Stat 205: Introduction to Nonparametric Statistics

Lecture 13 : Single Hidden-Layer Neural Nets

Instructor David Donoho; TA: Yu Wang

The images in this lecture are scraped from Google Images. Many similar images are available. The intent is merely to make the lecture more vivid by providing 'eye candy'. No attempt is made to identify all sources.

Some Background Reading

- Emmanuel Candés.
  Ridgelets Theory and Applications 1996.
- Ballestriero and Baraniuk(2018),
  Mad Max: Affine Spline Insights into Deep Learning
- Savarese, Evron, Soudry, Srebro (2018):
  How do infinite-width networks look in function space?
- Ballestriero and Baraniuk (2018)
  A Spline Theory of Deep Networks
- Ergen and Pilanci (2020):
  Convex Geometry of Two-Layer ReLu network
- Ergen and Pilanci (2021):
  Revealing the structure of deep networks by convex duality.

Some Recent History

- 1950's: Perceptrons
- 1980's: Deep Neural Networks experiments { eg Hinton, LeCun etc }
- 1990's: Digit Recognition {MNIST, LeCun}
- 2000's: Deep Nets Winter
- 2012: Imagenet & rebirth
- 2013: Google massive investment
- 2013-2022: tens of billion of payroll and hardware investment

Modern Neural Nets Terminology, 1

- Inputs
  $x \in \mathbf{R}^d$ eg an image.
- Layers
  - Intermediate results: $x \mapsto h^1 \mapsto h^2 \mapsto \ldots \mapsto h^L$
  - Activations:

$$h^1 \in \mathbf{R}^{d_1}, \ldots, h^\ell \in \mathbf{R}^{d_\ell}, \ldots, h^L \in \mathbf{R}^{d_L}.$$

  - $\ell = 1$: first layer; $\ell = L$: last layer.
- Outputs
  Regression $f_h(x) = \sum_j w_j^L h_j^L$;
  Classification $f_h(x) = \text{argmax}_{c=1}^C h_j^L$.

Modern Neural Nets Terminology, 2

- Weights $W^\ell = (W^\ell_{i,j})$ where each W^ℓ is $d_{\ell-1} \times d_\ell$.
- Nonlinearity

$$\begin{aligned}\mathrm{relu}(x) &= (x)_+ \\ &= \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}\end{aligned}$$

- Preactivations

$$z^\ell = h^{\ell-1} W^\ell; \text{ meaning } z^\ell_j = \sum_i W^\ell_{i,j} h^{\ell-1}_i$$

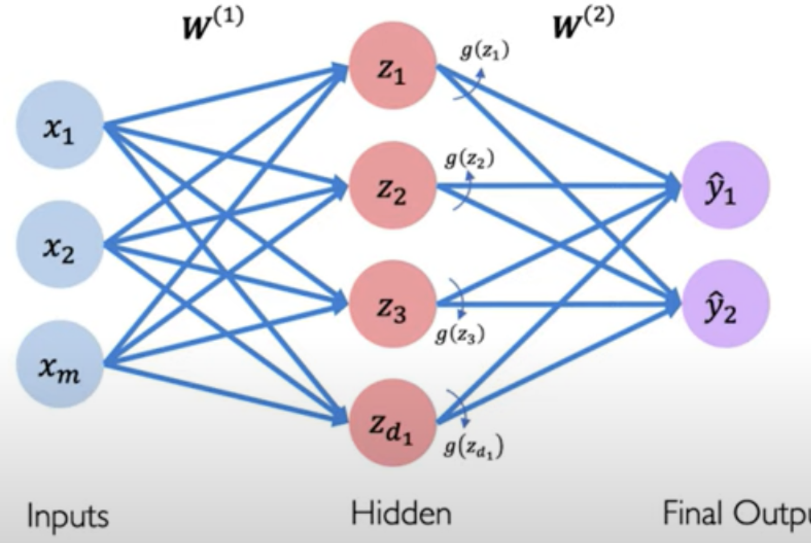
- Activations

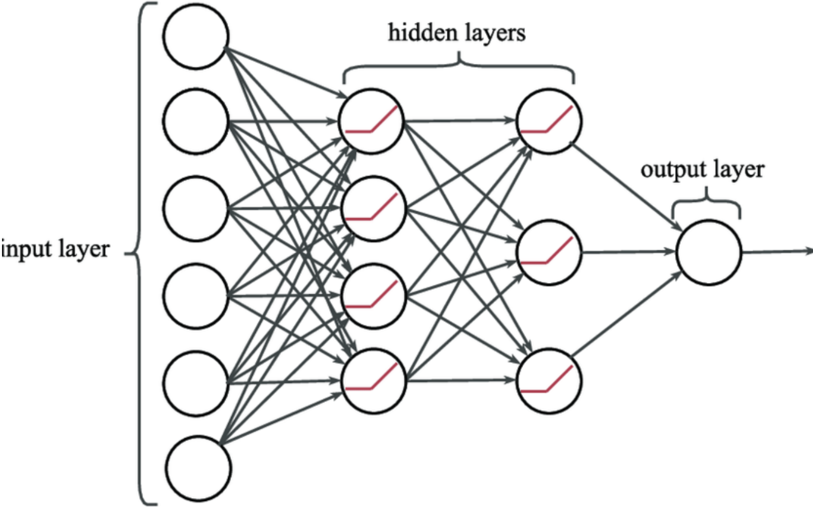
$$h^\ell = \mathrm{relu}(z^\ell - b^\ell); \text{ meaning } h^\ell_j = \mathrm{relu}([\sum_i W^\ell_{i,j} h^{\ell-1}_i] - b^\ell_j).$$

- Biases:

$$\mathrm{relu}(x - b) = (x - b)_+$$

b is the location of a knot or 'kink' in the relu:


$W^{(1)}$
$W^{(2)}$
x_1
x_2
x_m
z_1
z_2
z_3
z_{d_1}
$g(z_1)$
$g(z_2)$
$g(z_3)$
$g(z_{d_1})$
$\hat{y}_1$
$\hat{y}_2$
Inputs
Hidden


hidden layers
input layer
output layer

Single-Hidden Layer, Dimension 1

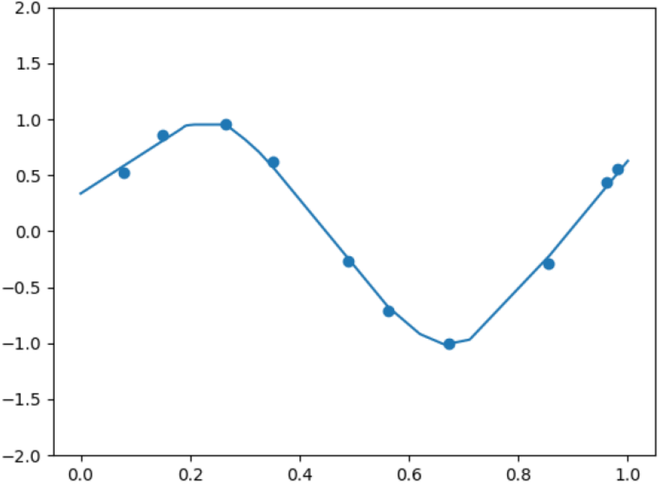
- SIngle-Hidden Layer $L = 2$, arbitrary dimension

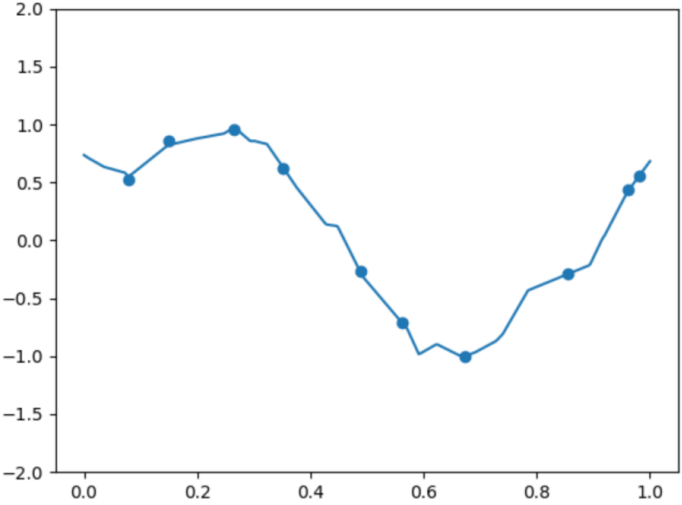
$$f(x) = \sum_j w_j^2 \text{relu}([x.W_{\cdot,j}^1] - b_j)$$

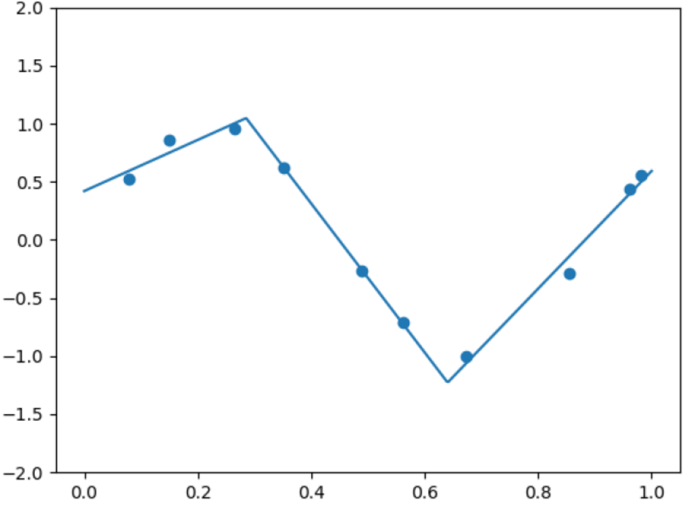
- Simplified notation for dimension 1: i.e. $x \in \mathbb{R}^1$.

$$f(x) = \sum_j c_j \cdot \text{relu}(x - b_j)$$

- **Observation:**
  In the $L = 1$, $d = 1$, regression setting, $f(x)$ is a piecewise linear function on $\mathbf{R}$*, with knots at the* $(b_j)_{j=1}^{d_1}$.


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0

Example

- Example: $x \in \mathbf{R}^1$, $L = 2$

$$x \in [0,1], \qquad b^1_j = j/d_1, \qquad j = 1, \ldots d_1.$$

$L = 2$ simplifies to:

$$f(x) = \sum_j c_j \cdot \mathrm{relu}(x - j/d_1)$$

Namely: linear spline with equipspaced knots.

Infinite-Width Limit

- Width: d_ℓ where each W^ℓ is $d_{\ell-1} \times d_\ell$.
- Infinite Width Limit
$$d_\ell \to \infty.$$
- This limit is not practical/ desired by practitioners; however it allows math analysis and theoretical understanding.

When can approximation by relu work? 1/3

Theorem. *If f is a real-valued function defined on $[0,1]$ with 2 continuous derivatives, it can be approximated by a linear combination of relu's. For example, consider the sequence of approximations with width $d_1^{(k)}$:*

$$f^{(k)}(x) = f(0) + c_0 \cdot \mathrm{relu}(x) + \sum_{j=1}^{d_1^{(k)}} c_j^{(k)} \mathrm{relu}(x - b_j^{(k)})$$

where

$$b_j^{(k)} = j/d_1^k, \quad j = 1, \ldots, d_1^{(k)},$$

and

$$c_j^{(k)} = f''(b_j^{(k)})/d_1^{(k)};$$

and

$$c_0 = f'(0);$$

then in the infinite-width limit $k \to \infty$:

$$f^{(k)}(x) \to f(x).$$

(Discuss)

When can approximation by relu work?, 2/3

$$f(x) - f(0) = \int_0^x f'(t)dt.$$

i.e.

$$f(x) - f(0) = \int_0^1 f'(t)1_{\{t<x\}}dt.$$

Integrate by parts:

Object	Primitive	Derivative
F	$f'(t)$	$f''(t)$
G	$-(x-t)_+$	$1_{\{t<x\}}$

So from

$$\int F(t)dG(t) = FG|_{t=0}^1 - \int G(t)dF(t)$$

we obtain

$$\int_0^1 f'(t)1_{\{t<x\}}dt = f'(1)\cdot 0 - f'(0)\cdot x_+ + \int_0^1 f''(t)(x-t)_+dt$$

Hence:

$$f(x) = f(0) + f'(0)x + \int_0^1 f''(t)(x-t)_+dt.$$

When can approximation by relu work?, 3/3

From

$$f(x) = f(0) + f'(0)x + \int_0^1 f''(t)(x-t)_+ dt.$$

we get

$$\begin{aligned}
f(x) &= f(0) + f'(0)\cdot \mathrm{relu}(x) + \int_0^1 f''(t)\mathrm{relu}(x-t)dt \\
&\approx f(0) + f'(0)\cdot \mathrm{relu}(x) + Ave_j\{f''(b_j^{(k)})\cdot \mathrm{relu}(x - b_j^{(k)}s)\} \\
&= f(0) + c_0^{(k)}\cdot \mathrm{relu}(x) + \sum_{j=1}^{d_1^{(k)}} c_j^{(k)}\mathrm{relu}(x - b_j^{(k)}) \\
&= f^{(k)}(x)
\end{aligned}$$

Infinite-Width Limit

- Width: d_ℓ where each W^ℓ is $d_{\ell-1} \times d_\ell$.
- Infinite Width Limit
$$d_\ell \to \infty.$$
- In this limit, more relu's than we have data points!
- There will be (many!) knots in between data points!
- What will regularization do?

Training Neural Nets Paradigm 1/2

- MSE Loss

$$\mathcal{L}(y, f) = Ave_i(y_i - f(x_i))^2$$

- Regularization

$$P(f) = \text{ differentiable functional of the weights defining } f$$

- Training

$$\hat{f}_\lambda = \text{argmin}\mathcal{L}(y, f) + \lambda P(f).$$

Training Neural Nets Paradigm, 2/2

- Training
$$\hat{f}_\lambda = \text{argmin}\mathcal{L}(y, f) + \lambda P(f).$$
- Examples
  - Weight Decay: traditional parametrization:
  $$P(f) = \sum_{\ell=1}^{L} \sum_j (w_j^{(\ell)})^2$$
  This is often called *weight-decay*.
  - Ridge Regression (simplified 1-d setting):
  $$L_2(f) = \sum_j c_j^2$$
  - Lasso (simplified 1-d setting):
  $$L_1(f) = \sum_j |c_j|$$

Example: L_1 Penalization

Training Problem

$$\hat{f}_\lambda = \text{argmin}\mathcal{L}(y, f) + \lambda L_1(f).$$

$$L_1(f) = \sum_j |c_j|$$

$$f(x) = f(0) + c_0 x + \sum_j c_j \text{relu}(x - b_j)$$

Theorem. *Suppose we have a dataset* $(x_i, y_i), i = 1, \ldots, n,$ *that* $x_1 = 0$ *and that* (b_j) *contains, as a subset, all the data points* x_i:$\{x_i\} \subset \{b_j\}$. *There is a minimizer* f^* *of the above training problem having only the data points as knots, i.e. we may write*

$$f^*(x) = f(0) + \sum_i c_i \text{relu}(x - x_i)$$

Example: L_1 interpolation

Interpolation

$$\min L_1(f) : y_i = f(x_i), i = 1, \ldots, n$$

$$f(x) = f(0) + c_0 x + \sum_j c_j \mathrm{relu}(x - b_j)$$

Theorem. *Suppose we have a dataset $(x_i, y_i), i = 1, \ldots, n$, that $x_1 = 0$ and that (b_j) contains, as a subset, all the data points x_i:$\{x_i\} \subset \{b_j\}$. There is a minimizer f^* of the above training problem having only the data points as knots, i.e. we may write*

$$f^*(x) = f(0) + \sum_i c_i \mathrm{relu}(x - x_i)$$

This minimizer is simply linear interpolation, but other minimizers may exist.

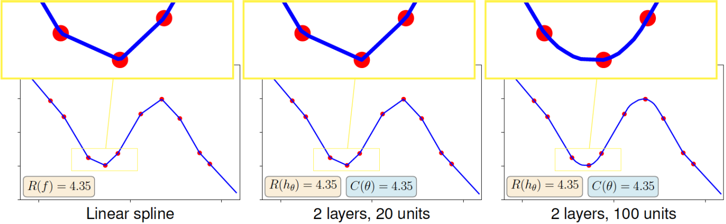


Figure 1: Linear interpolation (**left**) and two trained ReLU networks with 1 hidden layer consisting of 20 (**middle**) and 100 (**right**) units respectively, optimized to perfectly fit a set of 10 points, for which the minimum cost of perfect fitting is $\overline{R}(f^*) = 4.35$. Training was done by minimizing the squared loss with a small regularization of $\lambda = 10^{-5}$. All three functions achieve the optimal cost $\overline{R}(\cdot)$ in function space, and both networks yield optimal cost in parameter space $C(\theta)$. The two networks arrived at different global minima in function space, with the same value of $\overline{R}(f)$. For example, in the area highlighted, changing the derivative gradually instead of abruptly does not effect its total variation, and so also yields an optimal solution.

Example: Ridge Penalization, 1/2

Training Problem

$$\hat{f}_\lambda = \text{argmin} \mathcal{L}(y, f) + \lambda L_2(f).$$

$$L_2(f) = \sum_j |c_j|^2$$

$$f(x) = f(0) + c_0 x + \sum_j c_j \text{relu}(x - b_j)$$

$$\Psi(x) = (1, \text{relu}(x - b_0), \text{relu}(x - b_1), \ldots, \text{relu}(x - b_{d_1})).$$

$$\Psi = (\Psi_{i,j}) = (\Psi(x_i)_j), \qquad i = 1, \ldots, n, \quad j = 1, \ldots, d_1$$

Theorem. *Suppose we have a dataset* $(x_i, y_i), i = 1, \ldots, n,$ *that* $x_1 = 0$*. The minimizer* f^* *of the above training problem has the form*

$$\hat{f}(x) = \Psi(x)(K + \lambda I)^{-1}(\Psi^T y)$$

Example: Ridge Penalization 2/2

For $k = 1, 2, \ldots,$ let $d^{(k)} = d_1^{(k)}$. Consider the partial feature map $\Phi^{(k)}$ induced by equispaced biases $b_j^{(k)} = j/d^{(k)}$, $j = 0, \ldots, d^{(k)} - 1$.

$$\Phi^{(k)}(x) = \Phi_{ij}^{(k)} = \mathrm{relu}(x_i - b_j^{(k)})$$

We have the *Limit Kernel*

$$\begin{aligned}
\mathcal{K}(x, z) &= \lim_{d^{(k)} \to \infty} d^{(k)\,-1} \Phi^{(k)T}(x) \Phi^{(k)}(z). \\
&= \int_0^1 \mathrm{relu}(x - t)\mathrm{relu}(z - t) dt \\
&= min(x, z)^2 (3max(x, z) - min(x, z))/6,
\end{aligned}$$

For the full feature map $\Psi(x) = (1, \Phi(x))$, the matrix $K_{i,j} = \Psi(x_i)^T \Psi(x_j)$ obeys

$$K \approx J + (\mathcal{K}(x_i, x_j))$$

where $J = 11^T$ is the matrix of all ones.

Actual NN Training

- Non-simplified (canonical) form

$$f(x) = \sum_j w_j^2 \text{relu}(w_j^1 x - b_j)$$

- Training Problem

$$\hat{f}_\lambda = \text{argmin} \mathcal{L}(y, f) + \lambda \cdot (\sum_j (w_j^2)^2 + \sum_j (w_j^1)^2.$$

Weight-decay Interpolation

- Interpolation

$$y_i = f(x_i), \qquad i = 1, \ldots, n.$$

$$f(x) = \sum_j w_j^2 \mathrm{relu}(w_j^1 x - b_j)$$

- Weight-decay interpolation

$$\hat{f}^{wd} = \mathrm{argmin} \sum_j (w_j^2)^2 + \sum_j (w_j^1)^2 : \text{ subject to } y_i = f(x_i).$$

- Equivalent form (post-calibration)

$$f(x) = f(0) + c_0 x + \sum_j c_j \mathrm{relu}(x - b_j)$$

$$\hat{f}^{\ell_1} = \mathrm{argmin} \|c\|_1 : \text{ subject to } y_i = f(x_i).$$

With appropriate algorithm, $\hat{f}^{\ell_1} = \hat{f}^{wd}$ [surprise!]

Reasoning:

$$w_j^2 \mathrm{relu}(w_j^1 x - b_j) = c_j \mathrm{relu}(x - \tilde{b}_j)$$

where we calibrate $\tilde{b}_j \cdot c_j \equiv b_j w_j^2$ and $c_j \equiv w_j^2 w_j^1$.

Lemma.

$$\min \sum_j x_j^2 + y_j^2 \text{ subject to } x_j y_j = z_j$$

is achieved when

$$x_j = y_j = \sqrt{z_j}$$

and achieves the value

$$\sum_j x_j^2 + y_j^2 = 2 \sum |z_j|$$

How do infinite width bounded norm networks look in function space?

Pedro Savarese[1] savarese@ttic.edu
Itay Evron[2] evron.itay@gmail.com
Daniel Soudry[2] daniel.soudry@gmail.com
Nathan Srebro[1] nati@ttic.edu

[1] Toyota Technological Institute at Chicago, Chicago IL, USA
[2] Department of Electrical Engineering, Technion, Haifa, Israel

Abstract

We consider the question of what functions can be captured by ReLU networks with an unbounded number of units (infinite width), but where the overall network Euclidean norm (sum of squares of all weights in the system, except for an unregularized bias term for each unit) is bounded; or equivalently what is the minimal norm required to approximate a given function. For functions $f : \mathbb{R} \to \mathbb{R}$ and a single hidden layer, we show that the minimal network norm for representing f is $\max(\int |f''(x)| \, \mathrm{d}x \, , \, |f'(-\infty) + f'(+\infty)| \,)$, and hence the minimal norm fit for a sample is given by a linear spline interpolation.