Algebraic Approach to Ridge-Regularized Mean Squared Error Minimization in Minimal ReLU Neural Network (Joint work with R. Fukasaku and Y. Kabata; arXiv:2508.17783)

Akifumi Okuno[1,2,3]

[1]Inst. Stat. Math., [2]SOKENDAI, [3]RIKEN (AIP/CBS)

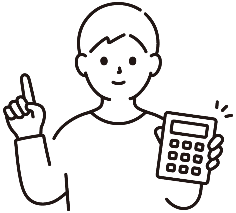
https://okuno.net/slides/2025-12-OCAMI.pdf

Today's Overview

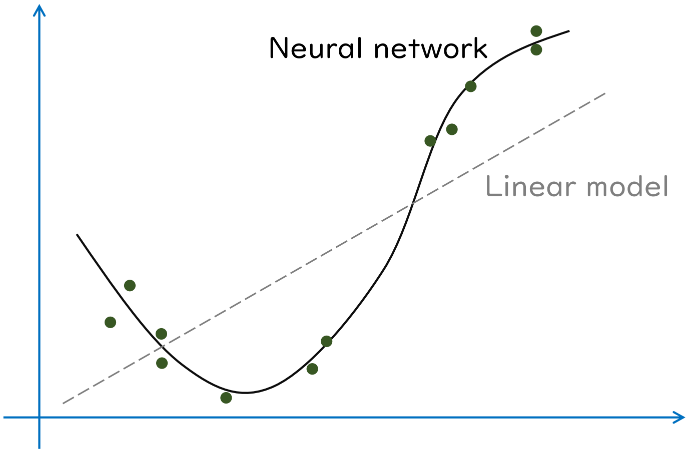
- Neural networks with the activation function $\mathrm{ReLU}(z) = \max\{0, z\}$:

$$\mathbb{R}^d \ni x \mapsto [\![a, \mathrm{ReLU}(Bx + c)]\!] + m \in \mathbb{R},$$

  are highly non-convex and difficult to optimize.
  Nevertheless, all local solutions can be enumerated using computational algebra (Fukasaku, Kabata, and Okuno; arXiv:2508.17783)



Foundations and Challenges of Neural Networks



Neural networks are flexible nonlinear predictive models.

Definition of Neural Networks

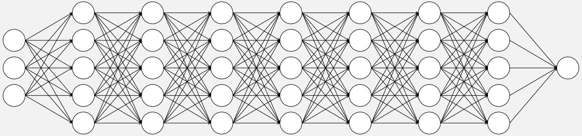
- **Linear regression model**:

$$f_\theta^{\mathrm{LM}}(x) = Wx + b$$

- **Neural network** (whose special case is the perceptron):

$$f_\theta^{\mathrm{NN}}(x) = W^{(Q+1)}\sigma\left(W^{(Q)}\sigma\left(\cdots\sigma\left(W^{(1)}x + b^{(1)}\right)\cdots\right) + b^{(Q)}\right) + b^{(Q+1)}.$$

  - σ is the activation function, applied elementwise (e.g., $1/(1+\exp(-z))$ or $\mathrm{ReLU}(z) = \max\{0, z\}$).
  - Many other architectures exist beyond this form.
  - When the number of layers Q is large, we refer to it as a deep neural network.



Neural Networks as Universal Approximators

- Let f be continuous on $I_n = [0,1]^n$. Then with one hidden layer ($Q = 1$) and sufficiently many units, there exists a neural network f^{NN} that approximates f arbitrarily well.[1]
  - Classical results: Cybenko (1989), Funahashi (1989).

- Increasing the depth Q yields exponential gains in expressive power (Telgarsky, 2016),
- Increasing Q enables highly efficient approximation rates (Yarotsky, 2017),
- As $Q \to \infty$, universal approximation holds even with fixed width (Hanin, 2017).

[1] With sigmoid activation $\sigma(z) = 1/(1 + \exp(-z))$, the approximation error can be uniformly controlled.

Implementation Is Extremely Easy (If You Just Use Them)

```
# NN Definition
class NeuralNetwork(nn.Module):
    def __init__(self, input_dim=2, activation_func='sigmoid'):
        super(NeuralNetwork, self).__init__()
        self.first = nn.Linear(input_dim, 100)
        self.hidden1 = nn.Linear(100, 100)
        self.hidden2 = nn.Linear(100, 100)
        self.hidden3 = nn.Linear(100, 100)
        self.output = nn.Linear(100, 1)
        self.activation = nn.Sigmoid() if activation_func == 'sigmoid' else nn.ReLU()

    def forward(self, x):
        x = self.activation(self.first(x))
        x = self.activation(self.hidden1(x))
        x = self.activation(self.hidden2(x))
        x = self.activation(self.hidden3(x))
        return self.output(x)
```

Once the structure is specified, modern libraries handle the training automatically.

A Wide Variety of Applications

(Generated by ChatGPT)

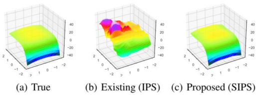
Buy a GPU, Problem Solved!

Thank you very much!

This makes everyone happy.

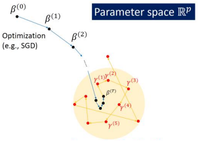
...But Reality Is Not That Simple

- From the viewpoint of statistical science, many essential issues remain unresolved.



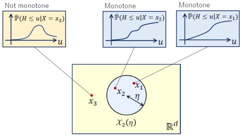
More Expressive Siamese NN

Okuno et al. (AISTATS2019)



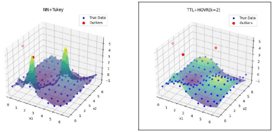
WAIC + Overparameterized NN
+ Langevin dynamics

Okuno and Yano (JCGS2023)



NN + Ordinal Regression

Okuno and Harada (JCGS2024)



NN + Variation Regularization

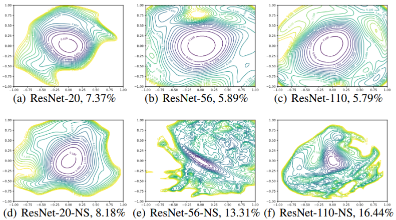
Okuno and Yagishita (in revision)

Despite substantial progress, the theoretical picture remains unclear.

Core Difficulties

- Nonlinearity.
- Redundant parametrization.
  - Overparameterization lead to degeneracy of the Fisher information
    $\Rightarrow$ many classical statistical theories break down.
  - Optimization becomes non-convex; heuristics dominate in practice.

$$\text{Typical training loss: } L(\theta) = \min_{\theta} \sum_{i=1}^{n} \{y_i - f_\theta(x_i)\}^2.$$

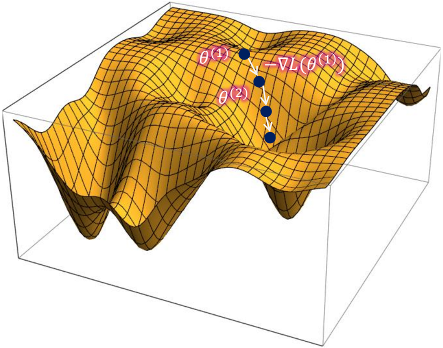


Reproduced from Li et al. (NeurIPS 2018), Fig. 5

The Loss Landscape Is Extremely Bumpy

- Gradient descent update:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma \nabla L(\theta^{(t)}).$$

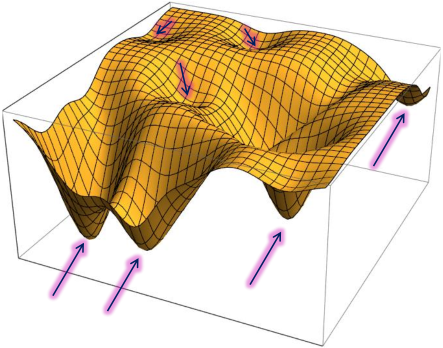


- For concave (single-valley) functions, many theoretical guarantees exist.
- For multimodal loss, the convergence limit depends on which basin the optimizer falls.
- Solutions may be non-isolated $\Rightarrow$ further problems for statistical theory.

Goals and Starting Points

What We Ultimately Want to Do

We want to *enumerate all local minima* of the loss function.



- Local minima may not be isolated; they can form higher-dimensional solution sets.
- Can computational algebra determine *all* equations that such solutions satisfy?

Ideas Leading to Our Approach

Let us take another look at the structure of a (single-output) neural network:

$$f_\theta^{\mathrm{NN}}(x) = W^{(Q+1)}\sigma\left(W^{(Q)}\sigma\left(\cdots\sigma\left(W^{(1)}x + b^{(1)}\right)\cdots\right) + b^{(Q)}\right) + b^{(Q+1)}.$$

- Often in theory, the activation $\sigma(z)$ is replaced by the identity function.

$$\begin{aligned} f_\theta^{\mathrm{LNN}}(x) &= W^{(Q+1)}\left\{W^{(Q)}\left\{\cdots\left(W^{(1)}x + b^{(1)}\right)\cdots\right\} + b^{(Q)}\right\} + b^{(Q+1)} \\ &= \widetilde{W}^{(Q+1)}\widetilde{W}^{(Q)}\ldots\widetilde{W}^{(1)}x + \widetilde{b}, \end{aligned}$$

  - The model reduces to linear regression with certain parameter constraints.
  - Also known as linear neural networks or reduced-rank regression.
    (Aoyagi and Watanabe, 2005; Mehta et al., 2022; Aoyagi, 2024)

Activation Patterns of ReLU

The ReLU activation $\sigma(z) = \max\{0, z\}$ can be expressed via activation patterns.

For fixed $W \in \mathbb{R}^{m \times d}$, $b \in \mathbb{R}^m$, and $x \in \mathbb{R}^d$, there exists $e = e(W, b, x) \in \{0, 1\}^m$ such that

$$\mathrm{ReLU}(Wx + b) = \mathrm{diag}(e)(Wx + b),$$

where $\mathrm{diag}(e)$ is the diagonal matrix with diagonal entries e.

- Example: If $Wx + b = (3, -2, 2, 1, -1)$, then $e = (1, 0, 1, 1, 0)$ and

$$\mathrm{ReLU}(Wx + b) = (3, 0, 2, 1, 0) = \mathrm{diag}(e)(Wx + b).$$

- Arora et al. (2018), Pilanci and Ergen (2020), Mishkin et al. (2022), etc.

Generalization to Multi-layer Networks

For parameters $\theta = (W^{(\ell)}, b^{(\ell)})_{\ell=1}^{L}$ and fixed input $x \in \mathbb{R}^d$, each layer $\ell = 1, \ldots, L$ has an activation pattern $e^{(\ell)} = e^{(\ell)}(\theta, x) \in \{0, 1\}^{m_\ell}$ such that

$$f_{\theta,E}^{\mathrm{NN}}(x) = W^{(Q+1)}\mathrm{diag}(e^{(Q)})\Big\{W^{(Q)}\mathrm{diag}(e^{(Q-1)})\{\cdots$$
$$\cdots \mathrm{diag}(e^{(1)})(W^{(1)}x + b^{(1)})\cdots\} + b^{(Q)}\Big\} + b^{(Q+1)}.$$

- If $E = (e^{(\ell)})$ is fixed, the ReLU network reduces to a sequence of matrix products.
- The loss

$$\ell_{\lambda,E}(\theta) = \sum_{i=1}^{n}\{y_i - f_{\theta,E}^{\mathrm{NN}}(x_i)\}^2 + \lambda\|\theta\|_2^2$$

  becomes a polynomial in the parameters.

Our Basic Idea

- The loss $\ell_{\lambda,E}(\theta)$ is a polynomial in θ.
- Its minimizer should satisfy the estimating equation:

$$\frac{\partial \ell_{\lambda,E}(\theta)}{\partial \theta} = 0,$$

  which is also a polynomial system.
- This is precisely the type of problem addressed by computational algebra.

Our Work (Fukasaku, Kabata, and Okuno; arXiv:2508.17783)

So in Principle...

$$\frac{\partial \ell_\lambda(\theta)}{\partial \theta} = \frac{\partial \left\{\sum_{i=1}^{n}(y_i - f_\theta(x_i))^2 + \lambda\|\theta\|_2^2\right\}}{\partial \theta} = 0$$

If (Dr. Fukasaku) could simply solve this equation, everything would be resolved⋯

But reality is not that kind.[2]

[2] Things are not so easy in practice.

The Main Difficulties

- The activation pattern $E = (e^{(\ell)})$ depends on both the parameters θ and the inputs x.
  - (Ideal world) Fix the activation pattern first, then solve for the optimal parameters.
  - (Reality) Once parameters are chosen, the activation pattern is determined.
  - So the dependency is reversed.
- Moreover, the dependence on the inputs themselves is a major obstacle.
  - The model effectively changes depending on the data inputs, making it difficult to analyze.

A Very "Forceful" Idea

Why not simply consider *all* activation patterns?

- Assume each possible activation pattern and solve the estimating equations.
- Among the solutions, keep only those whose parameters satisfy the assumed pattern.
- Repeat for all activation patterns, and finally merge the obtained solutions.

Thus, we solve the estimating equations (via computational algebra) for each possible activation pattern.[a]

[a] Easy to say, hard to execute.

Detailed Setup and Simplifying Assumptions

- For simplicity, restrict attention to a network with $Q = 1$ hidden layer:[3]

$$f_\theta^{\mathrm{NN}}(x) = [\![\, a \,,\, \mathrm{ReLU}(Bx + c) \,]\!] \,, \quad \theta = (a, B, c),$$

  where the number of units is L ($a, c \in \mathbb{R}^L$, $B \in \mathbb{R}^{L \times d}$).

- Eliminate a in advance. Define $\psi = (B, c)$ and consider

$$\ell_\lambda(\psi) = \min_a \left\{ \sum_{i=1}^n (y_i - f_\theta(x_i))^2 + \lambda \|\theta\|_2^2 \right\}.$$

- The minimizer in a is given analytically (ridge regression), so $\ell_\lambda(\psi)$ becomes a rational function. We therefore minimize $\ell_\lambda(\psi)$ algebraically.

[3] The essential ideas extend to general depth.

Activation Patterns and Partitioning of Parameter Space

- Consider a dataset $\{(x_i, y_i)\}_{i=1}^{n}$.
- Define $\xi_{i\ell}(\psi) = [\![\, b_\ell \,,\, x_i \,]\!] + c_\ell$ and

$$e_{i\ell} = e_{i\ell}(\psi) = \begin{cases} 1 & \text{if } \xi_{i\ell}(\psi) \geq 0, \\ -1 & \text{if } \xi_{i\ell}(\psi) < 0. \end{cases}$$

  (We now use ± 1 instead of $\{0, 1\}$ for convenience.)
- Then

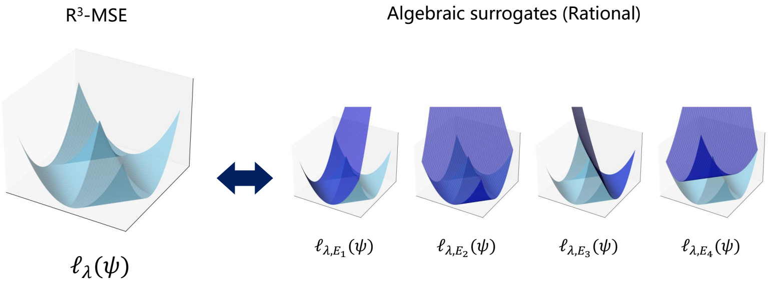
$$\mathrm{ReLU}(\xi_{i\ell}(\psi)) = \frac{e_{i\ell} + 1}{2} \xi_{i\ell}(\psi).$$

- Define the region of parameters yielding activation pattern E:

$$\Psi(E) = \{\psi \in \Psi \mid \xi_{i\ell}(\psi) e_{i\ell} \geq 0, \ \forall i, \ell\}.$$

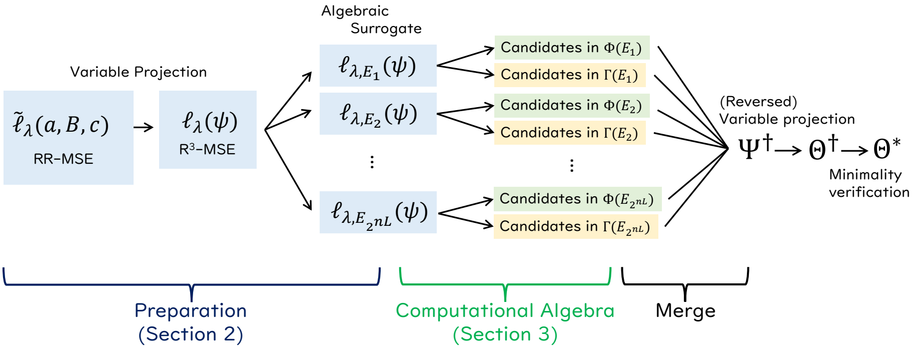
Function Decomposition and Surrogate Losses

- Our true objective is to minimize $\ell_\lambda(\psi)$.
- Partition parameter space into $\Psi(E_1), \Psi(E_2), \ldots$ based on activation patterns. In each region, $\ell_\lambda(\psi)$ equals a surrogate $\ell_{\lambda,E}(\psi)$ consistent with pattern E.



- The solutions (especially, interior points of each region) of $\frac{\partial \ell_{\lambda,E}(\psi)}{\partial \psi} = 0$ can be obtained by computational algebra.

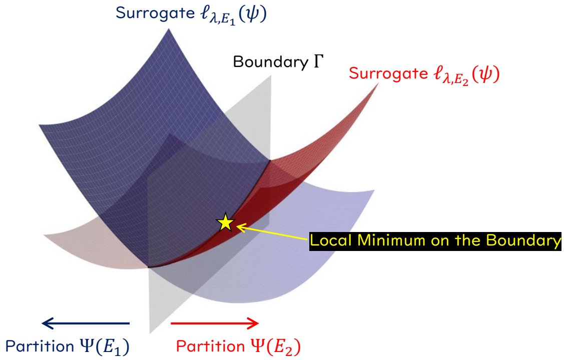
Overall Procedure



- Enumerating interior local minimizers (candidates) is relatively straightforward.
- Boundary solutions, however, are much more subtle.

Why Boundary Solutions Are Difficult

- For neighboring activation patterns $E_1, E_2 \in \{-1, +1\}^{n \times L}$, the surrogate losses ℓ_{λ, E_1} and ℓ_{λ, E_2} may each have minimizers *on the shared boundary*.



- Across the full space Ψ, neither surrogate may produce local minima. Yet *on the boundary*, switching between the surrogates can create new local minima.

Local Minima on the Boundary

- A point ψ lies on a boundary if $\xi_{i\ell}(\psi) = [\![\, b_\ell\,,\, x_i \,]\!] + c_\ell = 0$ for some (i, ℓ).
- Solve the Lagrange multiplier system:

$$\frac{\partial}{\partial \psi}\{\ell_{\lambda,E}(\psi) + \beta\, \xi_{i\ell}(\psi)\} = 0$$

which is a system of rational equations.

FKO (arXiv:2508.17783) Theorem 2

Any local minimum of ℓ_λ is either

(1) an interior local minimizer of some region $\Psi(E)$, or

(2) a local minimizer on a boundary between regions.

- Hence all local minima arise as solutions of polynomial (or rational) equations.

Algebraic Varieties

For polynomials $f_1, \ldots, f_r \in \mathbb{R}[\psi]$, define

$$\mathbb{V}(f_1, \ldots, f_r) = \{\psi \in \Psi \mid f_1(\psi) = \cdots = f_r(\psi) = 0\}.$$

Computing a Gröbner basis yields an explicit description of the variety.

In Our Setting

Interior solutions (where $\prod_{i,\ell} \xi_{i\ell}(\psi) \neq 0$) satisfy

$$\mathcal{S}_E = \mathbb{V}\left(\mathrm{num}\left(\frac{\partial \ell_{\lambda,E}(\psi)}{\partial \psi}\right)\right) \setminus \mathbb{V}\left(\mathrm{den}\left(\frac{\partial \ell_{\lambda,E}(\psi)}{\partial \psi}\right) \prod_{i,\ell} \xi_{i\ell}(\psi)\right).$$

- Boundary minimizers correspond to similar algebraic varieties.

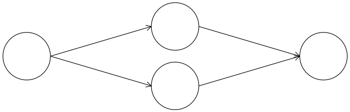
Doing This for All Activation Patterns

- There are at most 2^{nL} activation patterns. For each pattern, we compute the corresponding algebraic varieties.
- Each solution is a candidate stationary point.[4]
- We test local minimality and collect all true local minima.

[4] Not necessarily a minimizer, but all local minima are included among them.

A Concrete Example

- Input dimension $d = 1$, number of units $L = 2$, sample size $n = 5$.

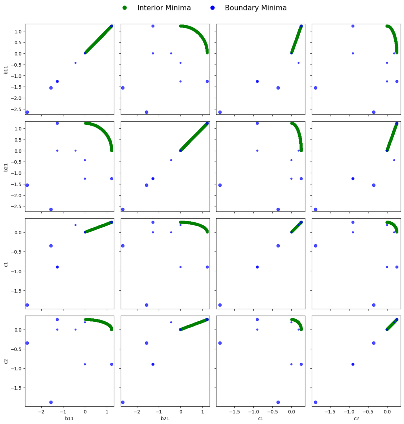


$$(x_1, y_1) = (-0.17,\ 0.05), \quad (x_2, y_2) = (0.44,\ 1.02), \quad (x_3, y_3) = (-1.00,\ 0.61),$$
$$(x_4, y_4) = (-0.40,\ -0.36), \quad (x_5, y_5) = (-0.71,\ -1.32).$$

- The number of possible activation patterns is $2^{nL} = 1024$.[5]

[5] So we must compute 1024 Gröbner bases!

Computation Results

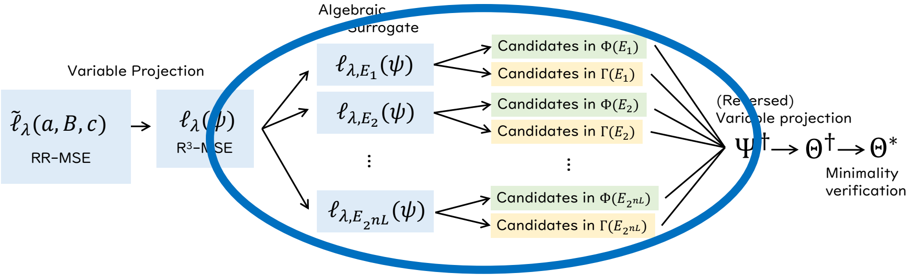


- Despite ridge regularization, an entire 1-dimensional solution set appears.
- All isolated points turned out to lie on boundaries.

Towards the Future

Remaining Challenges

- The computational cost is extremely large.
  - Increasing the number of parameters ⇒ both per-pattern computation and parallel load increase.
  - Increasing the sample size ⇒ the number of activation patterns increases exponentially.



- Future work includes parallelization and fast computation of Gröbner bases for the associated polynomial systems.

Please feel free to contact me.

okuno@ism.ac.jp

Slides available at: `https://okuno.net/slides/2025-12-OCAMI.pdf`