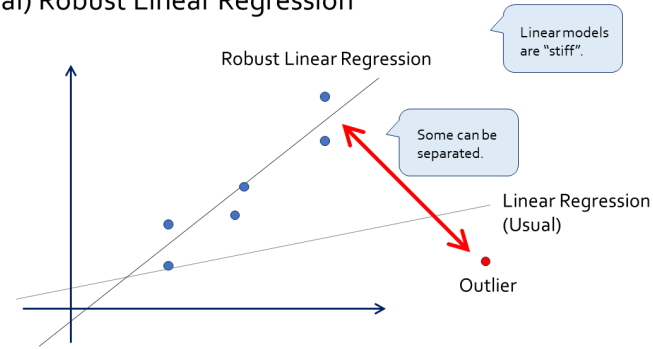
Outlier-Robust Neural Network Training: Efficient Optimization of Transformed Trimmed Loss with Variation Regularization (Submitted. arXiv:2308.02293)

Akifumi Okuno[1,2], Shotaro Yagishita[1]

[1]ISM, [2]RIKEN

Background: Traditional Robust Statistics

(Typical) Robust Linear Regression

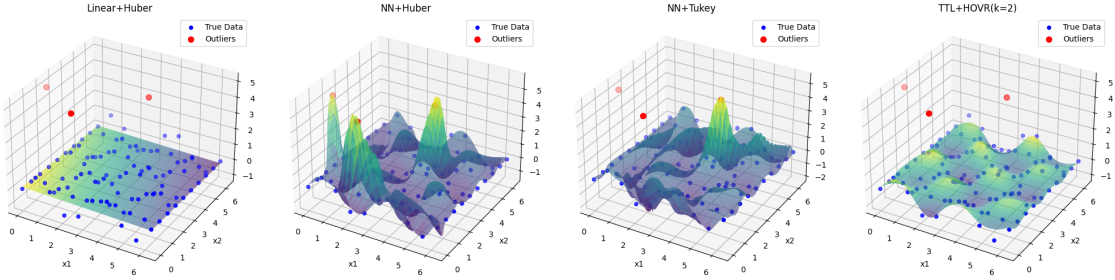


- Aims to ignore outliers.
- Uses prediction model with low-degrees-of-freedom (e.g., linear model).
- Can we apply the traditional approach to highly expressive models?

Outlier-Robust Neural Network Training

What did we do?

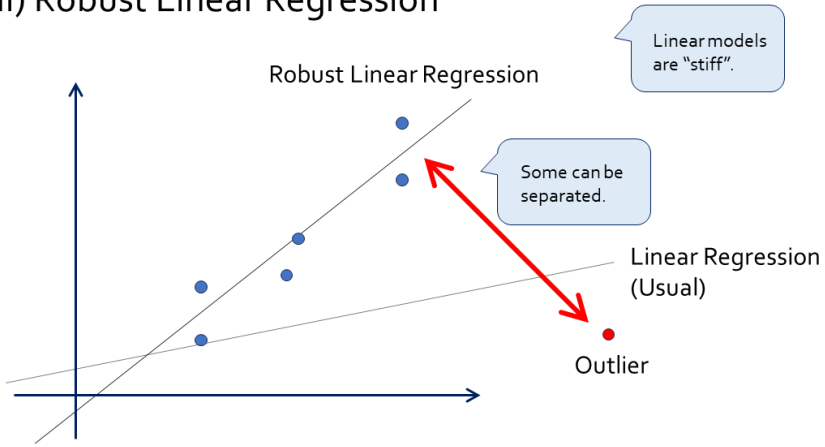
- We proposed an outlier-robust and flexible regression method.



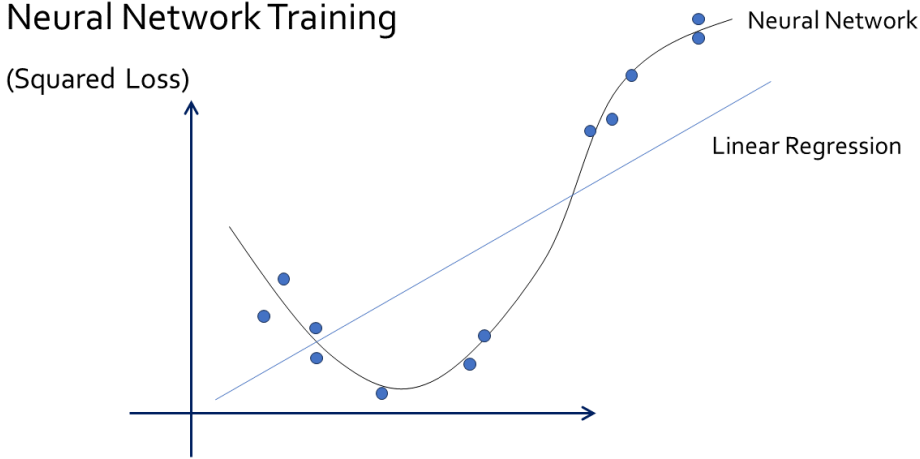
(a) Linear+Huber (b) NN+Huber (c) NN+Tukey (d) **Proposal**

Figure: $\mathbb{E}[Y \mid X = x] = f_*(x) = \sin(2x_1)\cos(2x_2)$.

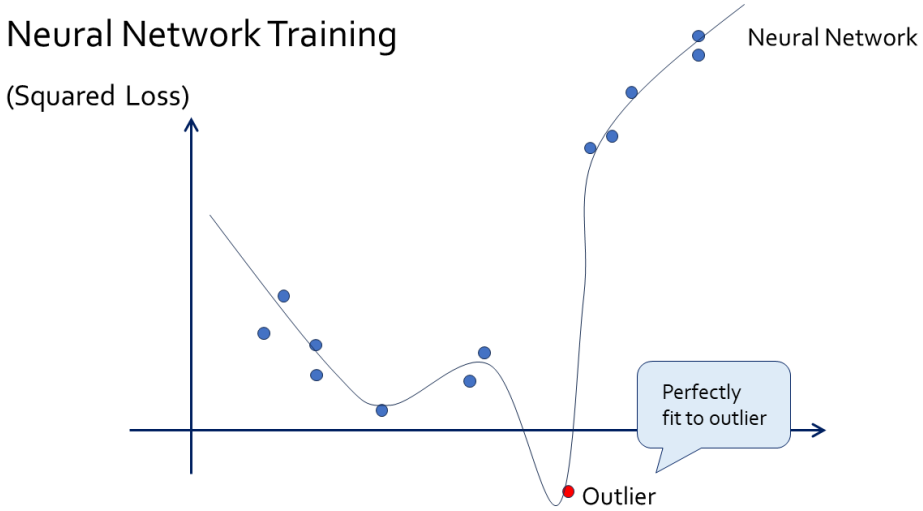
(Typical) Robust Linear Regression



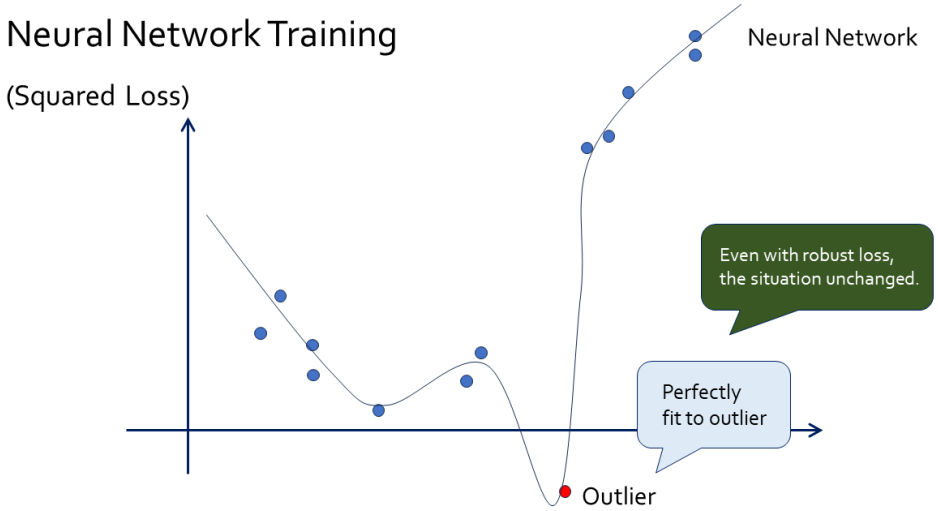
Neural Network Training



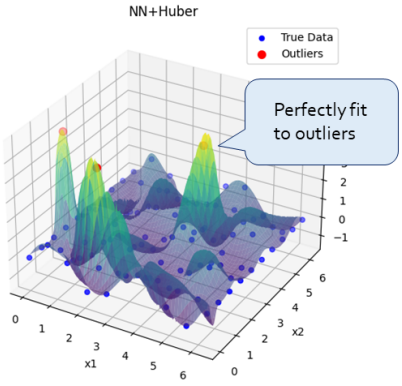
Neural Network Training

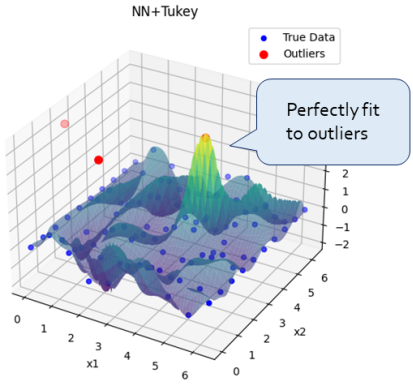


Neural Network Training

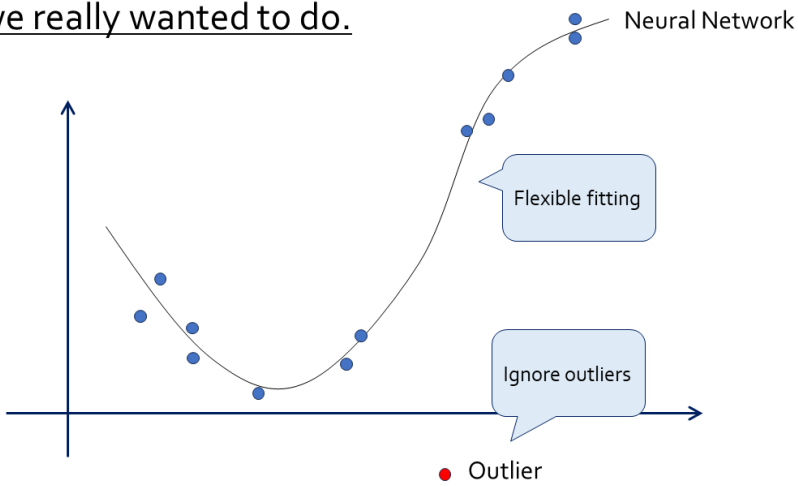


Actual Robust Loss + NN (+Gradient Descent)





What we really wanted to do.



So... what?

- We want the regression model $f_\theta(x)$ to be nicely flexitle ("wiggly"),
- but not excessively so.

We want to incorporate something like "stiffness"
into the regression function.

Regularization of Function Variation

Higher-Order Variation Regularization (HOVR; Okuno, arXiv:2308.02293v1):

$$C_{k,q}(f_\theta) := \int_\Omega \left| \frac{\partial^k f_\theta(x)}{\partial^k x} \right|^q \mathrm{d}x.$$

For example, in the case of a regression function $f_\theta(x) = \sum_j \theta_j \phi_j(x)$ using an orthonormal basis,

$$C_{k,2}(f_\theta) \lesssim \|\theta\|_2^2$$

can be interpreted as a generalization of parameter regularization.

- The derivatives can be computed using "autograd".
- The integration can be avoided probabilistically (without explicit computation): use a "stochastic" gradient method.

In other words?

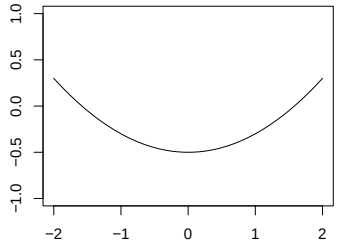


Figure: $C_{2,2}(f) \approx 0.64$

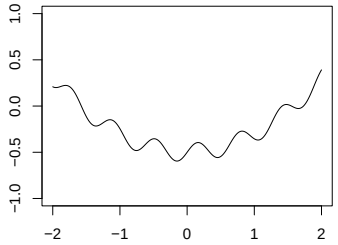


Figure: $C_{2,2}(f) \approx 197$

- Can suppress excessive variations.
- Corresponds to parameter regularization in linear/kernel regression models.
- For neural networks: **This study**.

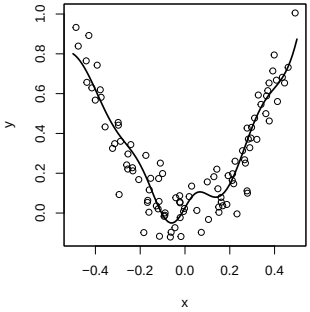


Figure: Weight decay (i.e., ridge)

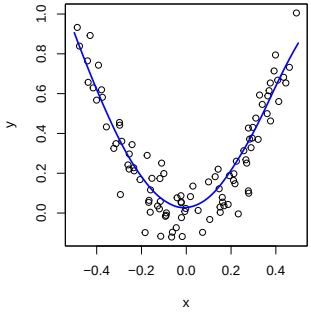


Figure: HOVR

- Akifumi Okuno, "A stochastic optimization approach to train non-linear neural networks with regularization of higher-order total variation", arXiv:2308.02293v1.
- By using stochastic gradient methods,
  integral-type regularization can be minimized exactly.

Classical Robust "Trimmed" Loss

Let the residuals be defined as $r_i(\theta) := y_i - f_\theta(x_i)$, and reorder the indices such that

$$|r_{(1;\theta)}(\theta)| \leq |r_{(2;\theta)}(\theta)| \leq \cdots \leq |r_{(n;\theta)}(\theta)|.$$

Then,

$$T_h(r(\theta)) = \frac{1}{n} \sum_{i=1}^{h} r_{(i;\theta)}(\theta)^2$$

is called the Trimmed Loss, which is known as a robust loss function.

- Outliers are expected to be "trimmed".

Our Idea

- Wouldn't it be a good idea to combine them...?
- Regularization prevents excessive variations, while robust loss ignores outliers.

$$\text{Loss function:} \quad \underbrace{\frac{1}{2} T_h(r(\theta))}_{\text{Robust Loss}} + \underbrace{\lambda \cdot C_{k,q}(f_\theta)}_{\text{"stiffen" the function}}$$

As will be discussed,

- The breakdown point (a traditional robust metric) is proved to be high.
- Efficient optimization algorithms can be developed.

Breakdown Point Analysis

Let $\widetilde{Z}^{(m)}$ be a dataset obtained by replacing m pairs (x_i, y_i) in $Z = \{(x_i, y_i)\}_{i=1}^n$.
The breakdown point is defined as:

$$\mathcal{E}^*_{k,q}(f_{\hat{\theta}}, Z) := \min \left\{ \frac{m}{n} \;\middle|\; \sup_{\widetilde{Z}^{(m)}} C_{k,q}(f_{\hat{\theta}(\widetilde{Z}^{(m)})}) = \infty \right\}.$$

Extension of Alfons et al. (2013) to Our Setting

$$\mathcal{E}^*_{k,q}(f_{\hat{\theta}}, Z) \geq \frac{n-h+1}{n}.$$

- In other words, the method can tolerate up to $n-h$ outliers in the data.
- The above theorem holds even for nonlinear models such as neural networks.

Efficient Optimization

- With an "auxiliary" parameter ξ, the Trimmed Loss T_h can be separated from the parameter θ.
- Co-author of this paper, Dr. Yagishita, proposed this separation recently:

Yagishita (arXiv:2410.04554)

$$\frac{1}{2}T_h(r(\theta)) = \underbrace{\min_{\xi\in\mathbb{R}^n}\left\{\frac{1}{n}\|r(\theta)-\xi\|_2^2 + T_h(\xi)\right\}}_{\text{Transformed Trimmed Loss (TTL)}}$$

- This transformation makes the optimization process easier.

Combination of HOVR and Trimmed Loss

$$\underbrace{\frac{1}{2}T_h(r(\theta))}_{\text{Robust Loss}} + \underbrace{\lambda \cdot C_{k,q}(f_\theta)}_{\text{stiffen the function}} = \min_{\xi}\{U_\lambda(\theta,\xi) - V_h(\xi)\}$$

where

$$U_\lambda(\theta,\xi) = \frac{1}{n}\{\|r(\theta)-\xi\|_2^2 + \|\xi\|_2^2\} + \lambda C_{k,q}(f_\theta) \text{ is Nonconvex and smooth,}$$
$$V_h(\xi) = \frac{1}{n}\|\xi\|_2^2 - T_h(\xi) \text{ is Convex and nonsmooth.}$$

In other words, minimizing Trimmed Loss + HOVR corresponds to minimizing

Augmented and Regularized Trimmed Loss (ARTL)

$$F_{h,\lambda}(\theta,\xi) = U_\lambda(\theta,\xi) - V_h(\xi)$$

with respect to the augmented parameters (θ,ξ).

Proposed Optimization Algorithm

- Let $u_\lambda^{(t)}(\theta^{(t)}, \xi^{(t)})$ be an unbiased stochastic gradient of $U_\lambda(\theta)$,
- Let $v_h(\xi)$ be a subgradient of $V_h(\xi)$.
- Define $g_{h,\lambda}^{(t)}(\theta, \xi) = u_\lambda^{(t)}(\theta, \xi) - (0, v_h(\xi))$.

Stochastic Gradient-Supergradient Descent (SGSD)

$$(\theta^{(t+1)}, \xi^{(t+1)}) \leftarrow (\theta^{(t)}, \xi^{(t)}) - \omega_t g_\lambda^{(t)}(\theta^{(t)}, \xi^{(t)}).$$

- This method overcomes issues related to integral terms and non-differentiable points.
- Exact convergence can be demonstrated.

Details Omitted, But...

- Differentiability with respect to θ is assumed (e.g., Sigmoid activation).
- Several assumptions, such as unbiasedness of the gradient, are imposed.

A Simple Form of the Convergence Theorem

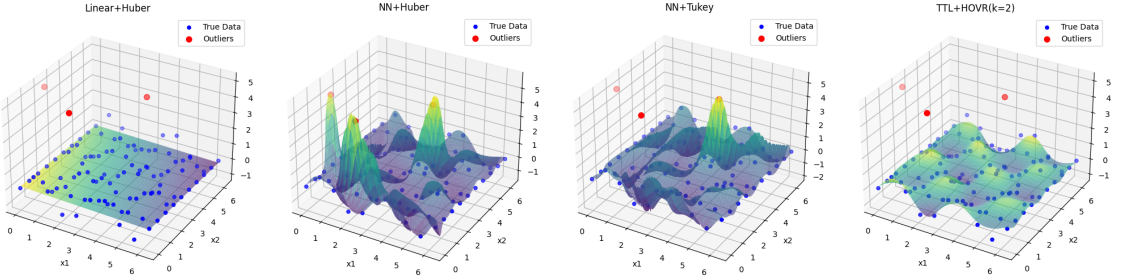
If the learning rate is set as $\omega_t = \alpha(1+t)^{-1/2}$, then

$$\mathbb{E}\left[\inf_{v \in \partial V_h(\xi)} \left\| \frac{\partial U_\lambda(\theta^{(\tau_T)}, \xi^{(\tau_T)})}{\partial(\theta, \xi)} - (0, v) \right\|_2^2\right] = \mathcal{O}(T^{-1/4}(\log T)^{1/2}).$$

- τ_T is a randomly chosen stopping time.
- The gradient converges to 0 (stationary point) except for the freedom in the subgradient.

Going Back to the First Example

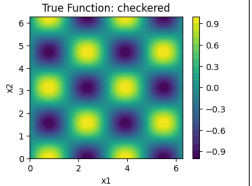
- A multi-layer perceptron with (input) 2 dimensions - 100 - 100 - 100 - (output) 1 dimension.
- Over 20,000 parameters with 3% outliers introduced.



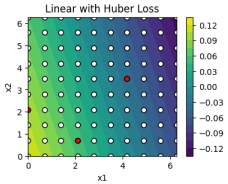
(a) Linear+Huber (b) NN+Huber (c) NN+Tukey (d) **Proposal**

Figure: $f_*(x) = \sin(2x_1)\cos(2x_2)$.

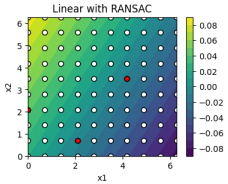
Type 1: checkered



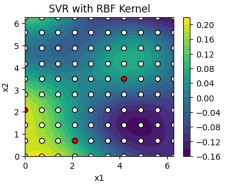
(a) Groundtruth



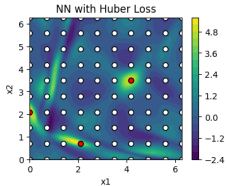
(b) Linear+Huber



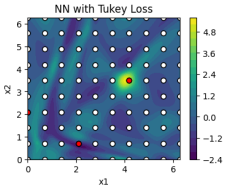
(c) Linear+RANSAC



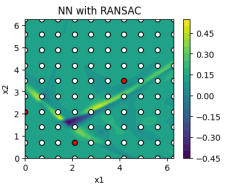
(d) SVR+RBF



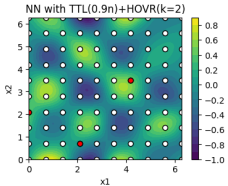
(e) NN+Huber



(f) NN+Tukey

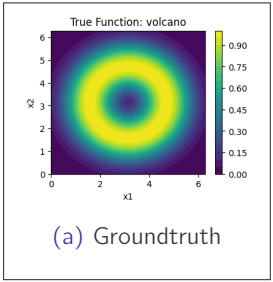


(g) NN+RANSAC

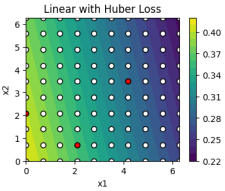


(h) **Proposal**

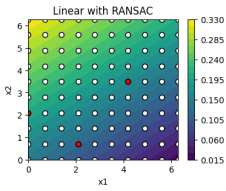
Type 2: volcano



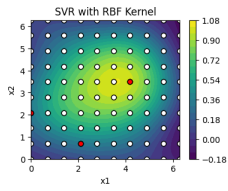
(a) Groundtruth



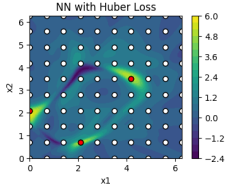
(b) Linear+Huber



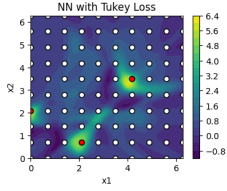
(c) Linear+RANSAC



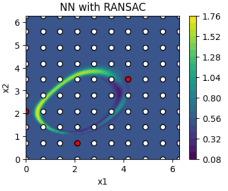
(d) SVR+RBF



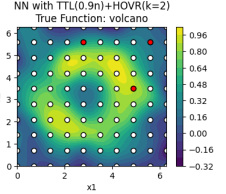
(e) NN+Huber



(f) NN+Tukey

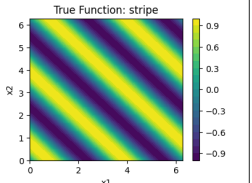


(g) NN+RANSAC

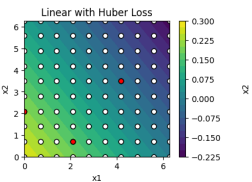


(h) **Proposal**

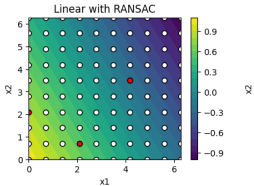
Type 3: stripe



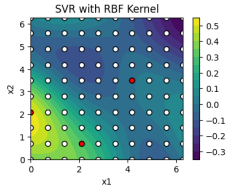
(a) Groundtruth



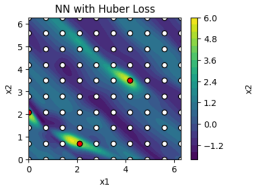
(b) Linear+Huber



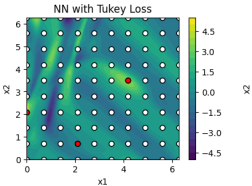
(c) Linear+RANSAC



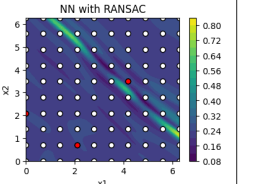
(d) SVR+RBF



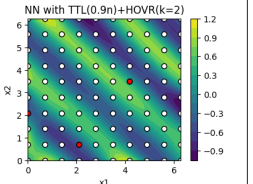
(e) NN+Huber



(f) NN+Tukey

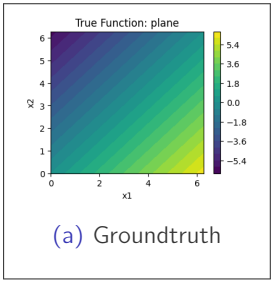


(g) NN+RANSAC

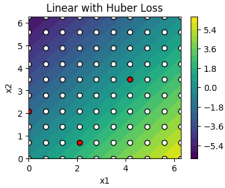


(h) **Proposal**

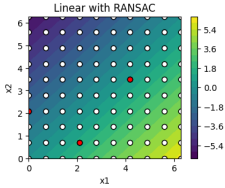
Type 4: plane



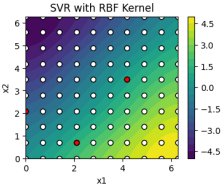
(a) Groundtruth



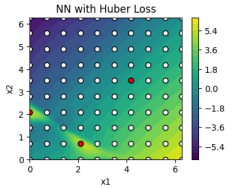
(b) Linear+Huber



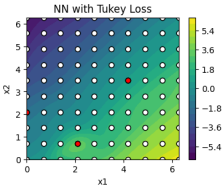
(c) Linear+RANSAC



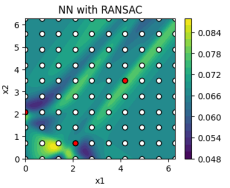
(d) SVR+RBF



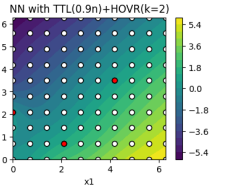
(e) NN+Huber



(f) NN+Tukey



(g) NN+RANSAC

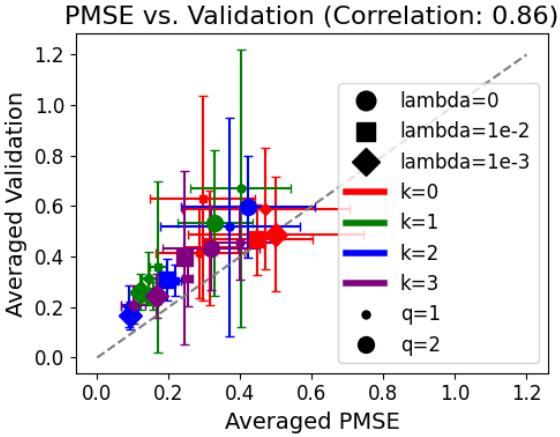


(h) **Proposal**

	Non-linear			Linear
	checkered	volcano	stripe	plane
Linear Reg. with Huber's Loss	0.124 (0.004)	0.130 (0.003)	0.498 (0.016)	0.001 (0.001)
Linear Reg. with RANSAC	0.140 (0.013)	0.186 (0.058)	0.871 (0.277)	**0.001** (0.001)
Support Vector Reg. with RBF Kernel	0.127 (0.008)	0.113 (0.020)	0.508 (0.031)	0.006 (0.002)
NN with Huber's Loss	0.634 (0.608)	1.031 (0.861)	0.488 (0.475)	0.043 (0.025)
NN with Tukey's Loss	0.458 (0.655)	0.413 (0.630)	0.304 (0.395)	0.017 (0.009)
NN with Label Noise Reg.	1.155 (1.068)	0.872 (0.659)	0.561 (0.498)	0.756 (0.945)
NN with RANSAC	0.160 (0.036)	0.142 (0.009)	0.527 (0.018)	0.011 (0.016)
NN with ARTL ($h = 0.9n, k = 1$)	0.088 (0.090)	0.082 (0.076)	0.223 (0.238)	0.010 (0.004)
NN with ARTL ($h = 0.9n, k = 2$)	**0.061** (0.016)	**0.040** (0.029)	**0.119** (0.047)	0.007 (0.001)

Parameter Selection

- Cross-Validation using Robust Trimmed Loss



Summary

- Even if the loss function is robust, neural networks may overfit to outliers.
- Proposed HOVR "stiffen" expressive models including neural networks.
- Proposed SGSD efficiently minimizes the ARTL, that combines the trimmed loss and variation regularization.

My personal webpage: `https://okuno.net`