EE269
Signal Processing for Machine Learning

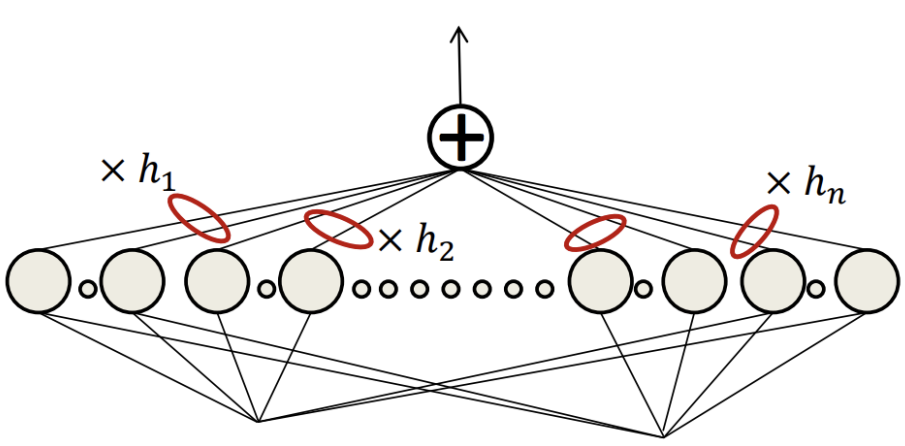
Lecture 16

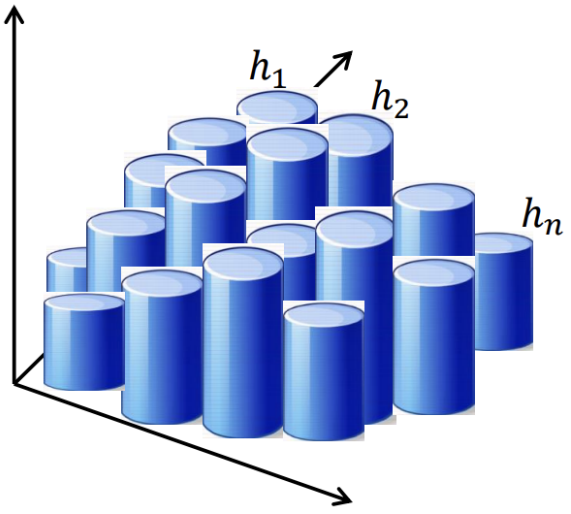
Instructor : Mert Pilanci

November 18, 2020

Stanford University

Two layer networks are universal approximators





Two layer networks (also called one hidden layer) can compose arbitrary functions to arbitrary precision

May require infinitely many neurons in the hidden layer

Deeper networks require **fewer** neurons for the same approximation error

Overparametrization

$(u)_+ := \max(0, u)$ (ReLU activation) denotes the positive part of a scalar

Let $y \in \pm 1$ denote training labels

Two layer scalar output ReLU network:

$$f(x) = \sum_{j=1}^{m} w_j^{(2)} (x^T w_j^{(1)})_+$$

The equation $f(x) = y$ (zero training loss) can have multiple solutions

The numerical optimizer (gradient descent, stochastic gradient, momentum etc) may have an inductive bias

Limits and challenges of deep learning

deep learning models

- often provide the best performance due to their large capacity
- $\rightarrow$ **challenging to train**

Limits and challenges of deep 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**

Limits and challenges of deep learning

deep learning models

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

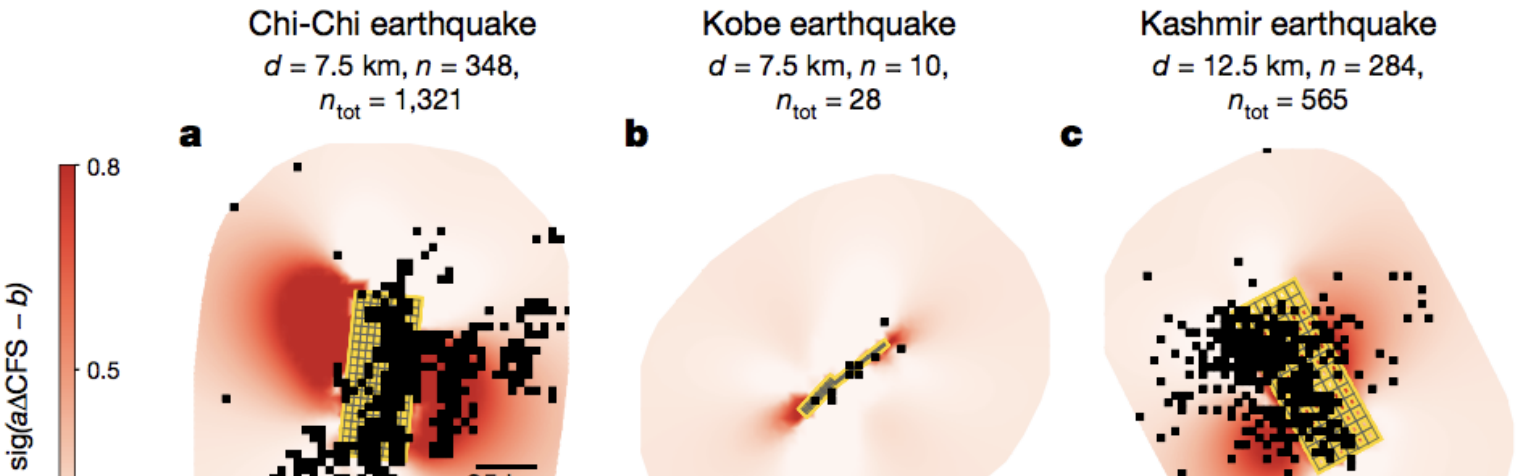
Letter | Published: 29 August 2018

Deep learning of aftershock patterns following large earthquakes

Limits and challenges of deep learning

deep learning models

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



Limits and challenges of deep learning

deep learning models

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

one year later, another paper

nature

Matters Arising | Published: 02 October 2019

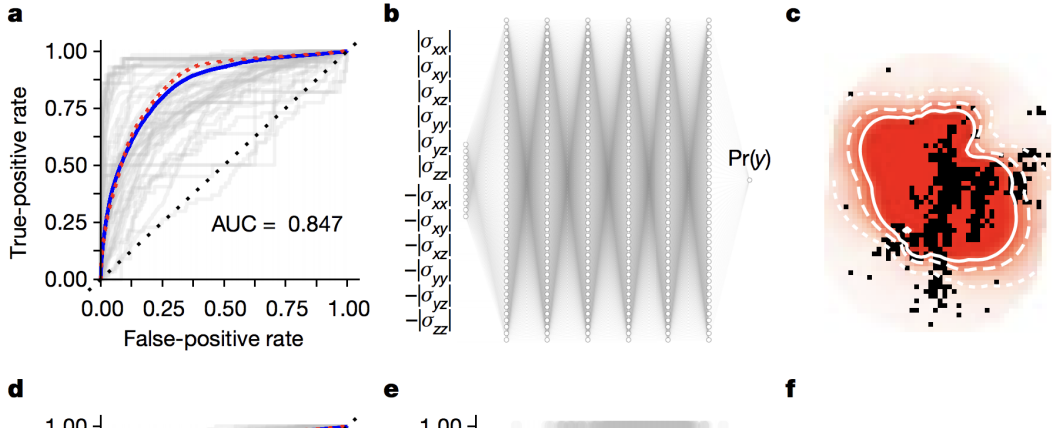
One neuron versus deep learning in aftershock prediction

Limits and challenges of deep learning

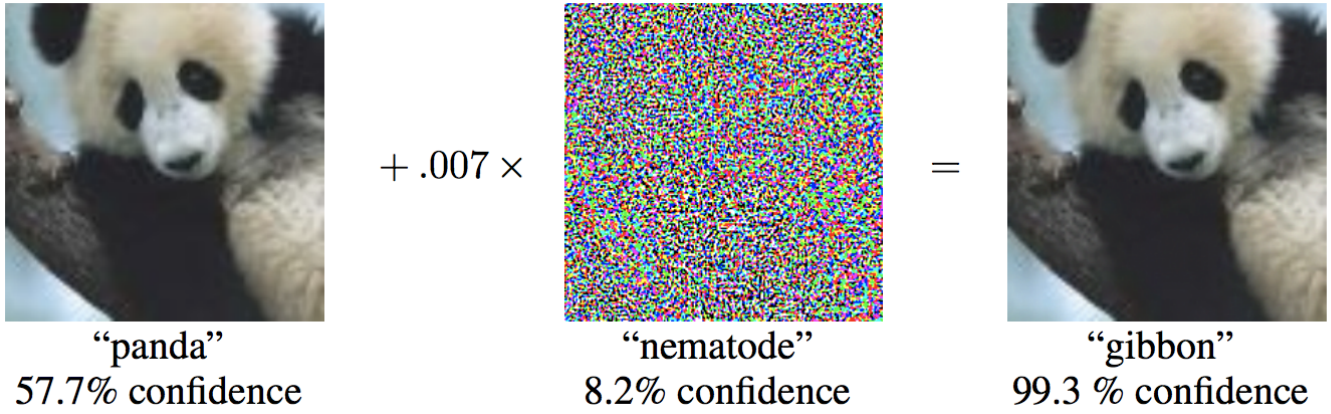
deep learning models

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

logistic regression (1 layer) has the same performance as the 6



Adversarial Examples



adversarial examples, Szegedy et al., 2014, Goodfellow et al., 2015

stop sign recognized as speed limit sign, Evtimov et al, 2017

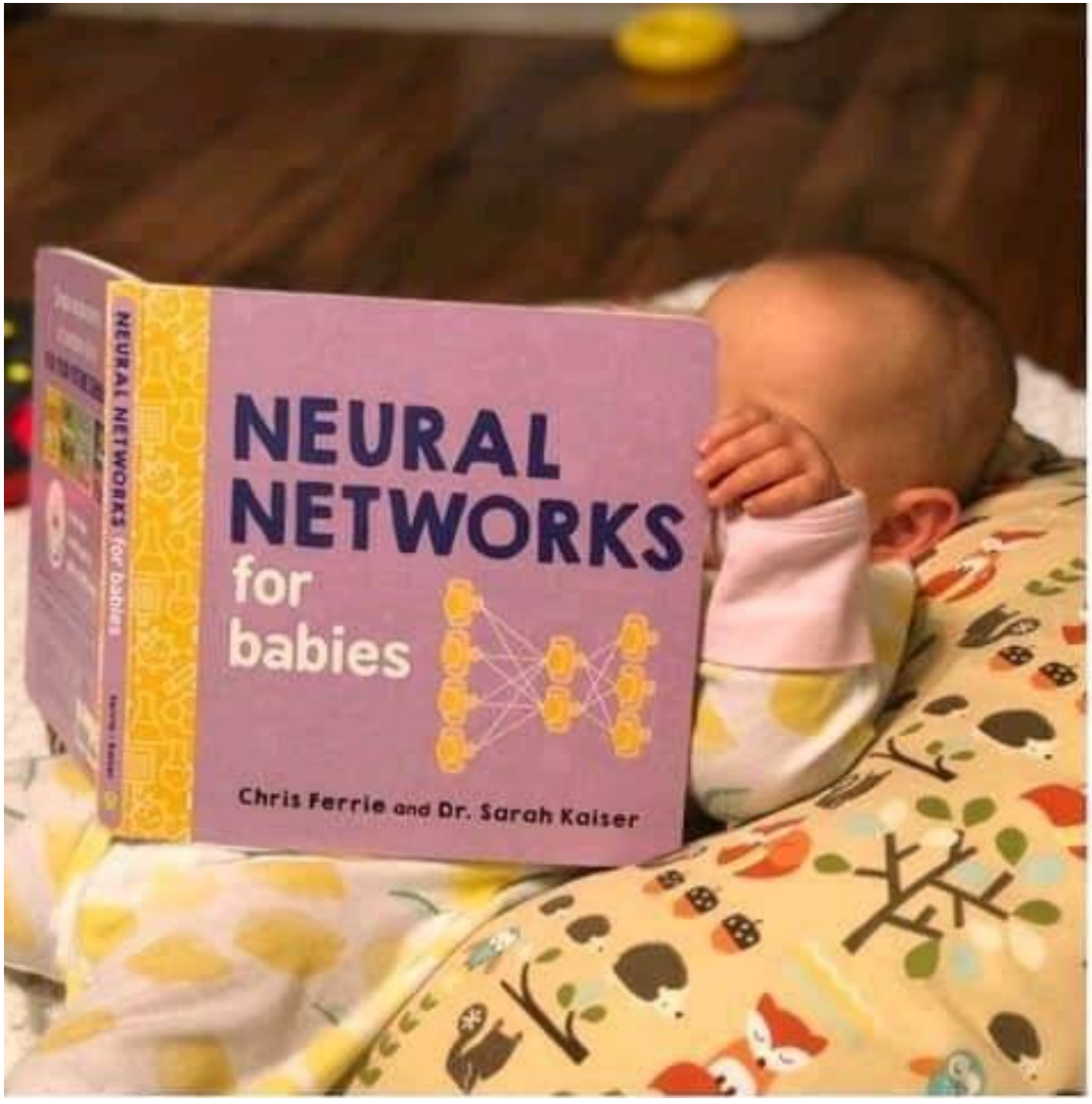
Limitations of Neural Networks

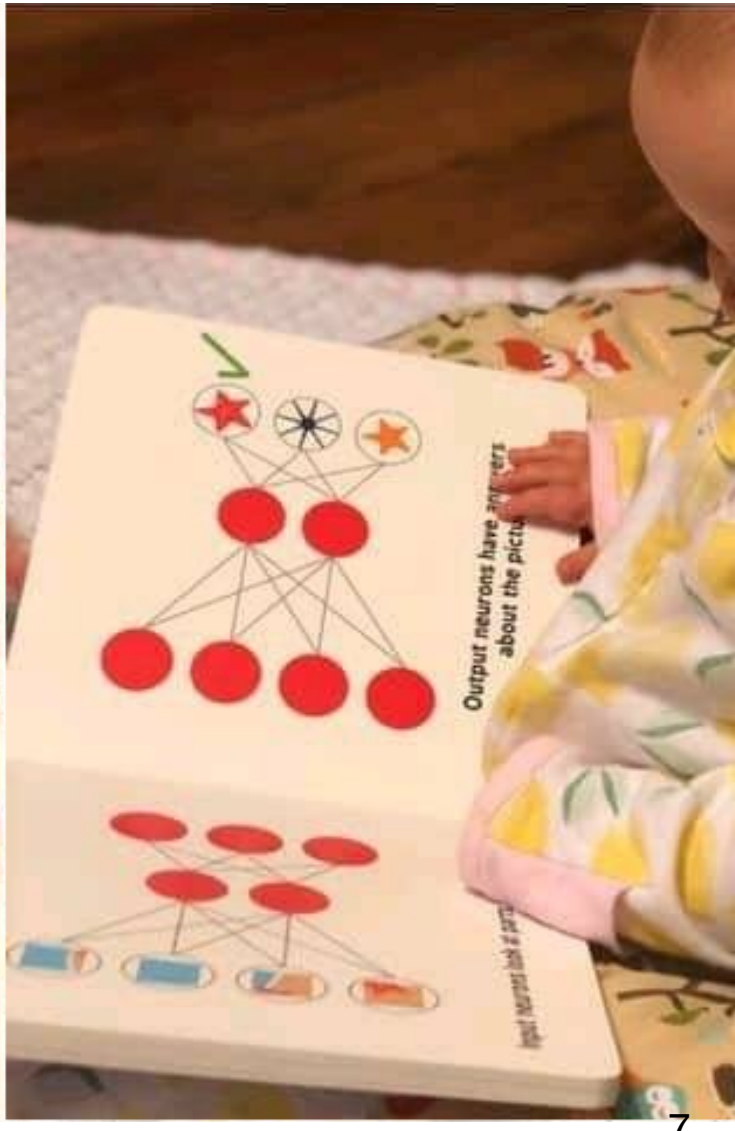
How can we understand what neural network models are learning? Are they automatically finding the best features?

Can we make feature extraction differentiable and expect it to work?

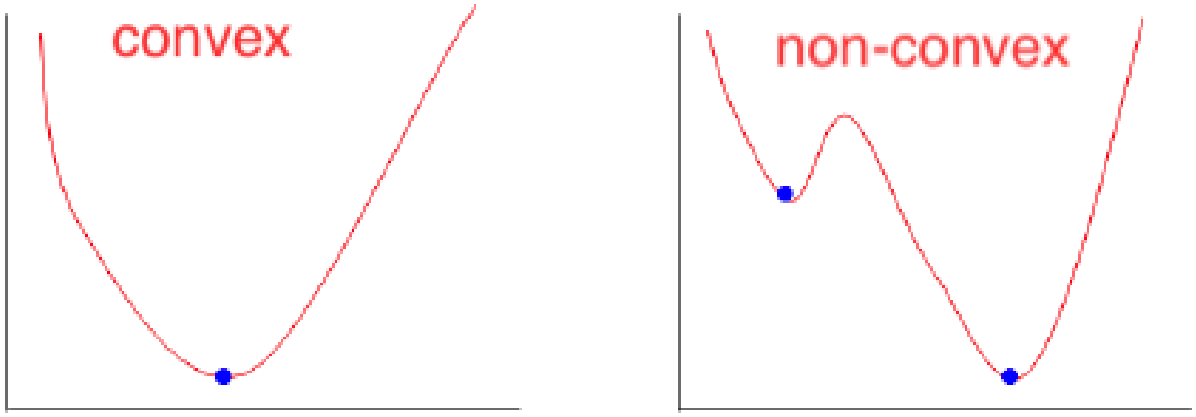
Can we replace domain knowledge with neural networks and data?

How neural networks work?





How neural networks work?



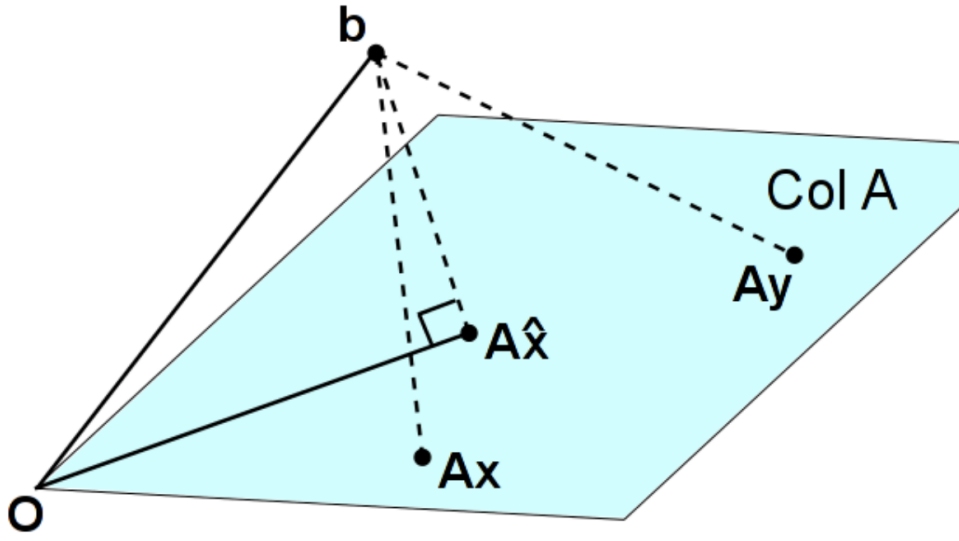
Least-Squares, Logistic Regression, Support Vector Machines etc. are

understood extremely well

Insightful theorems for neural networks?

Least Squares

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

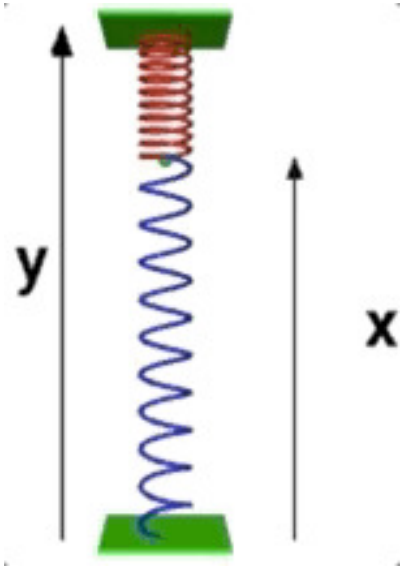


L2 regularization: mechanical model

$$\min_x \quad \underbrace{\frac{1}{2}(x-y)^2}_{\text{elastic energy}} \quad + \quad \underbrace{\frac{1}{2}\lambda x^2}_{\text{elastic energy}}$$

red spring constant $=1$

blue spring constant $= \lambda$

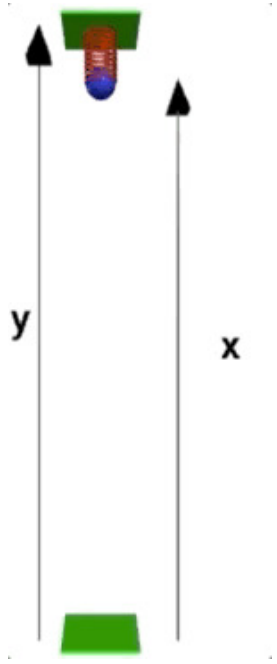


L1 regularization: mechanical model

$$\min_x \quad \underbrace{\frac{1}{2}(x-y)^2}_{\text{elastic energy}} \quad + \quad \underbrace{\lambda|x|}_{\text{potential energy}}$$

red spring constant =1

blue ball mass = λ (small)

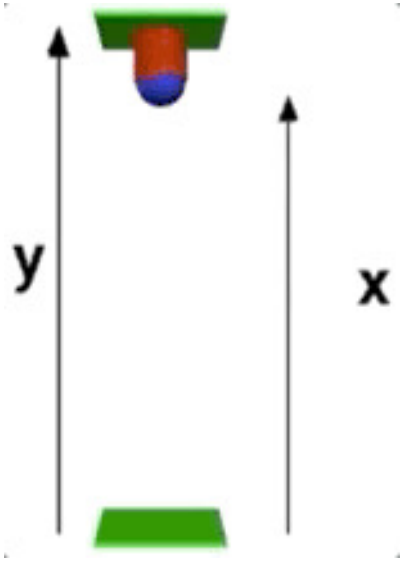


L1 regularization: mechanical model with large λ

$$\min_x \quad \underbrace{\frac{1}{2}(x-y)^2}_{\text{elastic energy}} \quad + \quad \underbrace{\lambda|x|}_{\text{potential energy}}$$

red spring constant $=1$

blue ball mass $= \lambda$ (large)



Least Squares with L1 regularization

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

L1 norm $\|x\|_1 = \sum_{i=1}^{d} |x_i|$

encourages solution x^* to be sparse

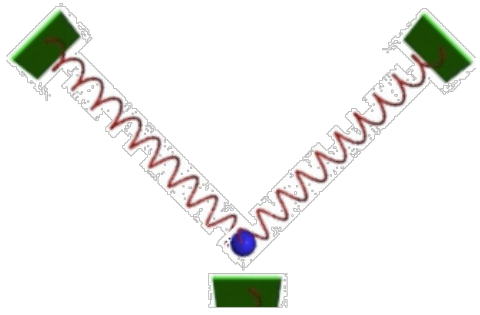
Least Squares with group L1 regularization

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

$\|x_i\|_2 = \sqrt{\sum_{j=1}^{d} x_{ij}^2}$

encourages solution x^* to be group sparse, i.e., most blocks x_i are zero

convex optimization and convex regularization methods are well understood



Neural Networks: one-dimensional signals

n scalar data samples $(d = 1)$

interpolation

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

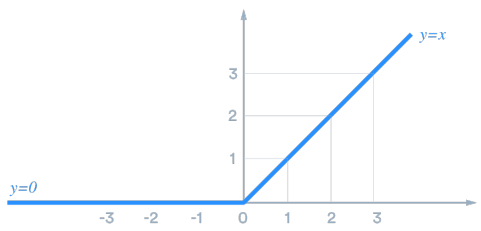
two-layer neural network with m neurons

$f(x) = W_2\phi(W_1 x)$

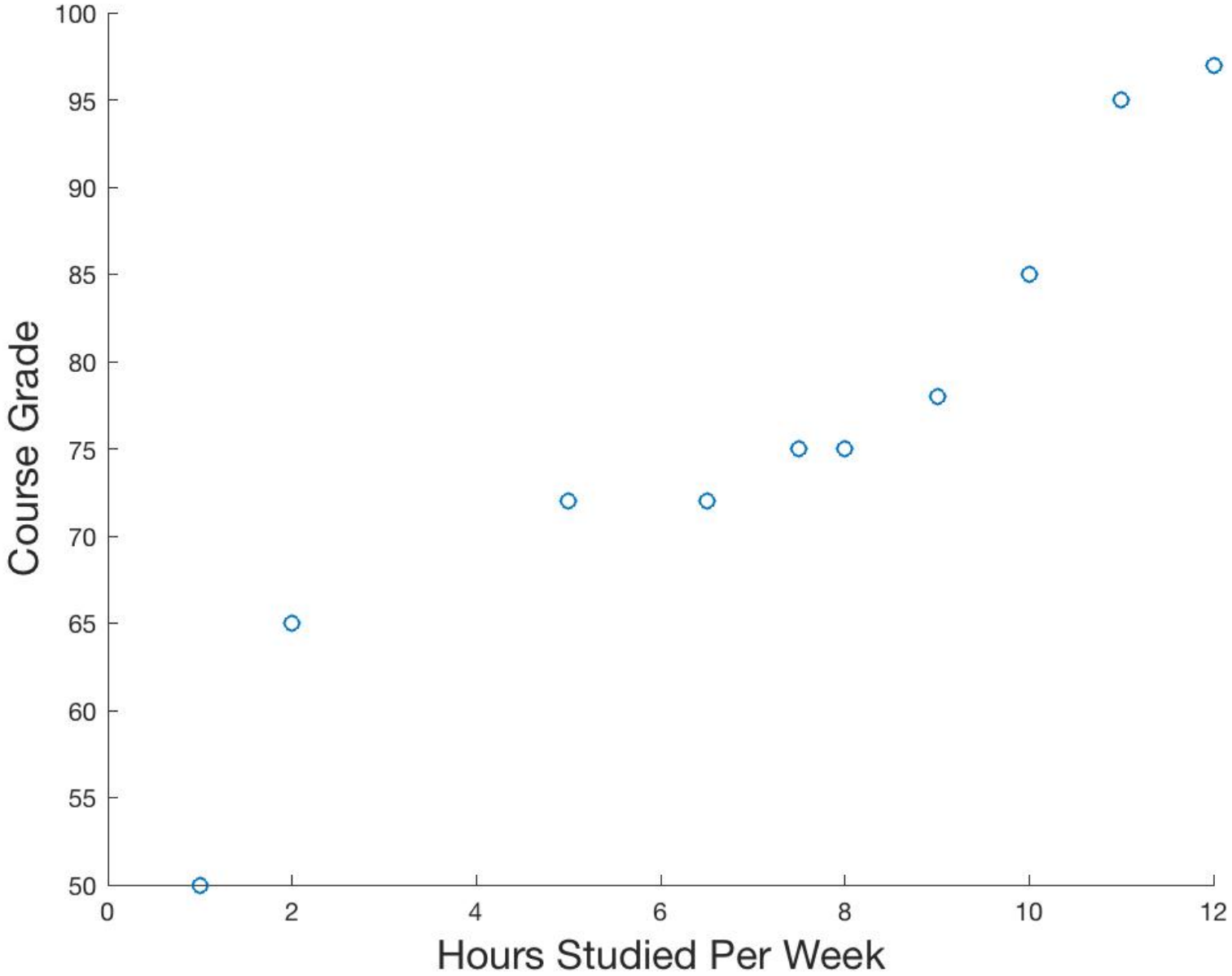
$W_1 \in \mathbb{R}^{d\times m}, W_2 \in \mathbb{R}^{m\times 1}$

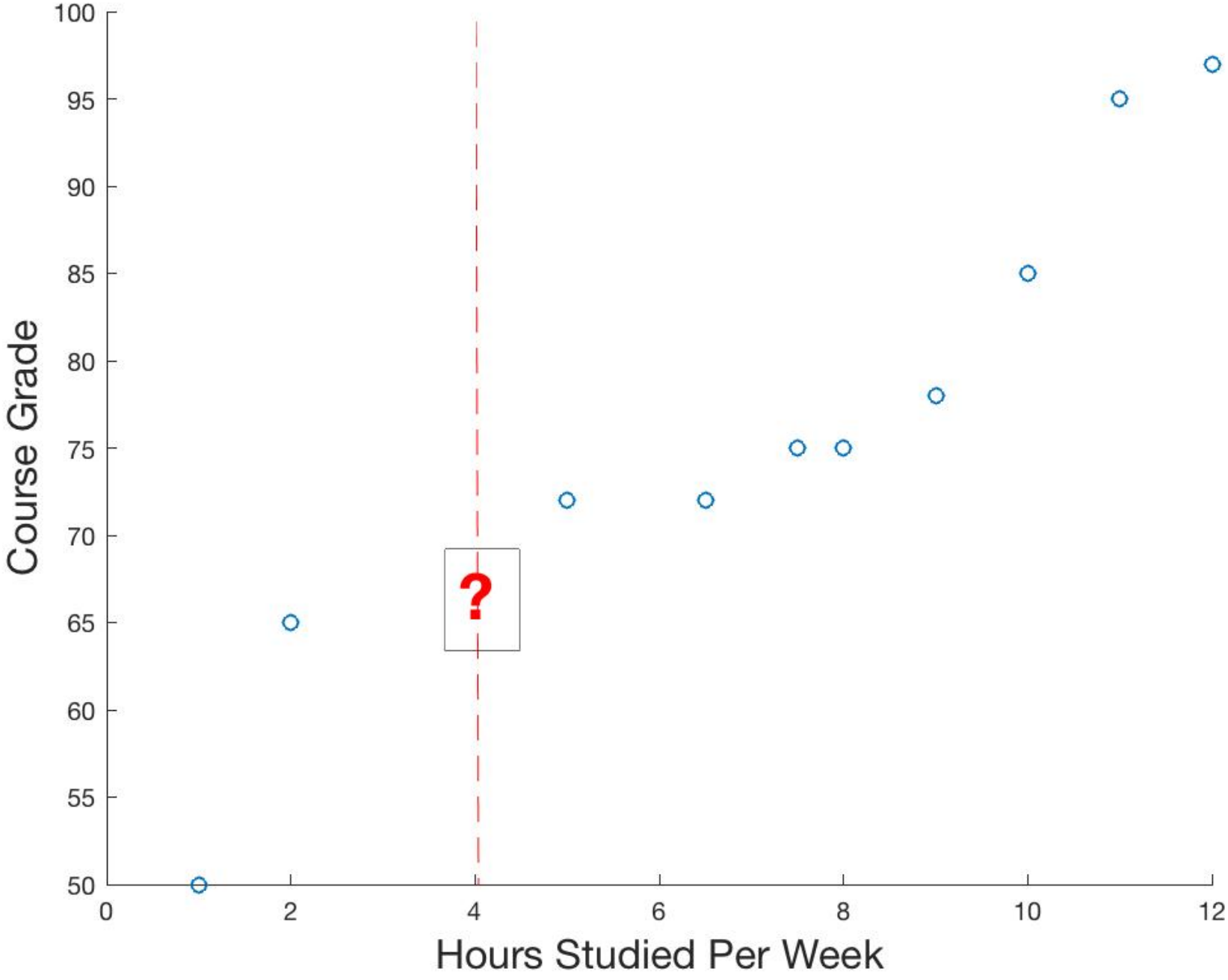
ϕ is the Rectified Linear Unit (ReLU) activation

$\text{ReLU}(u) = \max(0, u)$

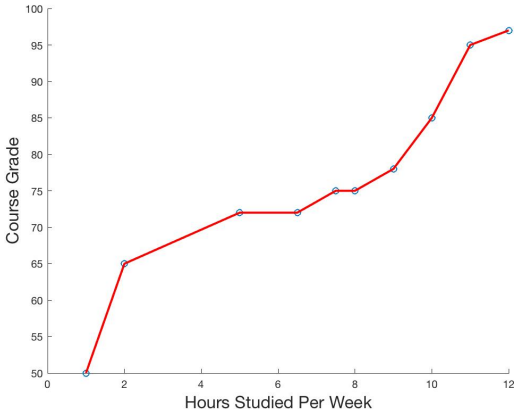


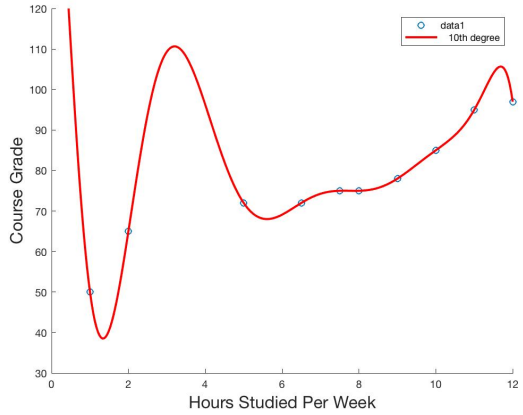
One dimensional signal: course grades

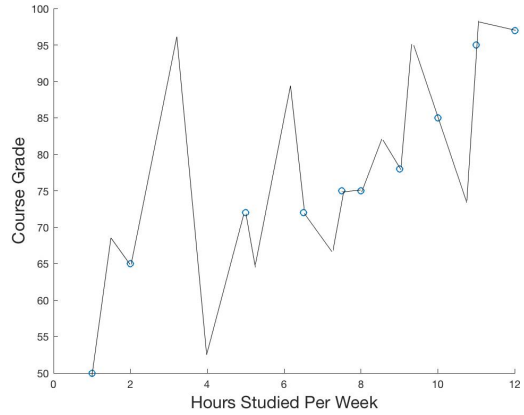



Course Grade
Hours Studied Per Week
?
100
95
90
85
80
75
70
65
60
55
50
0
2
4
6
8
10
12

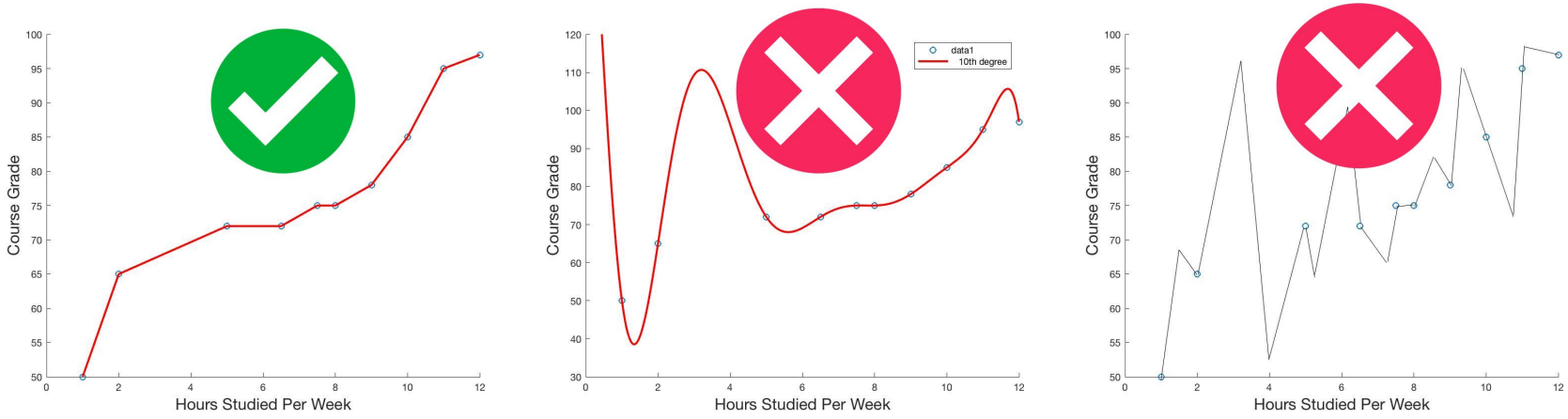
Many signal models result in zero training loss







Training a ReLU neural network with depth ≥ 2 results in zero training loss



interpolation is **piecewise linear** for ReLU networks of depth $2, 3, ...$

this can be formally proven using convex duality

identical to Sobolev kernel for one dimensional signals

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

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

where $\phi(u) = \text{ReLU}(u) = \max(0, u)$

Neural Networks are Convex Regularizers

$$
\begin{aligned}
p_{\text{non-convex}} := \quad & \text{minimize} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right) \\
& W_1 \in \mathbb{R}^{d\times m} \\
& W_2 \in \mathbb{R}^{m\times 1} \\
p_{\text{convex}} := \quad & \text{minimize} \quad L\left(Z, y\right) + \lambda \underbrace{R(Z)}_{\substack{\text{convex regularization}}} \\
& Z \in \mathcal{K} \subseteq \mathbb{R}^{d\times p}
\end{aligned}
$$

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

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

Theorem $p_{\text{non-convex}} = p_{\text{convex}}$, and an optimal solution to $p_{\text{non-co}}$ 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

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

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

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

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

$D_1, ..., D_p$ 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^* as $W_{1j}^* = \frac{u_i^*}{\sqrt{\|u_i^*\|_2}}$, $W_{2j} = \sqrt{\|u_i^*\|_2}$ or $W_{1j}^* = \frac{v_i^*}{\sqrt{\|v_i^*\|_2}}$, $W_{2j} = -\sqrt{\|v_i^*\|_2}$.

Regularization path

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

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

Theorem

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

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

non-convex NN models actually correspond to regularized convex models

Deriving the convex program: Scaling

The weight-decay (ℓ_2^2) regularized non-convex program is equivalent to an ℓ_1 penalized non-convex program

Theorem:

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

$$= \min_{\|W_{1_j}\|_2 = 1, W_2} \left\| \sum_{j=1}^{m} \phi(XW_{1j}) W_{2j} - y \right\|_2^2 + \lambda \sum_{j=1}^{m} |W_{2j}|$$

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

we define

$$\tilde{W}_{1j} := W_{1j} / \alpha_j$$
$$\tilde{W}_{2j} := W_{2j} \alpha_j$$

neural network output does not change

regularization term changes

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

we define

$$\tilde{W}_{1j} := W_{1j}/\alpha_j$$
$$\tilde{W}_{2j} := W_{2j}\alpha_j$$

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

$$p^* = \min_{\substack{\|u_j\|_2 \leq 1 \\ \forall j \in [m]}} \min_{\{\alpha_j\}_{j=1}^m} \frac{1}{2} \left\| \sum_{j=1}^m (X u_j)_+ \alpha_j - y \right\|_2^2 + \beta \sum_{j=1}^m |\alpha_j|$$

Replacing the inner minimization problem with its convex dual, we obtain

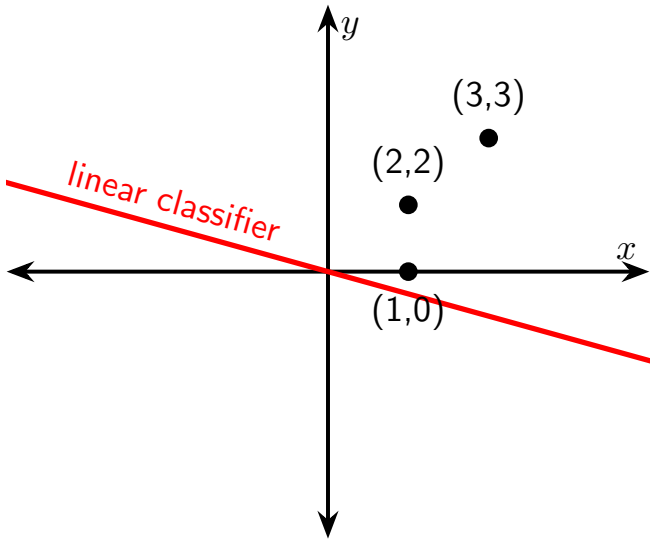
$$p^* = \min_{\substack{\|u_j\|_2 \le 1 \\ \forall j \in [m]}} \max_{\substack{v \in \mathbb{R}^n \text{ s.t.} \\ |v^T (X u_j)_+| \le \beta, \forall j \in [m]}} -\frac{1}{2}\|y - v\|_2^2 + \frac{1}{2}\|y\|_2^2 .$$

Interchanging the order of $\min$ and $\max$, we obtain the lower-bound d^* via weak duality

$$p^* \ge d^* := \max_{\substack{v \in \mathbb{R}^n \text{ s.t.} \\ |v^T (X u)_+| \le \beta \, \forall u \in \mathcal{B}_2}} -\frac{1}{2}\|y - v\|_2^2 + \frac{1}{2}\|y\|_2^2 . \quad (4)$$

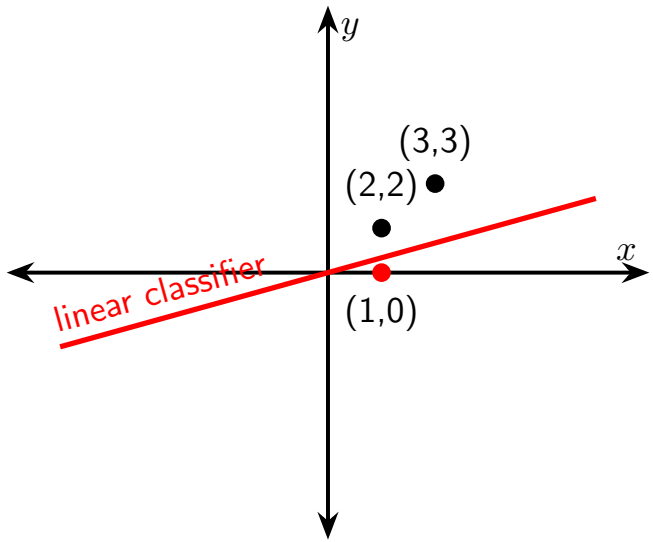
The above problem is a convex *semi-infinite* optimization problem with n variables and infinitely many constraints.

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



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

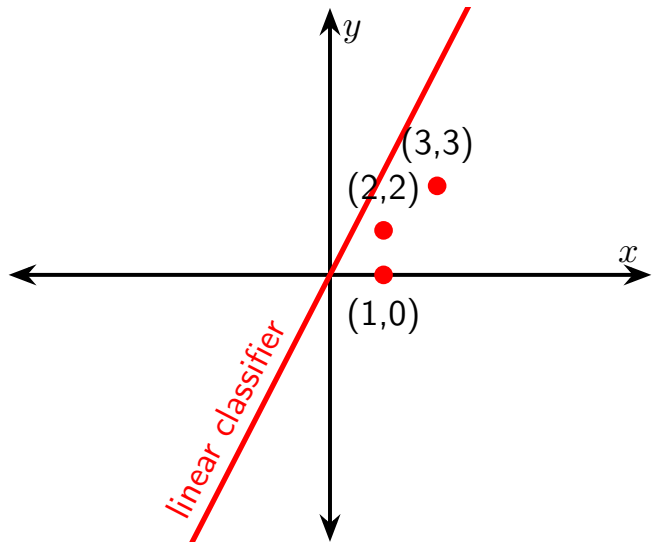
$n = 3$ samples in $\mathbb{R}^d$, $d = 2$ $\quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$



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

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

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

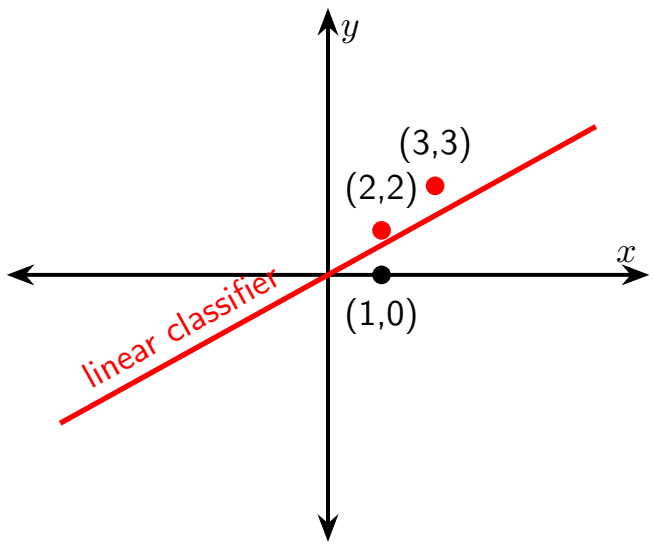


$$D_1 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}$$

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

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

$n = 3$ 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

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

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

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

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

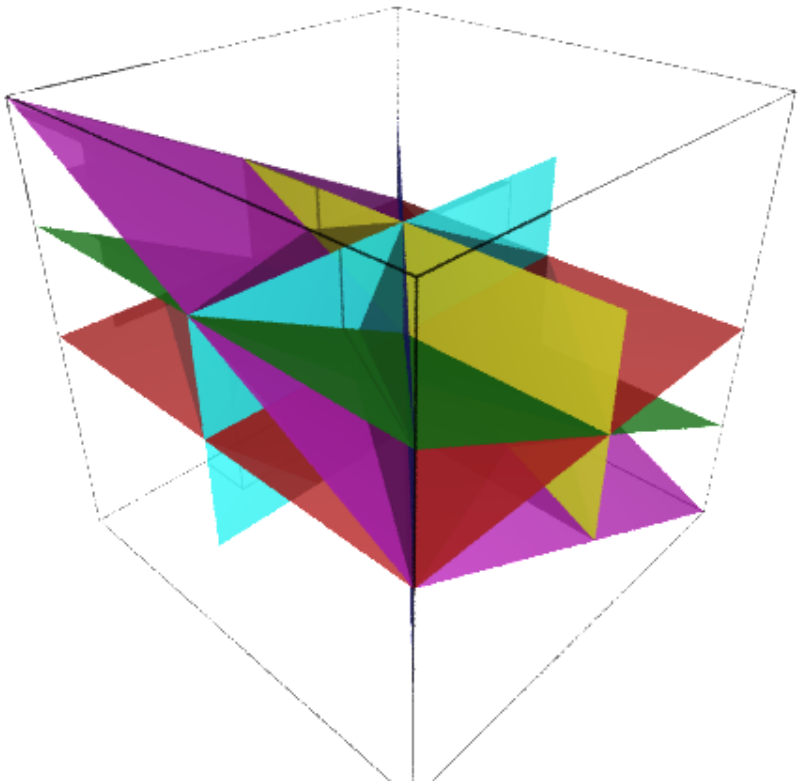
equivalent to the non-convex two-layer NN problem

Hyperplane Arrangements

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

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

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

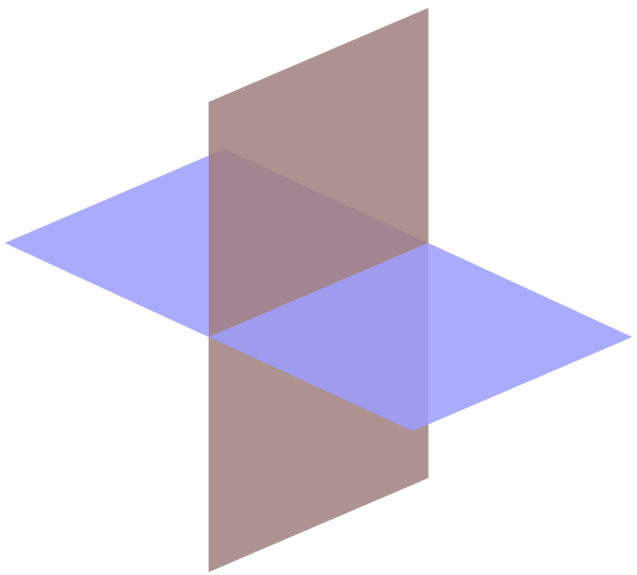


Convolutional Hyperplane Arrangements

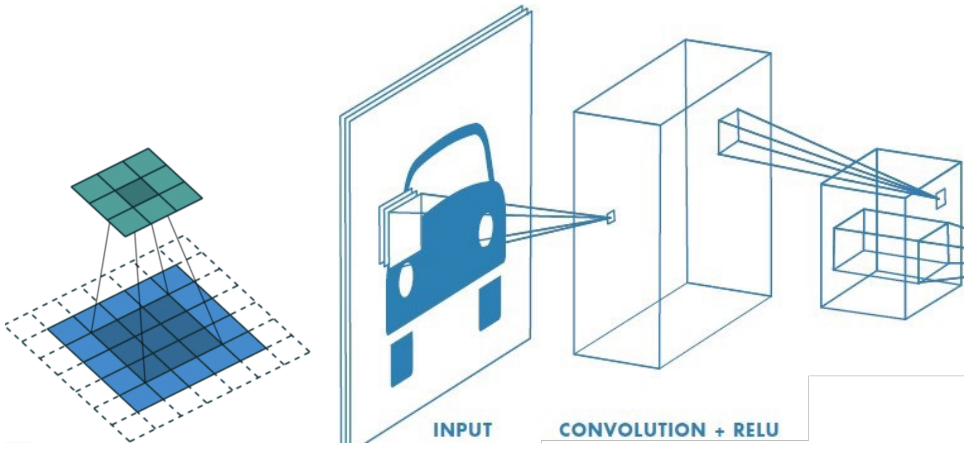
Let $X \in \mathbb{R}^{n \times d}$ be partitioned into patch matrices $X = [X_1, ..., X_K]$ where $X_k \in \mathbb{R}^{n \times h}$

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

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

Two layer CNN with pooling: Conv-Pooling-Relu-FC is equivalent to ℓ_1 penalty,
i.e., constrained Lasso $\min_{w\in\mathcal{K}} \|\Phi w - y\|_2^2 + \lambda\|w\|_1$

$$p_2^* = \min_{\substack{\{u_j, w_{1j}, w_{2j}\}_{j=1}^m \\ u_j \in \mathcal{B}_2, \forall j}} \frac{1}{2}\left\|\sum_{j=1}^m (\mathbf{X}U_j w_{1j})_+ w_{2j} - y\right\|_2^2 + \frac{\beta}{2}\sum_{j=1}^m \left(\|w_{1j}\|_2^2 + w_{2j}^2\right)$$

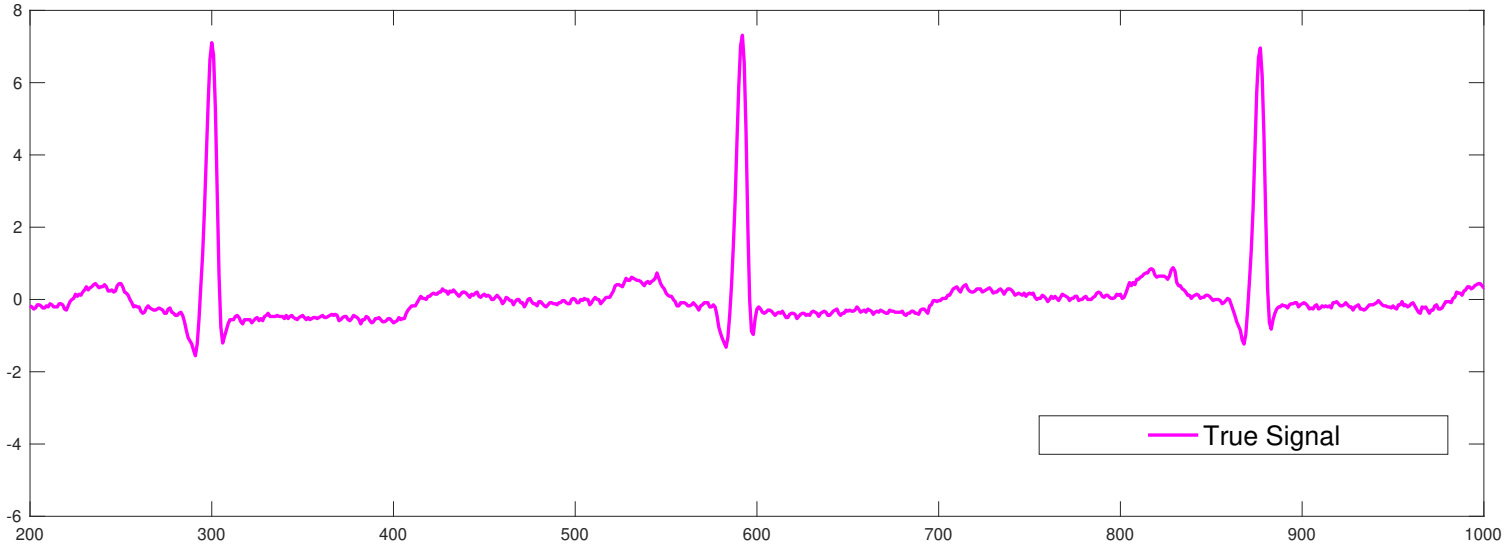
Theorem

Let $\tilde{\mathbf{X}} = \mathbf{X}F$ *and* $F \in \mathbb{C}^{d\times d}$ *be the DFT matrix. The equivalent convex problem is*

$$\min_{\substack{\{w_i, w_i'\}_{i=1}^p \\ w_i, w_i' \in \mathbb{C}^d, \forall i}} \frac{1}{2}\left\|\sum_{i=1}^p \mathit{diag}(S_i)\tilde{\mathbf{X}}\left(w_i' - w_i\right) - y\right\|_2^2 + \frac{\beta}{\sqrt{d}}\sum_{i=1}^p \left(\|w_i\|_1 + \|w_i'\|_1\right)$$

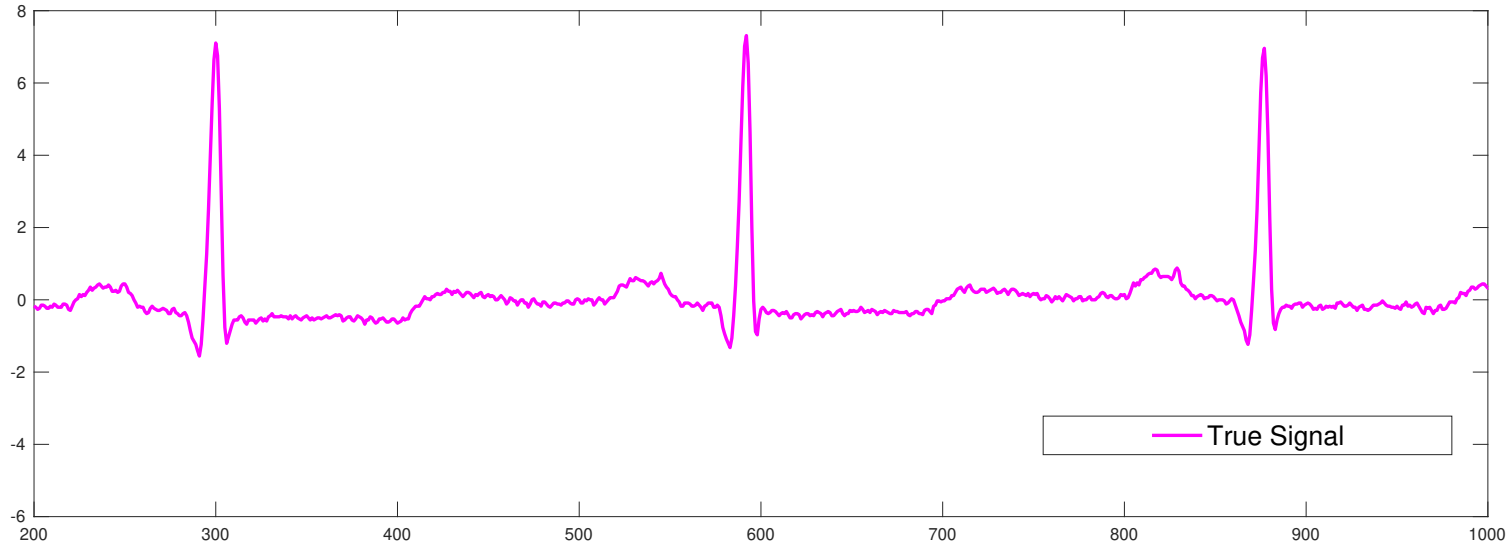
$$\text{s.t. } (2\mathit{diag}(S_i) - I_n)\tilde{\mathbf{X}}w_i \geq 0,\ (2\mathit{diag}(S_i) - I_n)\tilde{\mathbf{X}}w_i' \geq 0, \forall i,$$

Signal Prediction: Electrocardiogram (ECG)



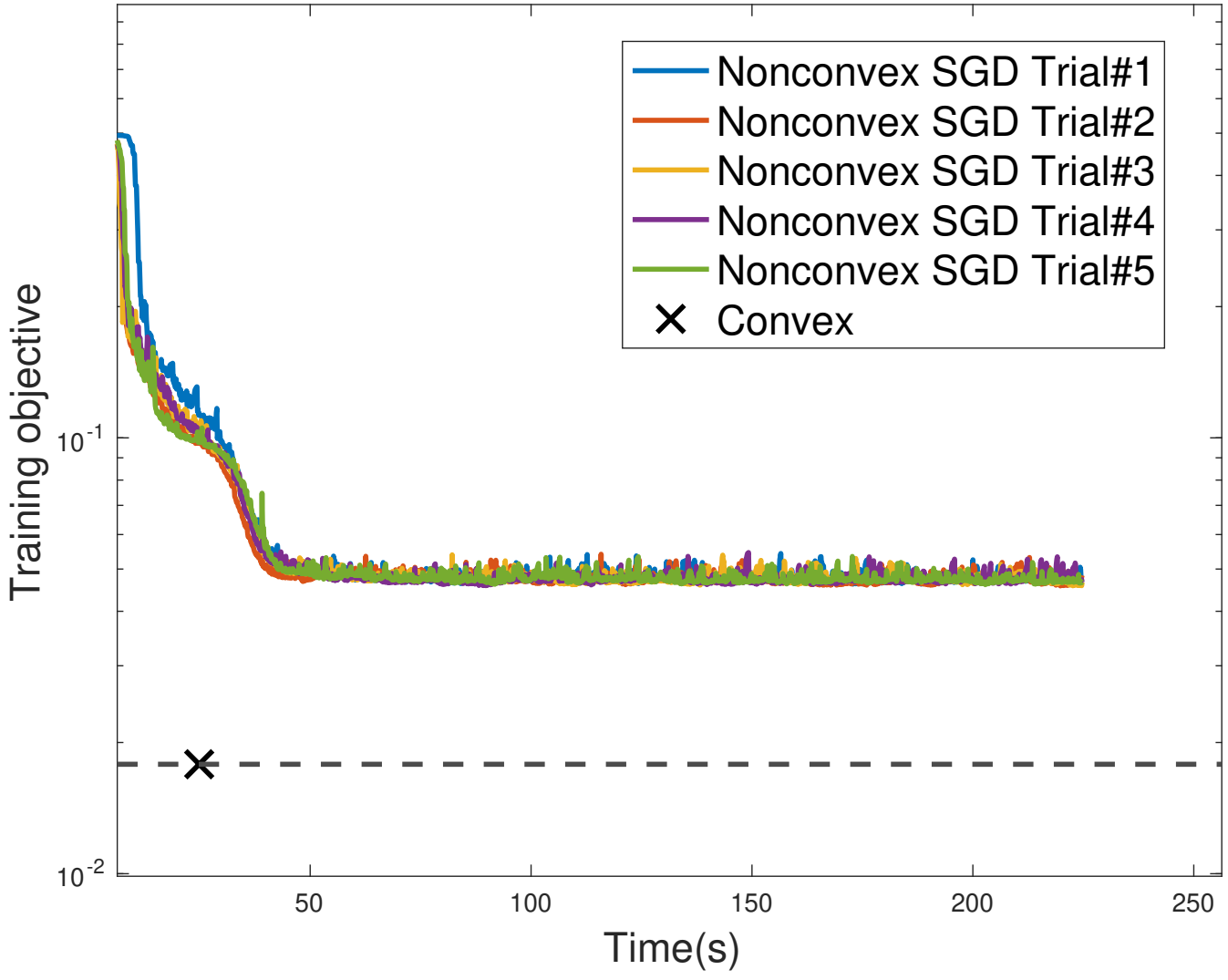
window size: 15 samples

training and test set

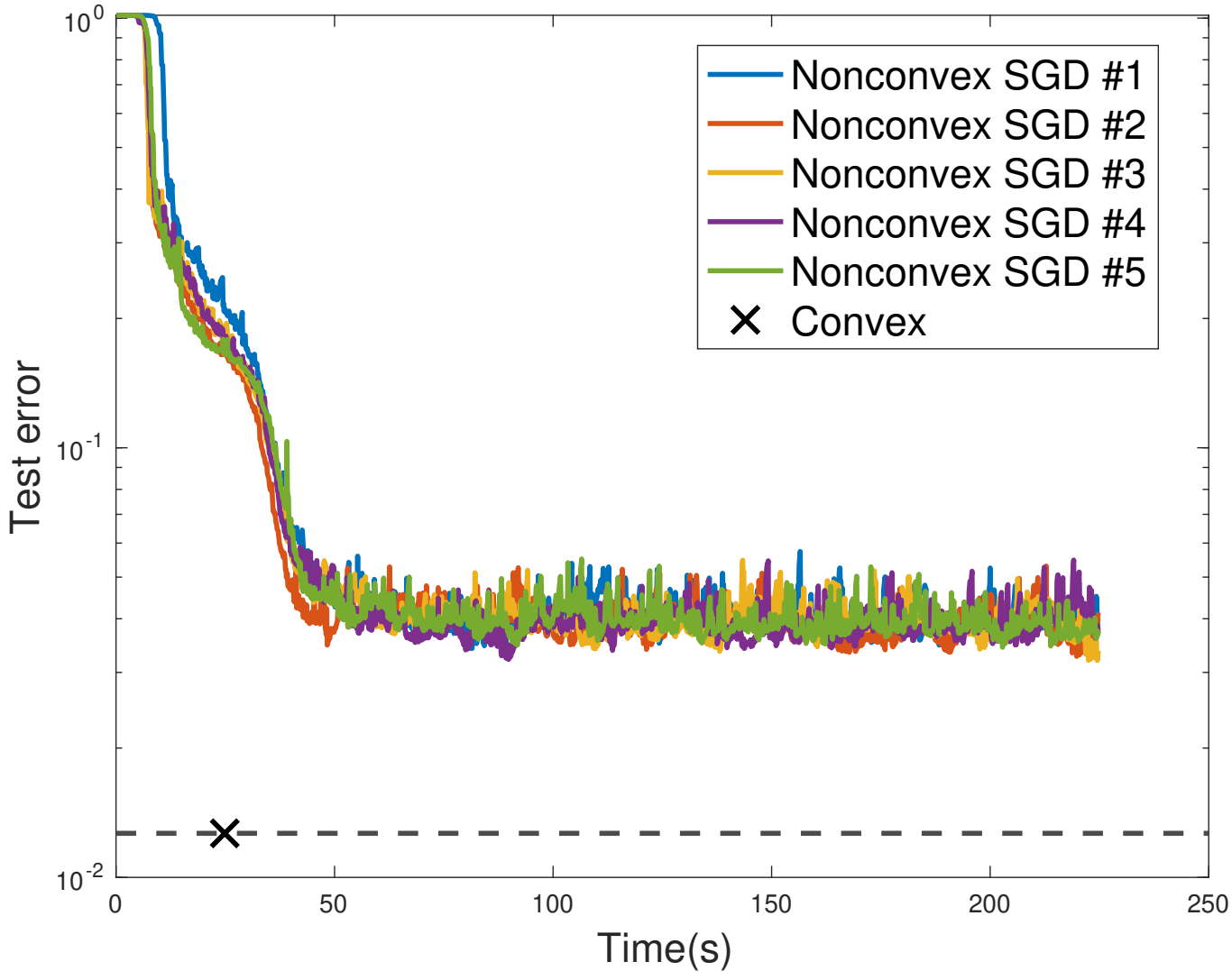


$$X = \begin{bmatrix} x[1] & \dots & x[d] \\ x[2] & \dots & x[d+1] \\ \vdots & & \\ x[n] & \dots & x[d+n-1] \end{bmatrix}, \quad y = \begin{bmatrix} x[d+1] \\ x[d+2] \\ \vdots \\ x[d+n] \end{bmatrix}$$

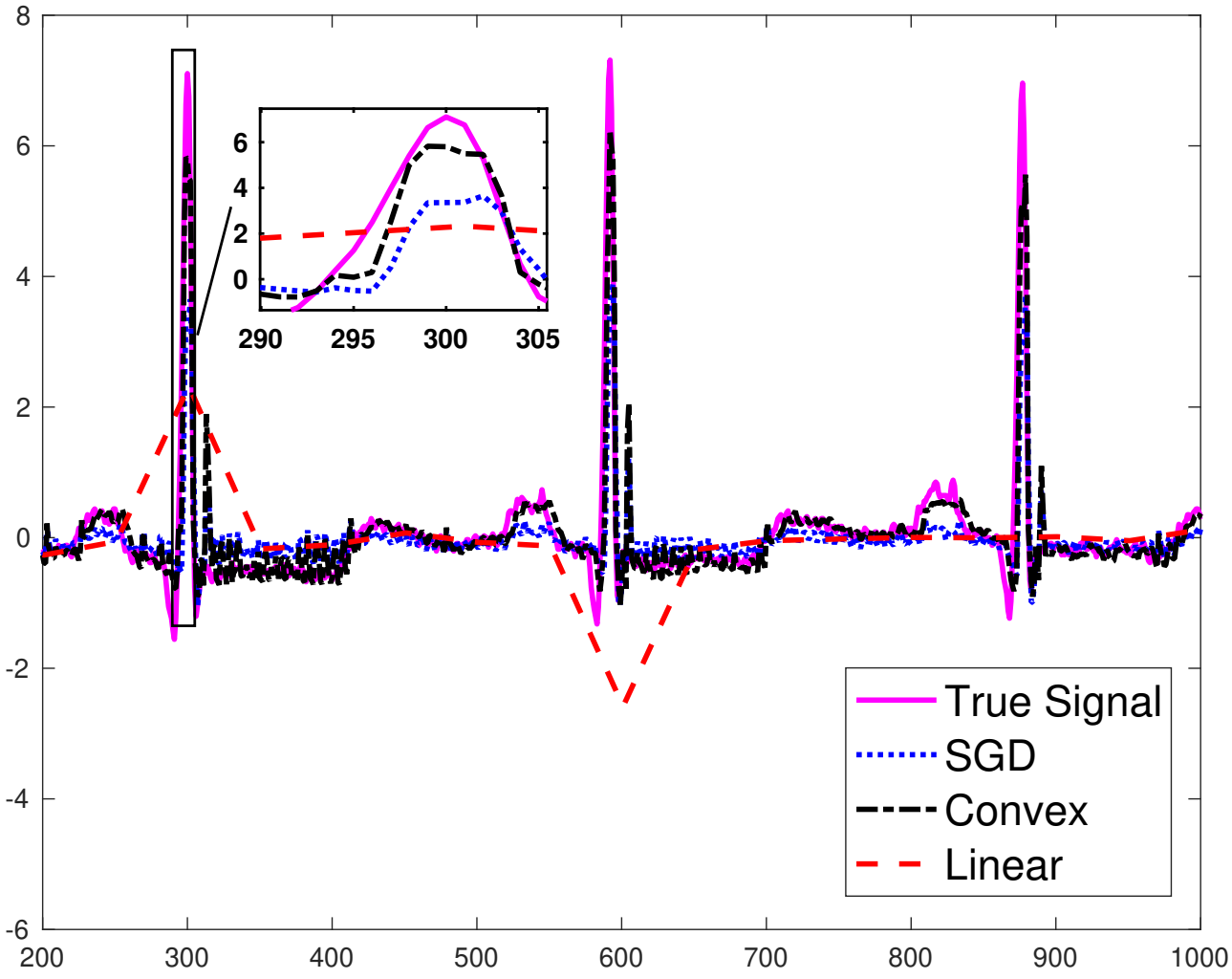
Signal Prediction: Training



Signal Prediction: Test Accuracy

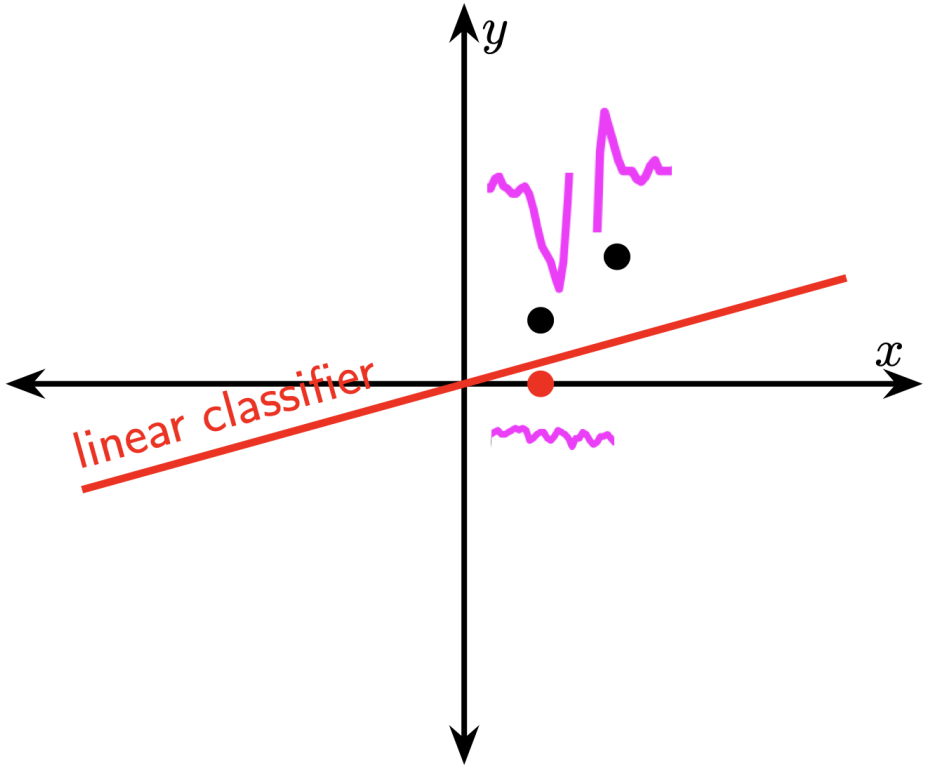


Signal Prediction: Test Accuracy



Neural Networks are Fully Explainable

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



Approximating the Convex Program

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

Sample $D_1, ..., D_p$ as Diag$(Xu \geq 0)$ where $u \sim N(0, I)$

$(1 + \frac{\sigma_{r+1}}{\lambda})$ 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