Deep Learning meets statistics: Improving neural networks with statistical theory

Sophie Langer

Algorithms & Computationally Intensive Inference seminar, Warwick Statistics | June 2021
Motivation
Motivation

Successful, but why?
Motivation

Successful applications
...but lack of mathematical understanding

Aim
Understanding theoretical properties of deep neural networks

Research field
Deep neural networks in nonparametric regression
→ Least squares regression estimates using deep neural networks
Motivation

**Figure:** Fundamental research topics of Deep Learning

- **Problem:** Describing the procedure of DNNs by hand is a highly complex task
- Theoretical results focus either on approximation, generalization or optimization of DNNs
- **But:** Results did not take into account all three aspects of Deep Learning, simultaneously

Cannot be used to improve estimators in practice

Should it not be the aim of statistical theory to not only understand but also improve estimators in practice?
A promising result

• Braun et al. (2021): Analysis of shallow networks learned by gradient descent
  ↔ Good rate of convergence results
  ↔ Improved performance on simulated data
  ↔ Can be considered as the *smoking gun* of deep learning theory, as it shows for the first time that statistical theory can be used to improve estimators in practice
Nonparametric regression

Aim
Study the **statistical** performance of neural networks

\[ \sim \sim \sim \text{Analyze neural networks in the context of nonparametric regression} \]

Prediction problem
Given \( \mathbb{R}^d \times \mathbb{R} \)-valued random vector \((X, Y)\)

\[ \text{Try to predict } Y \text{ by } f^*(X) \text{ from some } f^* : \mathbb{R}^d \rightarrow \mathbb{R} \]

\[ \sim \sim \sim \text{Find } f^* : \mathbb{R}^d \rightarrow \mathbb{R} \text{ such that} \]

\[
\mathbb{E}\{ |f^*(X) - Y|^2 \} = \min_{f: \mathbb{R}^d \rightarrow \mathbb{R}} \mathbb{E}\{ |f(X) - Y|^2 \}.
\]

It can be shown, that \( f^*(x) = m(x) := \mathbb{E}\{ Y | X = x \} \).

\[ \sim \sim \sim m : \mathbb{R}^d \rightarrow \mathbb{R} \text{ is the so-called regression function} \]
Nonparametric regression

- **Problem:** Distribution of \((X, Y)\) is unknown
- Given a set of \(n\) of i.i.d. observations \(\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}\) of \((X, Y)\)
- Reconstruct \(m\) by an estimator

\[
m_n(\cdot) = m_n(\cdot, \mathcal{D}_n) : \mathbb{R}^d \to \mathbb{R}
\]

such that

\[
\int |m_n(x) - m(x)|^2 p_X(dx)
\]

is small.
Define

\[ \mathcal{F}_n = \left\{ \left[ \sum_{k=1}^{\lceil \sqrt{n} \rceil} \alpha_k \cdot \sigma(\beta_k \cdot \mathbf{x} + \gamma_k) : \alpha_k, \gamma_k \in \mathbb{R}, \beta_k \in \mathbb{R}^d, \sum_{k=0}^{K_n} |\alpha_k| \leq L_n \right]\right\}, \]

where \( \sigma(u) = 1/(1 + \exp(-u)) \) (\( u \in \mathbb{R} \)) and let

\[ m_n(\cdot) = \arg \min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(X_i)|^2 \]

be the corresponding least squares estimator.
Barron’s result

Define

$$\mathcal{F}_n = \left\{ \sum_{k=1}^{[\sqrt{n}]} a_k \cdot \sigma(\beta_k \cdot x + \gamma_k) : a_k, \gamma_k \in \mathbb{R}, \beta_k \in \mathbb{R}^d, \sum_{k=0}^{K_n} |a_k| \leq L_n \right\},$$

where \( \sigma(u) = 1/(1 + \exp(-u)) \) (\( u \in \mathbb{R} \)) and let

$$m_n(\cdot) = \arg \min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(X_i)|^2$$

be the corresponding least squares estimator. Then

$$\mathbb{E} \int |m_n(x) - m(x)|^2 P_x(dx) \leq c_1 \cdot (\log n)^5 \cdot \frac{1}{\sqrt{n}}$$

holds whenever the Fourier transform of the regression function has a finite first moment.
An estimator learned by gradient descent

We study the rate of convergence of a neural network estimators learned by gradient descent
An estimator learned by gradient descent

We study the rate of convergence of a neural network estimators learned by gradient descent

We need the following definitions:

\[
\sigma(u) = \frac{1}{1 + \exp(-u)} \quad (u \in \mathbb{R}),
\]
An estimator learned by gradient descent

We study the rate of convergence of a neural network estimators learned by gradient descent.

We need the following definitions:

\[
\sigma(u) = 1/(1 + \exp(-u)) \quad (u \in \mathbb{R}),
\]

\[
f_{\text{net},w}(x) = \alpha_0 + \sum_{j=1}^{K_n} \alpha_j \cdot \sigma(\beta_j^T \cdot x + \gamma_j),
\]

where

\[
w = (\alpha_0, \alpha_1, \ldots, \alpha_{K_n}, \beta_1, \ldots, \beta_{K_n}, \gamma_1, \ldots, \gamma_{K_n}),
\]
An estimator learned by gradient descent

We study the rate of convergence of a neural network estimators learned by gradient descent.

We need the following definitions:

\[
\sigma(u) = \frac{1}{1 + \exp(-u)} \quad (u \in \mathbb{R}),
\]

\[
f_{\text{net}, \mathbf{w}}(\mathbf{x}) = \alpha_0 + \sum_{j=1}^{K_n} \alpha_j \cdot \sigma(\beta_j^T \cdot \mathbf{x} + \gamma_j),
\]

where

\[
\mathbf{w} = (\alpha_0, \alpha_1, \ldots, \alpha_{K_n}, \beta_1, \ldots, \beta_{K_n}, \gamma_1, \ldots, \gamma_{K_n}),
\]

and

\[
F(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} |Y_i - f_{\text{net}, \mathbf{w}}(\mathbf{X}_i)|^2 + \frac{c_2}{K_n} \cdot \sum_{k=0}^{K_n} \alpha_k^2.
\]
An estimator learned by gradient descent

- **Initial weights:**

\[
\mathbf{w}(0) = (\alpha_0(0), \ldots, \alpha_{K_n}(0), \beta_1(0), \ldots, \beta_{K_n}(0), \gamma_1(0), \ldots, \gamma_{K_n}(0))
\]

such that

\[
\alpha_0(0) = \alpha_1(0) = \cdots = \alpha_{K_n}(0) = 0
\]
An estimator learned by gradient descent

• Initial weights:

\[ \mathbf{w}(0) = (\alpha_0(0), \ldots, \alpha_{K_n}(0), \beta_1(0), \ldots, \beta_{K_n}(0), \gamma_1(0), \ldots, \gamma_{K_n}(0)) \]

such that

\[ \alpha_0(0) = \alpha_1(0) = \cdots = \alpha_{K_n}(0) = 0 \]

and \( \beta_1(0), \ldots, \beta_{K_n}(0), \gamma_1(0), \ldots, \gamma_{K_n}(0) \) independently randomly chosen such that

• \( \beta_k(0) \) are uniformly distributed on a sphere with radius \( B_n \)
• \( \gamma_j(0) \) are uniformly distributed on \( [-B_n \cdot \sqrt{d}, B_n \cdot \sqrt{d}] \).
An estimator learned by gradient descent

- **Initial weights:**

\[
\mathbf{w}(0) = (\alpha_0(0), \ldots, \alpha_{K_n}(0), \beta_1(0), \ldots, \beta_{K_n}(0), \gamma_1(0), \ldots, \gamma_{K_n}(0))
\]

such that

\[
\alpha_0(0) = \alpha_1(0) = \cdots = \alpha_{K_n}(0) = 0
\]

and \(\beta_1(0), \ldots, \beta_{K_n}(0), \gamma_1(0), \ldots, \gamma_{K_n}(0)\) independently randomly chosen such that

- \(\beta_k(0)\) are uniformly distributed on a sphere with radius \(B_n\)
- \(\gamma_j(0)\) are uniformly distributed on \([-B_n \cdot \sqrt{d}, B_n \cdot \sqrt{d}]\).

- **\(t_n\) gradient descent steps:**

\[
\mathbf{w}(t + 1) = \mathbf{w}(t) - \lambda_n \cdot \nabla_{\mathbf{w}} F(\mathbf{w}(t)) \quad (t = 0, \ldots, t_n - 1).
\]
An estimator learned by gradient descent

- The estimator:

\[ \tilde{m}_n(\cdot) = f_{\text{net}, \mathbf{w}(t_n)}(\cdot) \quad \text{and} \quad m_n(x) = T_{c_1 \cdot \log n} \tilde{m}_n(x) \]

where \( T_L z = \max\{\min\{z, L\}, -L\} \) for \( z \in \mathbb{R} \) and \( L \geq 0 \).
An estimator learned by gradient descent

- The estimator:
  \[ \tilde{m}_n(\cdot) = f_{\text{net},w(t_n)}(\cdot) \quad \text{and} \quad m_n(x) = T_{c_1 \cdot \log n} \tilde{m}_n(x) \]
  where \( T_L z = \max\{\min\{z, L\}, -L\} \) for \( z \in \mathbb{R} \) and \( L \geq 0 \).

- Main assumption: Fourier transform
  \[ \mathcal{F} m(\omega) = \frac{1}{(2\pi)^{d/2}} \cdot \int_{\mathbb{R}^d} e^{-i \cdot \omega^T x} \cdot m(x) \, dx \]
  of the regression function satisfies
  \[ |\mathcal{F} m(\omega)| \leq \frac{c_3}{\|\omega\|^{d+1+\epsilon}} \quad (\omega \in \mathbb{R}^d \setminus \{0\}) \quad (1) \]
  for some \( \epsilon \in (0, 1] \) and some \( c_3 > 0 \).
An estimator learned by gradient descent

Theorem: If

- Fourier transform $\mathcal{F}m$ satisfies (1)
- number of neurons $K_n \approx \sqrt{n}$
- $B_n \approx n^{5/2}$
- learning rate $\lambda_n \approx n^{-1.25}$
- gradient descent steps $t_n \approx n^{1.75}$

Then

$$E \int |m_n(x) - m(x)|^2 p_x(dx) \leq c_4 \cdot (\log n)^4 \cdot \frac{1}{\sqrt{n}}.$$
Set $\tilde{K}_n = [K_n/(\log n)^4]$. In the proof we show that with high probability
\[
\mathbf{w}(0) = (\alpha_0(0), \ldots, \alpha_{K_n}(0), \beta_1(0), \ldots, \beta_{K_n}(0), \gamma_1(0), \ldots, \gamma_{K_n}(0))
\]
is chosen such that
\[
\int \left| \sum_{k=1}^{\tilde{K}_n} \bar{\alpha}_k \cdot \sigma(\beta_k(0)^T \cdot \mathbf{x} + \gamma_k(0)) - m(\mathbf{x}) \right|^2 \mathbf{P}_\mathbf{x}(d\mathbf{x})
\]
is small for some (random) $1 \leq i_1 < \cdots < i_{\tilde{K}_n}$ and some (random) $\bar{\alpha}_{i_1}, \ldots, \bar{\alpha}_{i_{\tilde{K}_n}} \in \mathbb{R}$,
On the proof

Set $\tilde{K}_n = \lceil K_n / (\log n)^4 \rceil$. In the proof we show that with high probability

$$w(0) = (\alpha_0(0), \ldots, \alpha_{K_n}(0), \beta_1(0), \ldots, \beta_{K_n}(0), \gamma_1(0), \ldots, \gamma_{K_n}(0))$$

is chosen such that

$$\left\| \sum_{k=1}^{\tilde{K}_n} \bar{\alpha}_k \cdot \sigma(\beta_k(0)^T \cdot x + \gamma_k(0)) - m(x) \right\|^2 \mathbb{P}_x(dx)$$

is small for some (random) $1 \leq i_1 < \cdots < i_{\tilde{K}_n}$ and some (random) $\bar{\alpha}_{i_1}, \ldots, \bar{\alpha}_{i_{\tilde{K}_n}} \in \mathbb{R}$, and that during the gradient descent the inner weights

$$\beta_{i_1}(0), \gamma_{i_1}(0), \ldots, \beta_{i_{\tilde{K}_n}}(0), \gamma_{i_{\tilde{K}_n}}(0)$$

change only slightly.
A lower bound

Under the above assumption a much better rate of convergence than \(1/\sqrt{n}\) is not possible:
A lower bound

Under the above assumption a much better rate of convergence than \(1/\sqrt{n}\) is not possible:

**Theorem:** Let \(\mathcal{D}\) be the class of all distributions of \((X, Y)\) which satisfy the assumptions of the above Theorem. Then

\[
\inf_{\hat{m}_n} \sup_{(X,Y) \in \mathcal{D}} \mathbb{E} \int |\hat{m}_n(x) - m(x)|^2 P_X(dx) \geq c_5 \cdot n^{-\frac{1}{2} - \frac{1}{\sigma+1}},
\]

where the infimum is taken with respect to all estimates \(\hat{m}_n\), i.e., all measurable functions of the data.
A simplified estimator

Insights in our statistical analysis help us simplify our estimate as follows:

- Choose
  - \( \beta_1, \ldots, \beta_{K_n}, \gamma_1, \ldots, \gamma_{K_n} \) i.i.d.
  - \( \beta_1, \ldots, \beta_{K_n} \) uniformly distributed on \( \{ \mathbf{x} \in \mathbb{R}^d : \| \mathbf{x} \| = B_n \} \)
  - \( \gamma_1, \ldots, \gamma_{K_n} \) uniformly distributed on \( [-B_n \cdot \sqrt{d}, B_n \cdot \sqrt{d}] \)

- Denote the linear function space by

\[
\mathcal{F}_n = \left\{ f : \mathbb{R}^d \to \mathbb{R} : f(\mathbf{x}) = \alpha_0 + \sum_{j=1}^{K_n} \alpha_j \cdot \sigma \left( \beta_j^T \cdot \mathbf{x} + \gamma_j \right) \right\}
\]

for some \( \alpha_0, \ldots, \alpha_{K_n} \in \mathbb{R} \)
A simplified estimator

- Choose the estimate according to the principle of least squares

\[ \tilde{m}_n = \arg \min_{f \in F_n} \frac{1}{n} \sum_{i=1}^{n} |y_i - f(x_i)|^2. \]
A simplified estimator

- Choose the estimate according to the principle of least squares

\[ \tilde{m}_n = \arg \min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(X_i)|^2. \]

- Truncate it on some level \( \beta_n = c_1 \cdot \log n \)

\[ m_n = T_{\beta_n} \tilde{m}_n, \]

where \( T_L z = \max\{\min\{z, L\}, -L\} \) for \( z \in \mathbb{R} \) and \( L \geq 0 \).
A simplified estimator

Theorem: If

- the Fourier transform $\mathcal{F}m$ satisfies (1)
- number of summands $K_n \approx \sqrt{n}$
- $B_n = \frac{1}{\sqrt{d}} \cdot (\log n)^2 \cdot K_n \cdot n^2$.

Then

$$E \int |m_n(x) - m(x)|^2 P_x(dx) \leq c_6 \cdot (\log n)^4 \cdot \frac{1}{\sqrt{n}}.$$
A simplified estimator

- Same rate as for the neural network estimate learned by gradient descent, **but** much faster in computation
- Ability to learn a good hierarchical representation of the data is considered as a key factor of Deep Learning
  ~~~ So-called representation learning (see Goodfellow et al. (2016))
  **Suprisingly**: In our estimate it is much more a representation **guessing**
Application to simulated data

The setting:

- Generate $n \in \{200, 400\}$ independent observations from

$$Y = m_i(X) + \sigma_j \cdot \gamma_i \cdot \epsilon \quad (i \in \{1, \ldots, 6\}, j \in \{1, 2\})$$

with

- $X$ uniformly distributed on $[0, 1]^d$
- $\epsilon$ standard normally distributed
- $\sigma_j \in \{0.05, 0.2\}$
- $\gamma_i$ covers the range of $m_i$ on $X$
Application to simulated data

Implementation of the estimate:

- Choose
  - \( \beta_1, \ldots, \beta_{K_n} \) uniformly distributed on \( \{ x \in \mathbb{R}^d : \|x\| = B_n \} \)
  - \( \gamma_1, \ldots, \gamma_{K_n} \) uniformly distributed on \( [-B_n \cdot \sqrt{d}, B_n \cdot \sqrt{d}] \)
- Solve a linear equation system to compute the values of \( \alpha_0, \ldots, \alpha_{K_n} \)
Application to simulated data

Implementation of the estimate:

• Choose
  • $\beta_1, \ldots, \beta_{K_n}$ uniformly distributed on $\{x \in \mathbb{R}^d : \|x\| = B_n\}$
  • $\gamma_1, \ldots, \gamma_{K_n}$ uniformly distributed on $[-B_n \cdot \sqrt{d}, B_n \cdot \sqrt{d}]$

• Solve a linear equation system to compute the values of $\alpha_0, \ldots, \alpha_{K_n}$

• Choose the values of $B_n$ and $K_n$ by the splitting of the sample with
  \[ n_{train} = 0.8 \cdot n \text{ and } n_{test} = 0.2 \cdot n \]

out of the sets

• $K_n \in \{1, 2, \ldots, 6, 8, 16, \ldots, 131072\}$
• $B_n \in \{1, 2, \ldots, 6, 8, 16, \ldots, 524288\}$
Application to simulated data

Competitive approaches:

- **Standard nets** of the Deep Learning framework of *tensorflow* and *keras* with
  - ReLU activation function (abbrv. *relu-net*)
  - Sigmoid activation function (abbrv. *sig-net*)

and

- 1 hidden layer (abbrv. *net-1*)
- 3 hidden layers (abbrv. *net-3*)
- 6 hidden layers (abbrv. *net-6*)

- Choose number of neurons with the splitting of the sample procedure out of the set \{4, 8, 16, 32, 64\}
Application to simulated data

Test functions:

\[
m_1(x) = \exp \left( 0.5 \cdot \sum_{i=1}^{7} x_i^2 \right),
\]
\[
m_2(x) = \frac{1}{4000} \cdot \sum_{i=1}^{7} x_i^2 - \prod_{i=1}^{7} \cos \left( \frac{x_i}{\sqrt{i-1}} \right),
\]
\[
m_3(x) = \sum_{i=1}^{6} 100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2,
\]
\[
m_4(x) = \frac{1}{1 + \frac{\|x\|}{4}} + x_7^2 + x_4 \cdot x_5 \cdot x_2,
\]
4. Application to simulated data

The test functions:

\[
m_5(\mathbf{x}) = \tanh(0.2x_1 + 0.9x_2 + x_3 + x_4 + 0.2 \cdot x_5 + 0.6 \cdot x_6).
\]

\[
m_6(\mathbf{x}) = \cot \left( \frac{\pi}{1 + \exp(x_1^2 + 2 \cdot x_2 + \sin(6 \cdot x_4^2 - 3))} \right) + \exp(3 \cdot x_3 + 2 \cdot x_4 - 5 \cdot x_5 + \sqrt{x_6 + 0.9 \cdot x_7 + 0.1}),
\]

Warwick Statistics | S. Langer | 22
Application to simulated data

Evaluation:

- Quality of each of the estimates is determined by the empirical $L_2$-error, i.e., by

$$
\epsilon_{L_2,N} = \frac{1}{N} \sum_{k=1}^{N} (m_{n,i}(x_{n+k}) - m_i(x_{n+k})^2),
$$

- Input values $x_{n+1}, x_{n+2}, \ldots, x_{n+N}$ are newly generated independent realizations of the random value $X$
- $N = 10^5$
- Each estimation is performed 50 times with different values of $(X, \epsilon)$
### Application to simulated data

**Results:**

\[ m_1(x) = \exp \left( 0.5 \cdot \sum_{i=1}^{7} x_i^2 \right) \]

<table>
<thead>
<tr>
<th>noise</th>
<th>5%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample size</td>
<td>( n = 200 )</td>
<td>( n = 400 )</td>
</tr>
<tr>
<td>( \bar{e}_{L_2,N}(\text{avg)} )</td>
<td>0.6823</td>
<td>0.6823</td>
</tr>
<tr>
<td>relu-net-1</td>
<td>0.0554(0.0126)</td>
<td>0.0309(0.0132)</td>
</tr>
<tr>
<td>relu-net-3</td>
<td>0.0632(0.0110)</td>
<td>0.0428(0.0100)</td>
</tr>
<tr>
<td>relu-net-6</td>
<td>0.0669(0.0108)</td>
<td>0.0518(0.0143)</td>
</tr>
<tr>
<td>sig-net-1</td>
<td>0.0763(0.0088)</td>
<td>0.0615(0.0038)</td>
</tr>
<tr>
<td>sig-net-3</td>
<td>0.0967(0.0150)</td>
<td>0.0683(0.0036)</td>
</tr>
<tr>
<td>sig-net-6</td>
<td>0.1335(0.0808)</td>
<td>0.0676(0.0054)</td>
</tr>
<tr>
<td>lsq-est</td>
<td><strong>0.0014</strong>(0.0009)</td>
<td><strong>0.0006</strong>(0.0003)</td>
</tr>
</tbody>
</table>
Application to simulated data

Results:

\[ m_2(x) = \frac{1}{4000} \cdot \sum_{i=1}^{7} x_i^2 - \prod_{i=1}^{7} \cos \left( \frac{x_i}{\sqrt{i-1}} \right) \]

<table>
<thead>
<tr>
<th>noise</th>
<th>5%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample size</td>
<td>n = 200</td>
<td>n = 400</td>
</tr>
<tr>
<td>(\bar{\epsilon}_{L_2,N}(\text{avg}))</td>
<td>2.2190</td>
<td>2.2154</td>
</tr>
<tr>
<td>relu-net-1</td>
<td>0.1123(0.0267)</td>
<td>0.0582(0.0070)</td>
</tr>
<tr>
<td>relu-net-3</td>
<td>0.1015(0.0252)</td>
<td>0.0607(0.0104)</td>
</tr>
<tr>
<td>relu-net-6</td>
<td>0.0950(0.0261)</td>
<td>0.0642(0.0084)</td>
</tr>
<tr>
<td>sig-net-1</td>
<td>0.2278(0.0821)</td>
<td>0.0918(0.0266)</td>
</tr>
<tr>
<td>sig-net-3</td>
<td>0.3058(0.4057)</td>
<td>0.0745(0.0117)</td>
</tr>
<tr>
<td>sig-net-6</td>
<td>0.9654(0.6720)</td>
<td>0.1045(0.0420)</td>
</tr>
<tr>
<td>lsq-est</td>
<td><strong>0.0303</strong>(0.0095)</td>
<td><strong>0.0167</strong>(0.0043)</td>
</tr>
</tbody>
</table>
Application to simulated data

Results:

\[ m_3(x) = \sum_{i=1}^{6} 100 \cdot (x_{i+1} - (x_i)^2)^2 + (x_i - 1)^2 \]

<table>
<thead>
<tr>
<th>noise</th>
<th>sample size</th>
<th>5%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n = 200</td>
<td>n = 400</td>
<td>n = 200</td>
</tr>
<tr>
<td></td>
<td>( \bar{e}_{L_2,N}(\text{avg}) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>relu-net-1</td>
<td>0.6428(0.2389)</td>
<td>0.2083(0.0359)</td>
<td>0.7063(0.1625)</td>
</tr>
<tr>
<td>relu-net-3</td>
<td>0.6845(0.2401)</td>
<td>0.1846(0.0455)</td>
<td>0.6757(0.3474)</td>
</tr>
<tr>
<td>relu-net-6</td>
<td>0.5204(0.2810)</td>
<td>0.2114(0.1316)</td>
<td>0.6455(0.3179)</td>
</tr>
<tr>
<td>sig-net-1</td>
<td>0.9820(0.0297)</td>
<td>0.9442(0.0350)</td>
<td>0.9851(0.0354)</td>
</tr>
<tr>
<td>sig-net-3</td>
<td>1.0175(0.0322)</td>
<td>0.9934(0.0358)</td>
<td>1.0047(0.0248)</td>
</tr>
<tr>
<td>sig-net-6</td>
<td>1.1014(0.0275)</td>
<td>1.0016(0.0253)</td>
<td>1.0064(0.0271)</td>
</tr>
<tr>
<td>lsq-est</td>
<td><strong>0.2181</strong> (0.0577)</td>
<td><strong>0.1024</strong> (0.0161)</td>
<td><strong>0.3970</strong> (0.1589)</td>
</tr>
</tbody>
</table>
Application to simulated data

Results:

\[ m_4(x) = \frac{1}{1 + \frac{\|x\|}{4}} + x_7^2 + x_4 \cdot x_5 \cdot x_2 \]

<table>
<thead>
<tr>
<th>noise</th>
<th>5%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample size</td>
<td>n = 200</td>
<td>n = 400</td>
</tr>
<tr>
<td>( \bar{e}_{L_2,N}(\text{avg}) )</td>
<td>0.0049</td>
<td>0.0049</td>
</tr>
<tr>
<td>relu-net-1</td>
<td>0.2069(0.1520)</td>
<td>0.0290(0.0172)</td>
</tr>
<tr>
<td>relu-net-3</td>
<td>0.2531(0.2178)</td>
<td>0.0245(0.0283)</td>
</tr>
<tr>
<td>relu-net-6</td>
<td>0.0760(0.2178)</td>
<td>0.0276(0.0290)</td>
</tr>
<tr>
<td>sig-net-1</td>
<td>0.1245(0.0188)</td>
<td>0.0038(0.0077)</td>
</tr>
<tr>
<td>sig-net-3</td>
<td>0.1375(0.5441)</td>
<td>0.0032(0.0026)</td>
</tr>
<tr>
<td>sig-net-6</td>
<td>0.1766(0.8157)</td>
<td>0.0147(0.0284)</td>
</tr>
<tr>
<td>lsq-est</td>
<td>0.0188(0.0106)</td>
<td>0.0077(0.0046)</td>
</tr>
</tbody>
</table>
### Application to simulated data

**Results:**

\[
m_5(x) = \tanh(0.2 \cdot x_1 + 0.9 \cdot x_2 + x_3 + x_4 + 0.2 \cdot x_5 + 0.6 \cdot x_6).
\]

<table>
<thead>
<tr>
<th>noise</th>
<th>5%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = 200</td>
<td>n = 400</td>
<td>n = 200</td>
</tr>
<tr>
<td>$\bar{\epsilon}_{L_2,N}(\text{avg})$</td>
<td>0.0135</td>
<td>0.0135</td>
</tr>
<tr>
<td>relu-net-1</td>
<td>0.1989(0.0737)</td>
<td>0.1019(0.0264)</td>
</tr>
<tr>
<td>relu-net-3</td>
<td>0.1826(0.0667)</td>
<td>0.1061(0.0454)</td>
</tr>
<tr>
<td>relu-net-6</td>
<td>0.1744(0.1099)</td>
<td>0.1066(0.0264)</td>
</tr>
<tr>
<td>sig-net-1</td>
<td>0.0971(0.0183)</td>
<td>0.0784(0.0071)</td>
</tr>
<tr>
<td>sig-net-3</td>
<td><strong>0.0935</strong> (0.0202)</td>
<td>0.0796(0.0049)</td>
</tr>
<tr>
<td>sig-net-6</td>
<td>0.1396(0.0547)</td>
<td>0.0850(0.0087)</td>
</tr>
<tr>
<td>lsq-est</td>
<td>0.1025(0.0343)</td>
<td><strong>0.0516</strong> (0.0118)</td>
</tr>
</tbody>
</table>
Application to simulated data

Results:

\[
\begin{align*}
m_6(x) &= \cot \left( \frac{\pi}{1 + \exp(x_1^2 + 2 \cdot x_2 + \sin(6 \cdot x_4^2 - 3))} \right) \\
& \quad + \exp(3 \cdot x_3 + 2 \cdot x_4 - 5 \cdot x_5 + \sqrt{x_6 + 0.9 \cdot x_7 + 0.1})
\end{align*}
\]

<table>
<thead>
<tr>
<th>noise</th>
<th>5%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample size</td>
<td>(n = 200)</td>
<td>(n = 400)</td>
</tr>
<tr>
<td>(\bar{c}_{L_2,N}(\text{avg}))</td>
<td>591.77</td>
<td>592.74</td>
</tr>
<tr>
<td>relu-net-1</td>
<td>0.1897(0.0913)</td>
<td>0.0552(0.0346)</td>
</tr>
<tr>
<td>relu-net-3</td>
<td><strong>0.1349</strong> (0.0828)</td>
<td>0.0398(0.0315)</td>
</tr>
<tr>
<td>relu-net-6</td>
<td>0.1075(0.0856)</td>
<td><strong>0.0385</strong> (0.0247)</td>
</tr>
<tr>
<td>sig-net-1</td>
<td>0.5733(0.1122)</td>
<td>0.2027(0.0539)</td>
</tr>
<tr>
<td>sig-net-3</td>
<td>0.8355(0.1451)</td>
<td>0.3905(0.1181)</td>
</tr>
<tr>
<td>sig-net-6</td>
<td>1.0058(0.0728)</td>
<td>0.7017(0.1435)</td>
</tr>
<tr>
<td>lsq-est</td>
<td>0.3070(0.1534)</td>
<td>0.1784(0.0758)</td>
</tr>
</tbody>
</table>
Application to simulated data

Improvement of the results:

- Choose a combined estimator \textit{comb-new}, that chooses between the estimators
  - New least squares estimator
  - Standard nets
  
  the one with the smallest empirical $L_2$-error on the dataset $x_{test}$

- Compare this estimator with a classical combined estimator \textit{comb-class}, that chooses the best standard net according to the smallest empirical $L_2$-error on the test sample
Application to simulated data

Results:

\[ m_1(x) = \exp \left( 0.5 \cdot \sum_{i=1}^{7} x_i^2 \right) \]

<table>
<thead>
<tr>
<th>noise</th>
<th>sample size</th>
<th>5%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n = 200</td>
<td>n = 400</td>
<td>n = 200</td>
</tr>
<tr>
<td>\tilde{\epsilon}_{L_2,N} (avg)</td>
<td>0.6823</td>
<td>0.6823</td>
<td>0.6820</td>
</tr>
<tr>
<td>relu-net-1</td>
<td>0.0554(0.0126)</td>
<td>0.0309(0.0132)</td>
<td>0.0586(0.0099)</td>
</tr>
<tr>
<td>relu-net-3</td>
<td>0.0632(0.0110)</td>
<td>0.0428(0.0100)</td>
<td>0.0645(0.0094)</td>
</tr>
<tr>
<td>relu-net-6</td>
<td>0.0669(0.0108)</td>
<td>0.0518(0.0143)</td>
<td>0.0676(0.0125)</td>
</tr>
<tr>
<td>sig-net-1</td>
<td>0.0763(0.0088)</td>
<td>0.0615(0.0038)</td>
<td>0.0768(0.0087)</td>
</tr>
<tr>
<td>sig-net-3</td>
<td>0.0967(0.0150)</td>
<td>0.0683(0.0036)</td>
<td>0.1006(0.0140)</td>
</tr>
<tr>
<td>sig-net-6</td>
<td>0.1335(0.0808)</td>
<td>0.0676(0.0054)</td>
<td>0.1424(0.0712)</td>
</tr>
<tr>
<td>comb-classic</td>
<td>0.0552(0.0127)</td>
<td>0.0308(0.0127)</td>
<td>0.0572(0.0098)</td>
</tr>
<tr>
<td>lsq-est</td>
<td>0.0014(0.0009)</td>
<td>0.0006(0.0003)</td>
<td>0.0079(0.0025)</td>
</tr>
<tr>
<td>comb-new</td>
<td>0.0014(0.0009)</td>
<td>0.0006(0.0003)</td>
<td>0.0079(0.0025)</td>
</tr>
</tbody>
</table>
Application to simulated data

Results:

\[ m_2(x) = \frac{1}{4000} \cdot \sum_{i=1}^{7} x_i^2 - \prod_{i=1}^{7} \cos \left( \frac{x_i}{\sqrt{i-1}} \right) \]

<table>
<thead>
<tr>
<th>noise</th>
<th>5%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample size</td>
<td>n = 200</td>
<td>n = 400</td>
</tr>
<tr>
<td>(\bar{\epsilon}_{L_2,N}(\text{avg}))</td>
<td>2.2190</td>
<td>2.2154</td>
</tr>
<tr>
<td>relu-net-1</td>
<td>0.1123(0.0267)</td>
<td>0.0582(0.0070)</td>
</tr>
<tr>
<td>relu-net-3</td>
<td>0.1015(0.0252)</td>
<td>0.0607(0.0104)</td>
</tr>
<tr>
<td>relu-net-6</td>
<td>0.0950(0.0261)</td>
<td>0.0642(0.0084)</td>
</tr>
<tr>
<td>sig-net-1</td>
<td>0.2278(0.0821)</td>
<td>0.0918(0.0266)</td>
</tr>
<tr>
<td>sig-net-3</td>
<td>0.3058(0.4057)</td>
<td>0.0745(0.0117)</td>
</tr>
<tr>
<td>sig-net-6</td>
<td>0.9654(0.6720)</td>
<td>0.1045(0.0420)</td>
</tr>
<tr>
<td>comb-classic</td>
<td>0.0910(0.0238)</td>
<td>0.0560(0.0108)</td>
</tr>
<tr>
<td>lsq-est</td>
<td>0.0303(0.0095)</td>
<td>0.0167(0.0043)</td>
</tr>
<tr>
<td>comb-new</td>
<td>0.0303(0.0095)</td>
<td>0.0167(0.0043)</td>
</tr>
</tbody>
</table>
Application to simulated data

Results:

\[ m_3(x) = \sum_{i=1}^{6} 100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \]

<table>
<thead>
<tr>
<th>noise</th>
<th>sample size</th>
<th>5%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n = 200</td>
<td>n = 400</td>
<td>n = 200</td>
</tr>
<tr>
<td></td>
<td>avg</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>relu-net-1</td>
<td>0.6428(0.2389)</td>
<td>0.2083(0.0359)</td>
</tr>
<tr>
<td></td>
<td>relu-net-3</td>
<td>0.6845(0.2401)</td>
<td>0.1846(0.0455)</td>
</tr>
<tr>
<td></td>
<td>relu-net-6</td>
<td>0.5204(0.2810)</td>
<td>0.2114(0.1316)</td>
</tr>
<tr>
<td></td>
<td>sig-net-1</td>
<td>0.9820(0.0297)</td>
<td>0.9442(0.0350)</td>
</tr>
<tr>
<td></td>
<td>sig-net-3</td>
<td>1.0175(0.0322)</td>
<td>0.9934(0.0358)</td>
</tr>
<tr>
<td></td>
<td>sig-net-6</td>
<td>1.1014(0.0275)</td>
<td>1.0016(0.0253)</td>
</tr>
<tr>
<td></td>
<td>comb-classic</td>
<td>0.4630(0.2149)</td>
<td>0.1698(0.0386)</td>
</tr>
<tr>
<td></td>
<td>lsq-est</td>
<td>0.2181(0.0577)</td>
<td>0.1024(0.0161)</td>
</tr>
<tr>
<td></td>
<td>comb-new</td>
<td>0.2182(0.0577)</td>
<td>0.1024(0.0161)</td>
</tr>
</tbody>
</table>
Application to simulated data

Results:

\[ m_4(x) = \frac{1}{1 + \frac{\|x\|}{4}} + x_7^2 + x_4 \cdot x_5 \cdot x_2 \]

<table>
<thead>
<tr>
<th>noise</th>
<th>5%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample size</td>
<td>n = 200</td>
<td>n = 400</td>
</tr>
<tr>
<td>(\bar{\epsilon}_{L_2,N}(\text{avg}))</td>
<td>0.0049</td>
<td>0.0049</td>
</tr>
<tr>
<td>relu-net-1</td>
<td>0.2069 (0.1520)</td>
<td>0.0290 (0.0172)</td>
</tr>
<tr>
<td>relu-net-3</td>
<td>0.2531 (0.2178)</td>
<td>0.0245 (0.0283)</td>
</tr>
<tr>
<td>relu-net-6</td>
<td>0.0760 (0.2178)</td>
<td>0.0276 (0.0290)</td>
</tr>
<tr>
<td>sig-net-1</td>
<td>0.1245 (0.0188)</td>
<td>0.0038 (0.0077)</td>
</tr>
<tr>
<td>sig-net-3</td>
<td>0.1375 (0.5441)</td>
<td>0.0032 (0.0026)</td>
</tr>
<tr>
<td>sig-net-6</td>
<td>0.1766 (0.8157)</td>
<td>0.0147 (0.0284)</td>
</tr>
<tr>
<td>comb-classic</td>
<td>0.0445 (0.0455)</td>
<td><strong>0.0028</strong> (0.0033)</td>
</tr>
<tr>
<td>lsq-est</td>
<td><strong>0.0188</strong> (0.0106)</td>
<td>0.0077 (0.0046)</td>
</tr>
<tr>
<td>comb-new</td>
<td><strong>0.0184</strong> (0.0142)</td>
<td><strong>0.0028</strong> (0.0033)</td>
</tr>
</tbody>
</table>
Application to simulated data

Results:

\[ m_5(x) = \tanh(0.2 \cdot x_1 + 0.9 \cdot x_2 + x_3 + x_4 + 0.2 \cdot x_5 + 0.6 \cdot x_6). \]

<table>
<thead>
<tr>
<th>noise</th>
<th>5% sample size</th>
<th>5%</th>
<th>20% sample size</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( n = 200 )</td>
<td>( n = 400 )</td>
<td>( n = 200 )</td>
<td>( n = 400 )</td>
</tr>
<tr>
<td></td>
<td>( \bar{e}_{L_2,N}(\text{avg}) )</td>
<td>0.0135</td>
<td>0.0135</td>
<td>0.1345</td>
</tr>
<tr>
<td>relu-net-1</td>
<td>0.1989(0.0737)</td>
<td>0.1019(0.0264)</td>
<td>0.2846(0.1094)</td>
<td>0.1768(0.0618)</td>
</tr>
<tr>
<td>relu-net-3</td>
<td>0.1826(0.0667)</td>
<td>0.1061(0.0454)</td>
<td>0.3182(0.1858)</td>
<td>0.2230(0.0808)</td>
</tr>
<tr>
<td>relu-net-6</td>
<td>0.1744(0.1099)</td>
<td>0.1066(0.0264)</td>
<td>0.3519(0.1995)</td>
<td>0.2005(0.0621)</td>
</tr>
<tr>
<td>sig-net-1</td>
<td>0.0971(0.0183)</td>
<td>0.0784(0.0071)</td>
<td>0.1139(0.0202)</td>
<td>0.0880(0.0131)</td>
</tr>
<tr>
<td>sig-net-3</td>
<td>0.0935(0.0202)</td>
<td>0.0796(0.0049)</td>
<td>0.1190(0.0434)</td>
<td>0.0880(0.0138)</td>
</tr>
<tr>
<td>sig-net-6</td>
<td>0.1396(0.0547)</td>
<td>0.0850(0.0087)</td>
<td>0.1816(0.0434)</td>
<td>0.0983(0.0207)</td>
</tr>
<tr>
<td>comb-class</td>
<td>0.0948(0.0187)</td>
<td>0.0794(0.0869)</td>
<td>\textbf{0.1121}(0.0210)</td>
<td>\textbf{0.0870}(0.0116)</td>
</tr>
<tr>
<td>lsq-est</td>
<td>0.1025(0.0343)</td>
<td>\textbf{0.0516}(0.0118)</td>
<td>0.1682(0.0434)</td>
<td>0.0981(0.0181)</td>
</tr>
<tr>
<td>comb-new</td>
<td>\textbf{0.0923}(0.0198)</td>
<td>\textbf{0.0516}(0.0118)</td>
<td>0.1162(0.0392)</td>
<td>0.0918(0.0131)</td>
</tr>
</tbody>
</table>
Application to simulated data

Results:

\[
m_6(x) = \cot \left( \frac{\pi}{1 + \exp(x_1^2 + 2 \cdot x_2 + \sin(6 \cdot x_4^2 - 3))} \right) + \exp(3 \cdot x_3 + 2 \cdot x_4 - 5 \cdot x_5 + \sqrt{x_6 + 0.9 \cdot x_7 + 0.1})
\]

<table>
<thead>
<tr>
<th>noise</th>
<th>5%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample size</td>
<td>n = 200</td>
<td>n = 400</td>
</tr>
<tr>
<td>(\bar{e}_{L_2,N}(\text{avg}))</td>
<td>591.77</td>
<td>592.74</td>
</tr>
<tr>
<td>relu-net-1</td>
<td>0.1897(0.0913)</td>
<td>0.0552(0.0346)</td>
</tr>
<tr>
<td>relu-net-3</td>
<td>0.1349(0.0828)</td>
<td>0.0398(0.0315)</td>
</tr>
<tr>
<td>relu-net-6</td>
<td>0.1075(0.0856)</td>
<td>0.0385(0.0247)</td>
</tr>
<tr>
<td>sig-net-1</td>
<td>0.5733(0.1122)</td>
<td>0.2027(0.0539)</td>
</tr>
<tr>
<td>sig-net-3</td>
<td>0.8355(0.1451)</td>
<td>0.3905(0.1181)</td>
</tr>
<tr>
<td>sig-net-6</td>
<td>1.0058(0.0728)</td>
<td>0.7017(0.1435)</td>
</tr>
<tr>
<td>comb-classic</td>
<td><strong>0.0991</strong> (0.0749)</td>
<td><strong>0.0331</strong> (0.0242)</td>
</tr>
<tr>
<td>lsq-est</td>
<td>0.3070(0.1534)</td>
<td>0.1784(0.0758)</td>
</tr>
<tr>
<td>comb-new</td>
<td>0.1063(0.0803)</td>
<td><strong>0.0331</strong> (0.0394)</td>
</tr>
</tbody>
</table>
In the analysis of our first estimate all three aspects of Deep Learning, namely approximation, generalization and optimization, were considered simultaneously.

Statistical insights helped us to construct a simplified estimate, which can be much faster computed in applications.

The combined estimate outperforms all others in our simulation study.
Eliezer Yudkowsky „By far, the greatest danger for artificial intelligence is that people conclude too early that they understand it.“

Thank you for your attention!