficient solvers: conjugate gradient (CG), preconditioned CG, QR, Cholesky...

Least Squares with L1 Regularization

$$\min_x \|Ax - y\|_2^2 + \lambda \|x\|_1$$

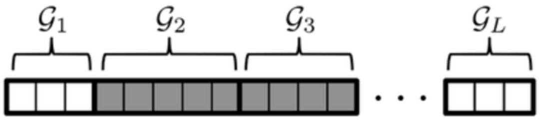
Lasso

- L1 norm $\|x\|_1 = \sum_{i=1}^d |x_i|$ encourages sparsity in the solution x^*

R. Tibshirani (1996), E.J. Candes & T. Tao (2005), D.L. Donoho (2006)

Least Squares with Group L1 regularization

$$\min_x \|\sum_{i=1}^{k} A_i x_i - y\|_2^2 + \lambda \sum_{i=1}^{k} \|x_i\|_2$$



Group Lasso

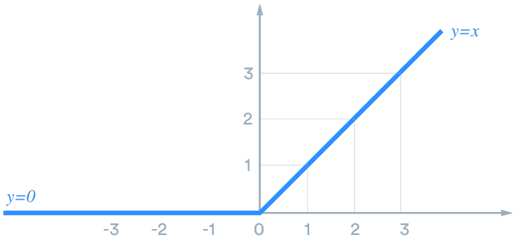
- encourages group sparsity in the solution x^*, i.e., most blocks x_i are zero
- convex optimization and convex regularization methods are well understood

Yuan & Lin (2007), Cevher et al. (2009, 2013, 2016)

Two-Layer Neural Networks with Rectified Linear Unit (ReLU) activation

$$p_{\text{non-convex}} := \underset{\substack{W_1 \in \mathbb{R}^{d\times m} \\ W_2 \in \mathbb{R}^{m\times 1}}}{\text{minimize}} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right)$$

where $\phi(u) = \text{ReLU}(u) = (u)_+$



Neural Networks are Convex Regularizers

$$p_{\text{non-convex}} := \text{minimize} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda \left(\|W_1\|_F^2 + \|W_2\|_F^2\right)$$

$$W_1 \in \mathbb{R}^{d\times m}$$

$$W_2 \in \mathbb{R}^{m\times 1}$$

$$p_{\text{convex}} := \text{minimize} \quad L\left(Z, y\right) + \lambda \underbrace{R(Z)}_{\text{convex regularization}}$$

$$Z \in \mathcal{K} \subseteq \mathbb{R}^{d\times p}$$

$$p_{\text{non-convex}} := \text{minimize} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right)$$
$$W_1 \in \mathbb{R}^{d\times m}$$
$$W_2 \in \mathbb{R}^{m\times 1}$$

$$p_{\text{convex}} := \text{minimize} \quad L\left(Z, y\right) + \lambda R(Z)$$
$$Z \in \mathcal{K} \subseteq \mathbb{R}^{d\times p}$$

Theorem $p_{\text{non-convex}} = p_{\text{convex}}$, and an optimal solution to $p_{\text{non-convex}}$ can be obtained from an optimal solution to p_{convex}.

M. Pilanci, T. Ergen **Neural Networks are Convex Regularizers: Exact Polynomial-time Convex Optimization Formulations for Two-Layer Networks, ICML 2020**

Squared Loss: ReLU Neural Networks are Convex Group Lasso Models

data matrix $X \in \mathbb{R}^{n \times d}$ and label vector $y \in \mathbb{R}^n$

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$p_{\text{non-convex}} = \text{ minimize}_{W_1, W_2} \left\| \sum_{j=1}^{m} \phi(XW_{1j})W_{2j} - y \right\|_2^2 + \lambda \left(\|W_1\|_F^2 + \|W_2\|_F^2 \right)$$

$$p_{\text{convex}} = \text{ minimize}_{u_1, v_1 \ldots u_p, v_p \in \mathcal{K}} \left\| \sum_{i=1}^{p} D_i X(u_i - v_i) - y \right\|_2^2 + \lambda \left(\sum_{i=1}^{p} \|u_i\|_2 + \|v_i\|_2 \right)$$

$D_1, ..., D_p$ are fixed diagonal matrices

Theorem $p_{\text{non-convex}} = p_{\text{convex}}$, and an optimal solution to $p_{\text{non-convex}}$ can be recovered from optimal non-zero u_i^*, v_i^*, $i = 1, ..., p$ as
$W_{1i}^* = \frac{u_i^*}{\sqrt{\|u_i^*\|_2}}$, $W_{2i} = \sqrt{\|u_i^*\|_2}$ or $W_{1i}^* = \frac{v_i^*}{\sqrt{\|v_i^*\|_2}}$, $W_{2i} = -\sqrt{\|v_i^*\|_2}$.

Regularization Path

$$p_{\text{convex}} = \ \text{minimize}_{u_1,v_1\ldots u_p,v_p\in\mathcal{K}} \left\| \sum_{i=1}^{p} D_i X(u_i - v_i) - y \right\|_2^2 + \lambda \left(\sum_{i=1}^{p} \|u_i\|_2 + \|v_i\|_2 \right)$$

- As $\lambda \in (0,\infty)$ increases, the number of non-zeros in the solution decreases

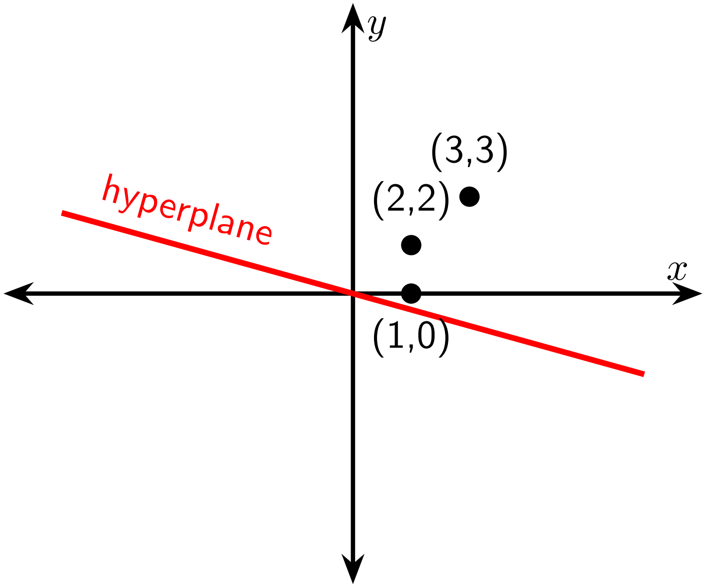
  Corollary

  Optimal solutions of p_{convex} generates the entire set of optimal architectures $f(x) = W_2\phi(W_1 x)$ with m neurons for $m = 1, 2, \ldots,$

  where $W_1 \in \mathbb{R}^{d\times m},\ W_2 \in \mathbb{R}^{m\times 1}$

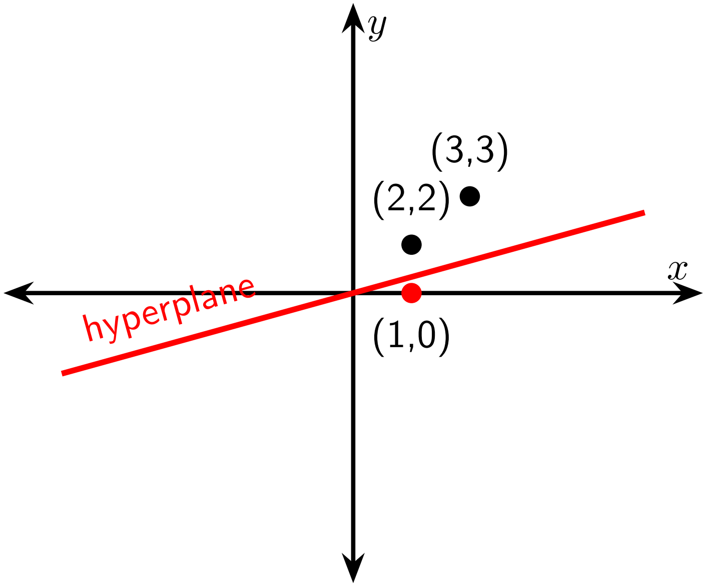
- **non-convex NN models correspond to regularized convex models**

$n = 3$ samples in $\mathbb{R}^d$, $d = 2$ $\quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$



$$D_1 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}$$

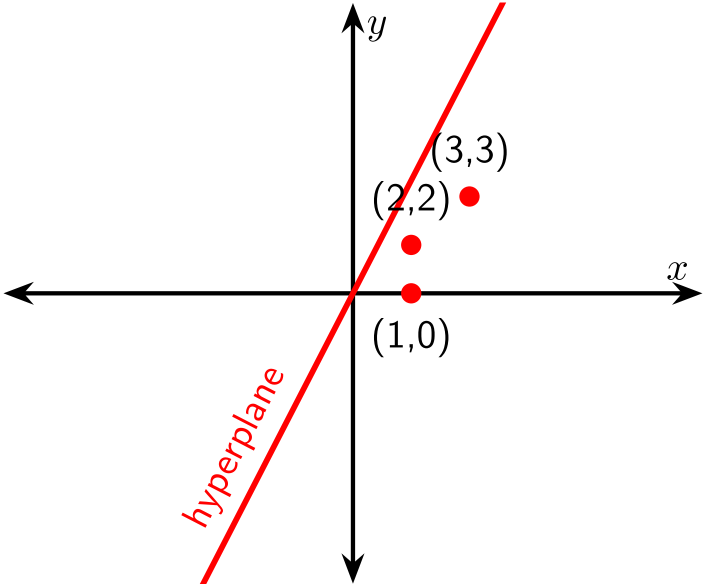
$n = 3$ samples in $\mathbb{R}^d$, $d = 2$ $\quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$



$$D_1 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}$$

$$D_2 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 0 & 0 \end{bmatrix}$$

$n = 3$ samples in $\mathbb{R}^d$, $d = 2$ $\quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$

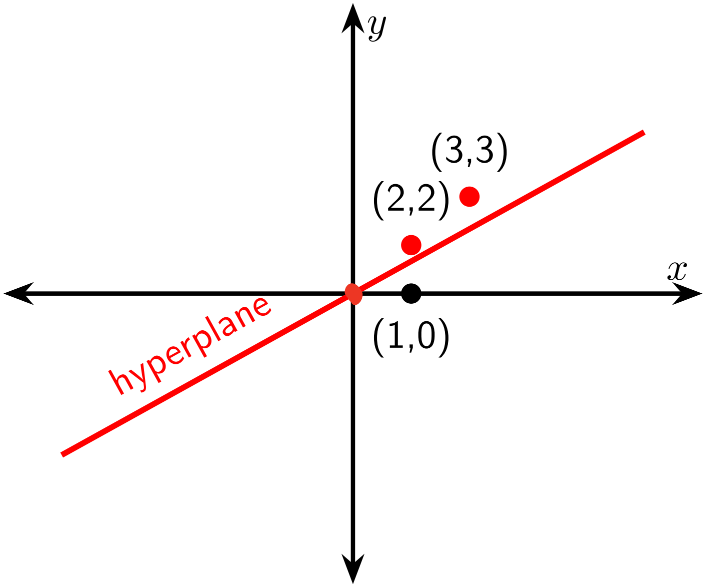


$$D_1 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}$$

$$D_2 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 0 & 0 \end{bmatrix}$$

$$D_3 X = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} X = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$n = 3$ samples in $\mathbb{R}^d$, $d = 2$ $\quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$



$$D_1 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}$$

$$D_2 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 0 & 0 \end{bmatrix}$$

$$D_4 X = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$

Example: Convex Program for $n = 3, d = 2$

$$n = 3 \text{ samples} \quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\min \left\| \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} (u_1 - v_1) + \begin{bmatrix} x_1^T \\ x_2^T \\ 0 \end{bmatrix} (u_2 - v_2) + \begin{bmatrix} 0 \\ 0 \\ x_3^T \end{bmatrix} (u_3 - v_3) - y \right\|_2^2$$

subject to

$$+ \lambda \Big(\sum_{i=1}^{3} \|u_i\|_2 + \|v_i\|_2 \Big)$$

$$D_1 X u_1 \geq 0, D_1 X v_1 \geq 0$$

$$D_2 X u_2 \geq 0, D_2 X v_2 \geq 0$$

$$D_4 X u_3 \geq 0, D_4 X v_3 \geq 0$$

equivalent to the non-convex two-layer NN problem

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \in \mathbb{R}^{n \times d} \xrightarrow[\text{network}]{\text{neural}} \bar{X} = [D_1 X, ..., D_p X] \in \mathbb{R}^{n \times p}$$

neural network $=$ convex regularization applied to $\bar{X}$

Computational Complexity

Learning two-layer ReLU neural networks with m neurons

$f(x) = \sum_{j=1}^{m} W_{2j}\phi(W_{j1}x)$

Previous result: ∘ Combinatorial $O(2^m n^{dm})$ (Arora et al., ICLR 2018)

$\boxed{\text{Convex program } O((\tfrac{n}{r})^r) \text{ where } r = \operatorname{rank}(X)}$

Computational Complexity

Learning two-layer ReLU neural networks with m neurons

$f(x) = \sum_{j=1}^{m} W_{2j}\phi(W_{j1}x)$

Previous result: ∘ Combinatorial $O(2^m n^{dm})$ (Arora et al., ICLR 2018)

$\boxed{\text{Convex program } O((\tfrac{n}{r})^r) \text{ where } r = \mathrm{rank}(X)}$

n : number of samples, d : dimension

(i) polynomial in n and m for fixed rank r

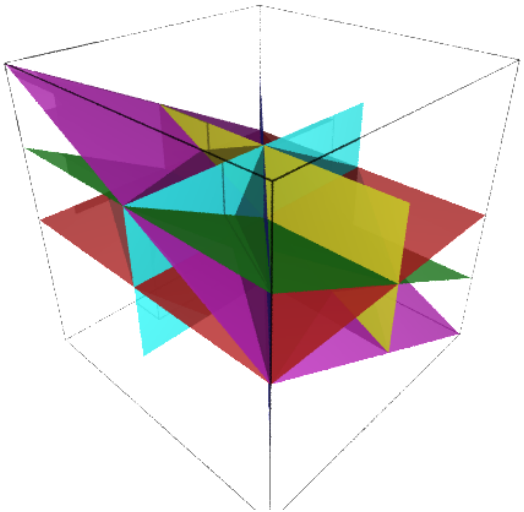
(ii) exponential in d for full rank data $r = d$. This can not be improved unless $P = NP$ even for $m = 1$.

Hyperplane Arrangements

Let $X \in \mathbb{R}^{n \times d}$

$$\{\mathbf{sign}(Xw) \ : w \in \mathbb{R}^d\}$$

at most $2\sum_{k=0}^{r-1}\binom{n}{k} \leq O\big((\frac{n}{r})^r\big)$ patterns where $r = \mathbf{rank}(X)$.

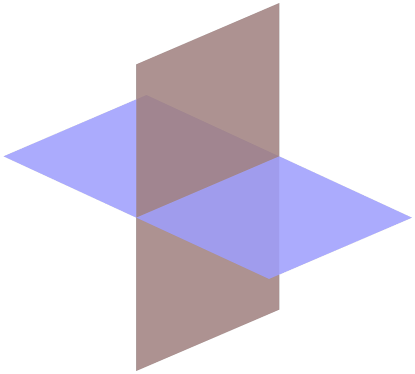


Convolutional Hyperplane Arrangements

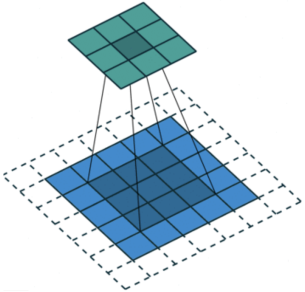
Let $X \in \mathbb{R}^{n \times d}$ be partitioned into patch matrices $X = [X_1, ..., X_K]$ where $X_k \in \mathbb{R}^{n \times h}$

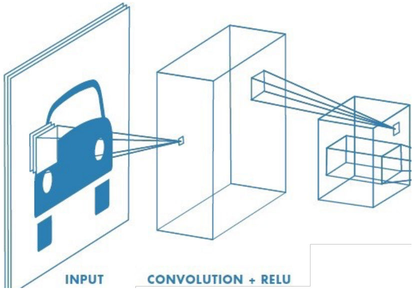
$$\{\mathbf{sign}(X_k w) \ : w \in \mathbb{R}^h\}_{k=1}^K$$

at most $O\big((\frac{nK}{h})^h\big)$ patterns where h is the filter size.



Convolutional Neural Networks can be optimized in fully polynomial time





- $f(x) = W_2\phi(W_1 x)$, $W_1 \in \mathbb{R}^{d\times m}$, $W_2 \in \mathbb{R}^{m\times 1}$

  m filters (neurons), h filter size

  typical example: 1024 filters of size 3×3 $(m = 1024, h = 9)$

  convex optimization complexity: polynomial in all parameters n, m and d

M. Pilanci, T. Ergen **Implicit Convex Regularizers of CNN Architectures, ICLR 2021**

Approximating the Convex Program

$$p_{\text{convex}} = \text{minimize}_{u_1,v_1\ldots u_p,v_p\in\mathcal{K}} \left\| \sum_{i=1}^{p} D_i X(u_i - v_i) - y \right\|_2^2 + \lambda\Big(\sum_{i=1}^{p} \|u_i\|_2 + \|v_i\|_2 \Big)$$

- Sample $D_1, ..., D_p$ as Diag$(Xu \geq 0)$ where $u \sim N(0, I)$
- Low rank approximation of $X \approx X_r$ where $\|X - X_r\|_2 \leq \sigma_{r+1}$
  $(1 + \frac{\sigma_{r+1}}{\lambda})$ approximation in $O\big((\frac{n}{r})^r\big)$ complexity
- Backpropagation (gradient descent) on the non-convex loss
  is a **heuristic** for the convex program

An Exact Characterization of All Optimal Solutions

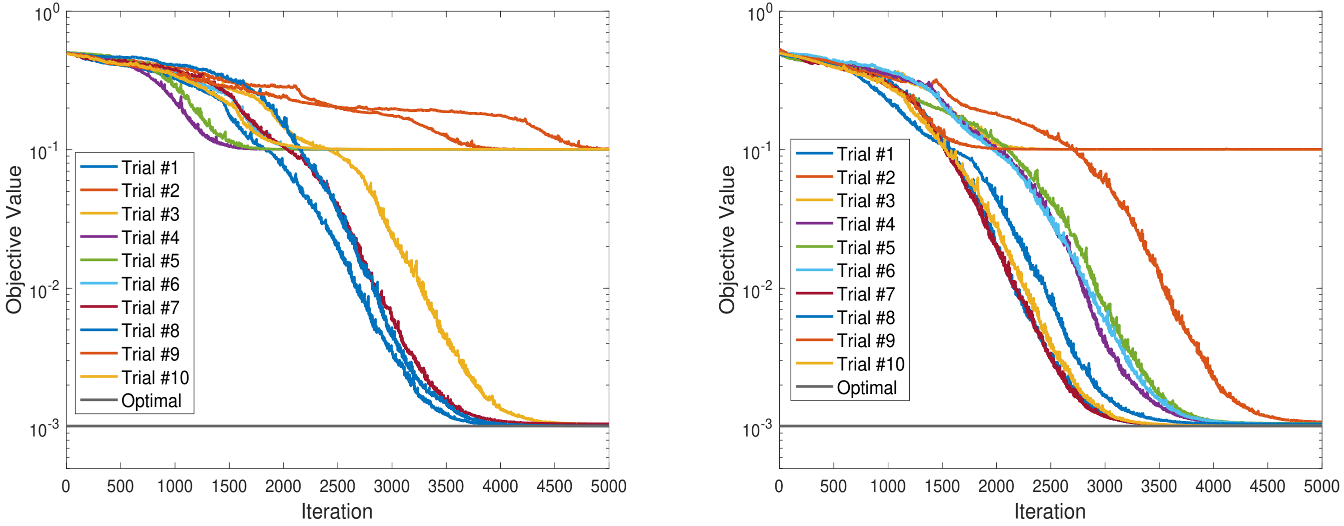
$$p_{\text{non-convex}} := \underset{\substack{W_1 \in \mathbb{R}^{d\times m} \\ W_2 \in \mathbb{R}^{m\times 1}}}{\text{minimize}} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right)$$

$$p_{\text{convex}} := \underset{Z \in \mathcal{K} \subseteq \mathbb{R}^{d\times p}}{\text{minimize}} \quad L\left(Z, y\right) + \lambda R(Z)$$

Theorem All optimal solutions of $p_{\text{non-convex}}$ can be found from the optimal solutions of p_{convex} up to permutation and neuron splitting. Hence, the optimal set of $p_{\text{non-convex}}$ is convex up to equivalence.

Y. Wang, J. Lacotte, M. Pilanci, **The Hidden Convex Optimization Landscape of Two-Layer ReLU Neural Networks, ICLR 2022**

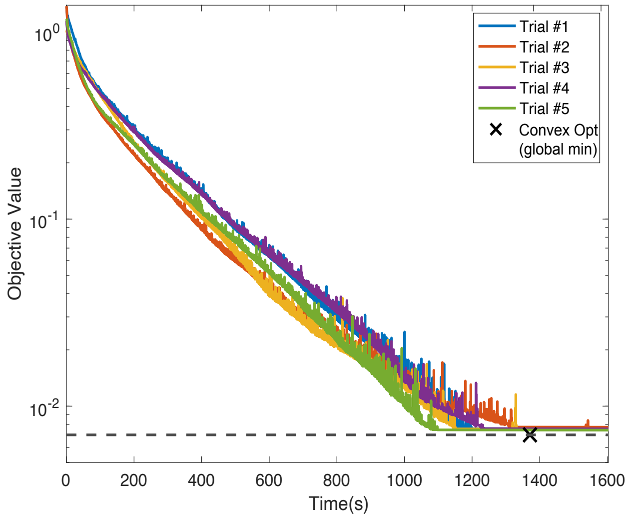
Numerical Experiment: Two-Layer Fully Connected ReLU

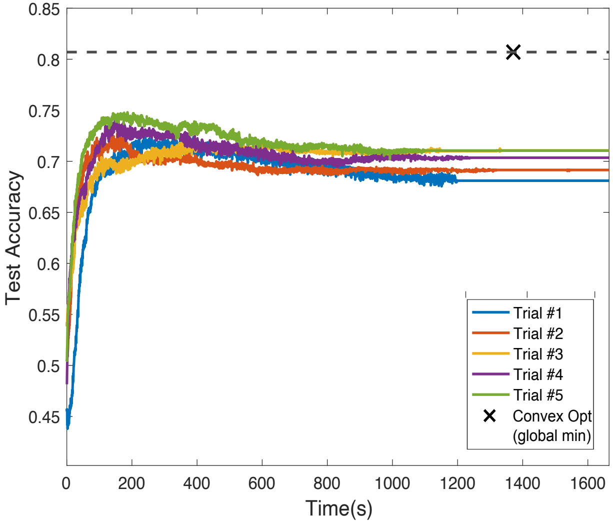


$m = 8$ $\qquad$ $m = 15$

Training cost of a two-layer ReLU network trained with SGD (10 initialization trials) and the convex program on a toy dataset ($d = 2$)

Numerical Experiment: Two-Layer Convolutional Network on CIFAR



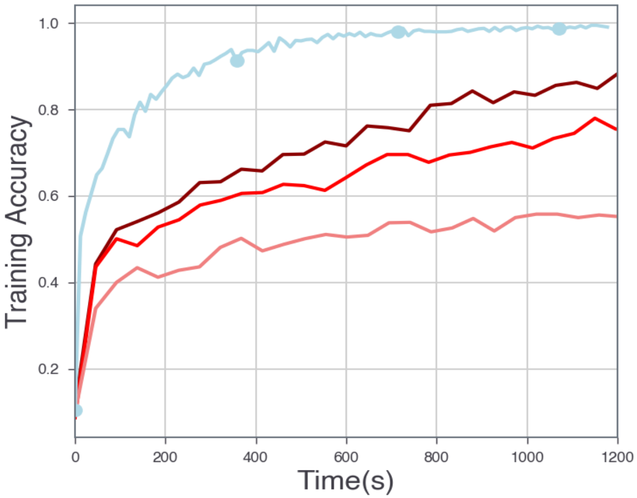


training error

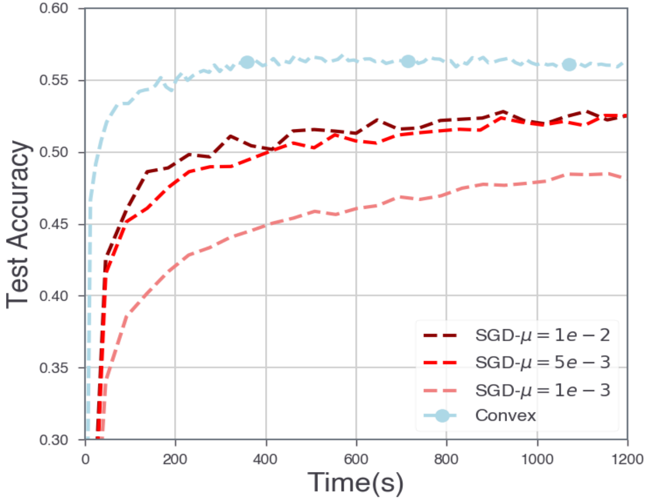
test accuracy

binary classification on a subset of the CIFAR Dataset

SGD for the Convex Program vs SGD for the Non-convex Problem



training accuracy



test accuracy

10-class classification on the CIFAR Dataset ($n = 50,000$, $d = 3072$) with randomly sampled arrangement patterns for the convex program

Plan for the rest of the talk

- Are all neural network problems convex? What is the role of the network architecture? What does gradient descent with no regularization do?

    vector output networks, e.g., autoencoders

    batch normalization layers

    gradient flow

    Generative Adversarial Networks (GANs)

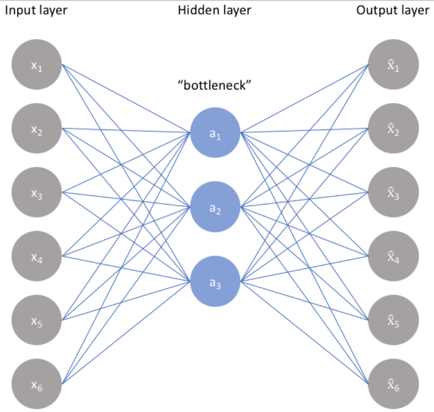
    deeper networks

- Numerical results

    convex vs non-convex neural networks

    convex GANs

Vector Output Two-layer ReLU Networks: Nuclear Norm Regularization



$$p_{\text{convex}} = \min_{U_1,V_1\ldots U_p,V_p\in\mathcal{K}} \left\|\sum_{i=1}^{p} D_i X(U_i - V_i) - y\right\|_2^2 + \lambda\left(\sum_{i=1}^{p} \|U_i\|_* + \|V_i\|_*\right)$$

Theorem $p_{\text{non-convex}} = p_{\text{convex}}$, and an optimal solution to $p_{\text{non-convex}}$ can be recovered from optimal non-zero U_i^*, V_i^*, $i = 1, \ldots, p$.

A. Sahiner, T. Ergen, J. Pauly, M. Pilanci **Vector-output ReLU Neural Network Problems are Copositive Programs, ICLR 2021**

ReLU Networks with Batch Normalization (BN)

- BN transforms a batch of data to zero mean and standard deviation one, and has two trainable parameters α, γ

$$\textbf{BN}_{\alpha,\gamma}(x) = \frac{(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)x}{\|(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)x\|_2}\gamma + \alpha$$

$$p_{\text{non-convex}} = \ \text{minimize}_{W_1,W_2,\alpha,\gamma} \left\|\mathbf{BN}_{\alpha,\gamma}(\phi(XW_1))W_2 - y\right\|_2^2 + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right)$$

$$\|$$

$$p_{\text{convex}} = \ \text{minimize}_{w_1,v_1\ldots w_p,v_p\in\mathcal{K}} \left\|\sum_{i=1}^{p} U_i(w_i - v_i) - y\right\|_2^2 + \lambda\left(\sum_{i=1}^{p}\|w_i\|_2 + \|v_i\|_2\right)$$

where $U_i\Sigma_i V_i^T = D_i X$ is the SVD of DX_i, i.e., BatchNorm whitens local data

T. Ergen, A. Sahiner, B. Ozturkler, J. Pauly, M. Mardani, M. Pilanci
Demystifying Batch Normalization in ReLU Networks, ICLR 2022

Unregularized Gradient Flow Converges to the Optimum of the Convex Program

Consider the **unregularized** problem

$$\min_{\theta} \mathcal{L}(\theta) = \min_{\theta=\{w_{11},w_{21}\ldots,w_{1p},w_{2p}\}} \ell\Big(\sum_{j=1}^{m}(Xw_{1j})_+ w_{2j}, y\Big)$$

and corresponding non-convex gradient flow

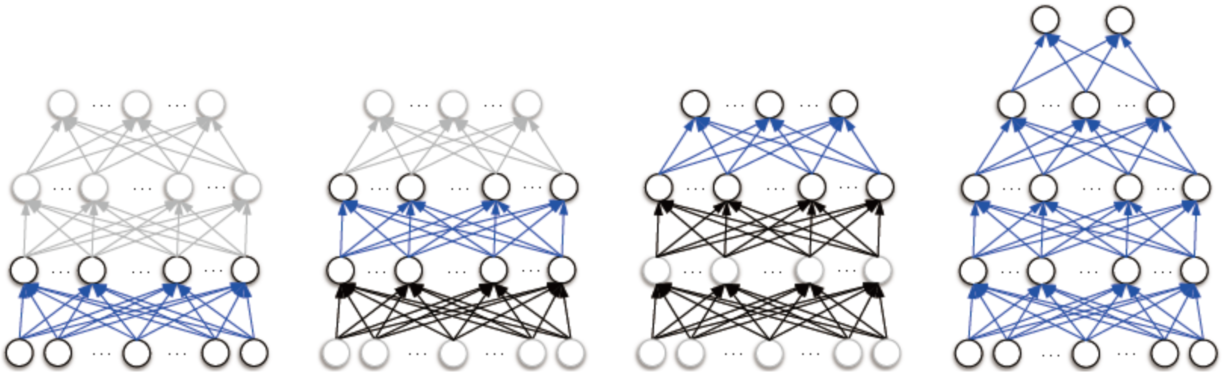
$$\frac{d}{dt}\theta(t) \in -\partial\mathcal{L}(\theta(t))$$

Theorem: Suppose that X is linearly separable, and ℓ is log loss. Then, $\theta(t)$ converges to the solution of the convex program

$$\text{minimize}_{u_1,v_1\ldots u_p,v_p\in\mathcal{K}} \sum_{i=1}^{p}\|u_i\|_2+\|v_i\|_2 \text{ s.t. } Diag(y)\sum_{i=1}^{p}D_iX(u_i-v_i)\geq 1$$

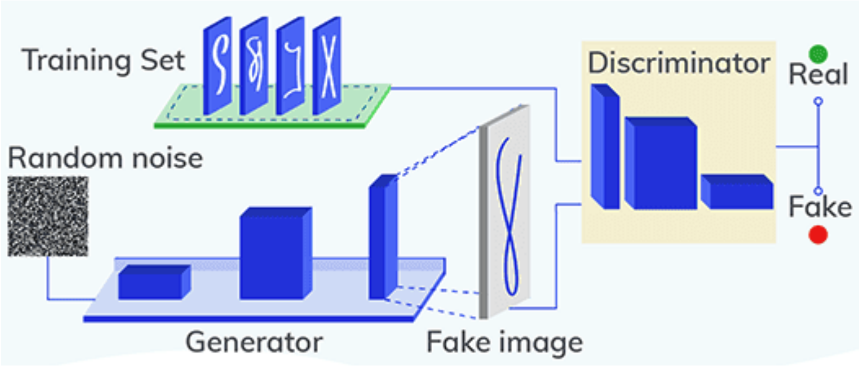
Y. Wang, M. Pilanci, **The Convex Geometry of Backpropagation: Neural Network Gradient Flows Converge to Extreme Points of the Dual Convex Program, ICLR 2022**

Layer-Wise Learning Deep Networks



	CIFAR-10	Imagenet
16 Layer NN (VGG16) (Simonyan et al. 2015)	92%	90.9%
Layerwise (2-Layer$\times 15$) (Belilovsky et al. 2019)	90.4%	88.7%

Convex Generative Adversarial Networks (GANs)



- Wasserstein GAN parameterized with neural networks

$$p^* = \min_{\theta_g} \max_{D:\ \text{1-Lipschitz}} \mathbb{E}_{x\sim p_x}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G_{\theta_g}(z))]$$

$$\cong \min_{\theta_g} \max_{\theta_d} \mathbb{E}_{x\sim p_x}[D_{\theta_d}(x)] - \mathbb{E}_{z\sim p_z}[D_{\theta_d}(G_{\theta_g}(z))]$$

Theorem Two layer generator two layer discriminator WGAN problems are convex-concave games.

- two-layer ReLU-activation generator $G_{\theta_g}(Z) = (ZW_1)_+ W_2$
- two-layer quadratic activation discriminator $D_{\theta_d}(X) = (XV_1)^2 V_2$

  Wasserstein GAN problem is equivalent to a convex-concave game, which can be solved via convex optimization

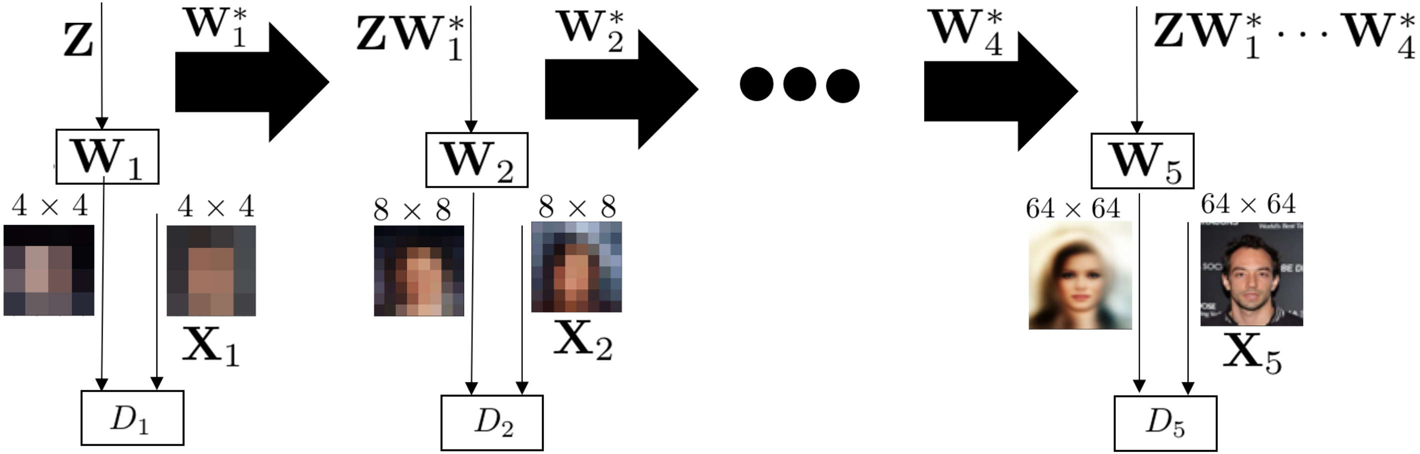
$$G^* = \text{argmin}_G \|G\|_F^2 \text{ s.t. } \|X^\top X - G^\top G\|_2 \leq \lambda$$

$$W_1^*, W_2^* = \text{argmin}_{W_1, W_2} \|W_1\|_F^2 + \|W_2\|_F^2 \text{ s.t. } G^* = (ZW_1)_+ W_2,$$

- the first problem can be solved via singular value thresholding as $G^* = U(\Sigma^2 - \lambda I)_+^{1/2} V^\top$ where $X = U\Sigma V^\top$ is the SVD of X.
- the second problem can be solved via convex optimization as shown earlier

Progressive GANs

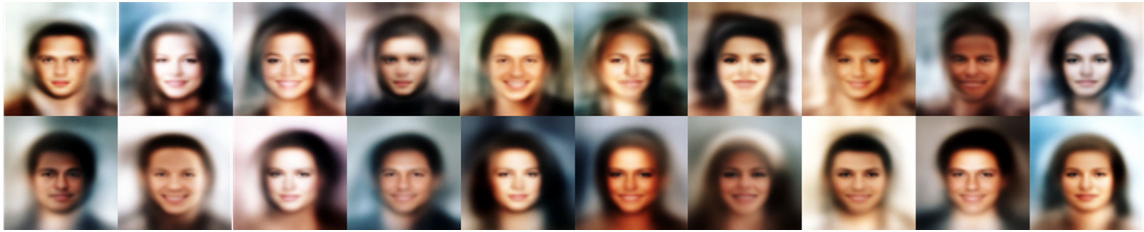
deeper architectures can be trained layerwise



Numerical Results

- real faces from the CelebA dataset

- fake faces generated using convex optimization



two-layer quadratic activation discriminator and linear generator trained via closed form optimal solution progressively for a total of 4 layers

A. Sahiner et al. **Hidden Convexity of Wasserstein GANs, ICLR 2022**

Three-layer Neural Networks: Double Hyperplane Arrangements

$$p_3^* = \min_{\substack{\{W_j, u_j, w_{1j}, w_{2j}\}_{j=1}^m \\ u_j \in \mathcal{B}_2, \forall j}} \frac{1}{2} \left\| \sum_{j=1}^m \left((\mathbf{X} W_j)_+ w_{1j} \right)_+ w_{2j} - y \right\|_2^2 + \frac{\beta}{2} \sum_{j=1}^m \left(\|W_j\|_F^2 + \|w_{1j}\|_2^2 + w_{2j}^2 \right),$$

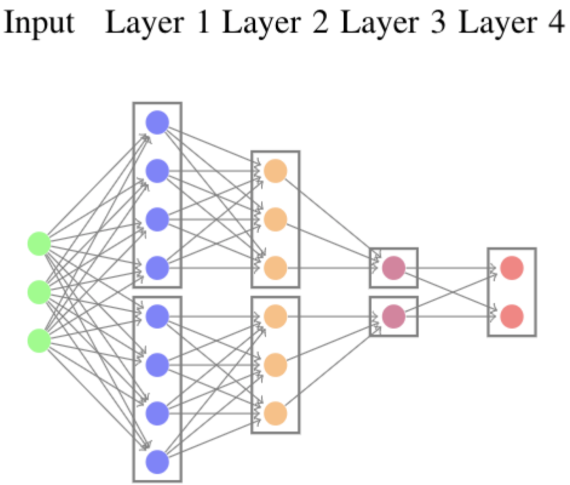
Theorem

The equivalent convex problem is

$$\min_{\{W_i, W_i'\}_{i=1}^p \in \mathcal{K}} \frac{1}{2} \left\| \sum_{i=1}^p \sum_{j=1}^P D_i D_j \tilde{\mathbf{X}} \left(W_{ij}' - W_{ij} \right) - y \right\|_2^2 + \frac{\beta}{2} \sum_{i,j=1}^p \|W_{ij}\|_F + \|W_{ij}'\|_F$$

T. Ergen, M. Pilanci **Global Optimality Beyond Two Layers: Training Deep ReLU Networks via Convex Programs, ICML 2021**

Deep ReLU Networks



arbitrarily deep ReLU neural networks with parallel architecture

Theorem There is a convex program such that $p_{\text{non-convex}} = p_{\text{convex}}$

Y. Wang, T. Ergen, M. Pilanci, **Parallel Deep Neural Networks Have Zero Duality Gap, arXiv 2021**.

Conclusion and Open Problems

- we can **train** ReLU and polynomial NNs in polynomial time
- convex optimization theory & solvers can be applied
- multi layer ReLU neural network problems are **convex** in higher dimensions
- neural networks seek **sparsity**
- architecture search $=$ regularizer search (block ℓ_2-ℓ_1, nuclear norm,...)
- we need faster algorithms to solve high-dimensional convex programs whose solutions are sparse and better layer-wise learning strategies

CODE: `github.com/pilancilab`

References

`stanford.edu/~pilanci`   CODE: `github.com/pilancilab`

- T. Ergen, M. Pilanci, Convex Geometry and Duality of Over-parameterized Neural Networks , Journal of Machine Learning Research (JMLR), 2021
- T. Ergen, M. Pilanci, Revealing the Structure of Deep Neural Networks via Convex Duality, ICML 2021
- B. Bartan, M. Pilanci, Training Quantized Neural Networks to Global Optimality via Semidefinite Programming, ICML 2021
- A. Sahiner, M. Mardani, B. Ozturkler, M. Pilanci, J. Pauly, Convex Regularization behind Neural Reconstruction, ICLR 2021
- T. Ergen, M. Pilanci, Convex geometry of two-layer relu networks: Implicit autoencoding and interpretable models, AISTATS 2020
- V. Gupta, B. Bartan, T. Ergen, M. Pilanci, Exact and Relaxed Convex Formulations for Shallow Neural Autoregressive Models, ICASSP 2021
- B. Bartan and M. Pilanci Convex Relaxations of Convolutional Nets, ICASSP 2019

References

- Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature, 2015
- I. Tolstikhin et al., An all-MLP architecture for vision, 2021, arXiv:2105.01601