Proceedings of Machine Learning Research 101:868–882, 2019 ACML 2019

Adaptive truncated residual regression for fine-grained regression problems

Hirotaka Hachiya[†‡] HIROTAKA.HACHIYA@RIKEN.JP
Yu Yamamoto[†‡] YU.YAMAMOTO@RIKEN.JP
Kazuro Hirahara[†] KAZURO.HIRAHARA@RIKEN.JP
Naonori Ueda[†] NAONORI.UEDA@RIKEN.JP
† *Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan*

‡ *Graduate School of System Engineering, Wakayama University, Wakayama, Japan*

Editors: Wee Sun Lee and Taiji Suzuki

Abstract

Recently, anchor-based regression methods have been applied to challenging regression problems, e.g., object detection and distance estimation, and greatly improved those performances. The key idea of anchor-based regression is to solve the regression of the residuals between selected anchors and original target variable, where the variance is expected to be smaller. However, similar to an ordinary regression method, the anchor-based regression could face difficulty on a fine-grained regression and ill-posed problems where the residual variables tend to be too small and complicated to accurately predict. To overcome these problems on the anchor-based regression, we propose to introduce an adaptive residual encoding in which the too small residual is magnified, and the too-large residual is truncated using adaptively tuned sigmoidal function. Our proposed method, called ATR-Nets (Adaptive Truncated Residual-Networks) with an end-to-end architecture could control the range of the target residual to be fitted based on the regression performance, Through experiments with toy-data and the system identification for earthquake asperity models, we show the effectiveness of our proposed method.

Keywords: neural network, anchor based regression, residual encoding and decoding, system identification

1. Introduction

Recently, anchor-based regression in Ren et al. (2015); Redmon and Farhadi (2017); Hachiya et al. (2018) have been applied to challenging regression problems, e.g., object detection and distance estimation. In these regression problems, the variance of target variables tend to be prohibitively large due to the large variation of the combination of target variables; for example, the combination of row-column coordinates, width, and height of bounding boxes is tremendously large. In the anchor-based regression, to overcome this large variance problem, the regression problem is reformulated to the combination of the anchor-selection and the residual regression.

More specifically, an ordinary regression method is performed to minimize the squared loss of $\mathcal{L}_{\mathrm{reg}}(\theta)$ as

$$\min_{\theta} \mathcal{L}_{\mathrm{reg}}(\theta) \equiv \mathbb{E}_{\mathbf{x},y}\left[\left(y - f_{\theta}(\mathbf{x})\right)^2\right] \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{d_x}$ is an input vector following independently and identical distribution, i.e., $\mathbf{x} \overset{\text{i.i.d}}{\sim} P(\mathbf{x})$ and y is an output scalar [1] of the target function $f(\mathbf{x})$, i.e., $y = f(\mathbf{x})$, and $f_\theta(\mathbf{x})$ is a model of the target function, e.g., neural network, with a training parameter vector θ.

Meanwhile, the anchor-based method is performed to minimize the squared residual loss of $\mathcal{L}_{\text{res}}(\theta)$ as

$$\min_\theta \mathcal{L}_{\text{res}}(\theta) \equiv \mathbb{E}_{\mathbf{x},y}\left[\left(y - \widehat{y_c} - R_\theta(\mathbf{x}, y_c)\right)^2\right] \tag{2}$$

where $\widehat{y_c}$ is a selected candidate for the target output y and $R_\theta(\cdot)$ is a *residual regressor*, respectively. This candidate $\widehat{y_c}$ is called *anchor*. That is, the goal of anchor-based regression is to accurately predict the residual r defined as

$$r \equiv y - \widehat{y_c}. \tag{3}$$

With a well-designed anchor, $\widehat{y_c}$, the variation of the residual r would become smaller than the one of the original target output y and thus the anchor-based regression would be expected to perform well.

However, similarly to an ordinary regression, the anchor-based regression faces difficulties as follows:

1. when the regression problem is *ill-posed* where multiple outputs y exist for a specific input $\mathbf{x}$, e.g., due to noises. Training the residual regression $R_\theta(\cdot)$ could become also unstable since different residuals r would exist for a specific anchor and result in a large variance even in residuals

2. for a fine-grained regression problem where the target variable y changes slightly e.g, in the third decimal place, training $R_\theta(\cdot)$ becomes difficult since the residual r and its resulting gradient would be too small to update parameters in a real-time

In this paper, to overcome these difficulties, we propose to adaptively encode the residual r into the range of $[0, 1]$ using a sigmoidal function $r_{\text{at}} = \sigma_\alpha(r) = \frac{1}{1+\exp(\alpha r)}$ where the steepness parameter α of the function is optimized based on the performance of regression. We have following two advantages:

1. large residuals caused by ill-posed problem, e.g., due to noises affecting the regression performance, would be compromised to ignore, e.g., $r_{\text{at}} = \sigma_{\alpha=10}(r \leq -3) \approx 0$ and $\sigma_{\alpha=10}(r \geq 3) \approx 1$.

2. Small residual changes, e.g., in the third decimal place are magnified into the change in $[0, 1]$, e.g., $\sigma_{\alpha=10}(r = 0.01) = 0.525$ and $\sigma_{\alpha=10}(r = 0.02) = 0.55$.

We evaluate our proposed method, ATR-Nets, with a toy problem and a system identification problem. The toy problem provides artificial ill-posed problems by intentionally adding noises, and we analyze the robustness of our proposed method against it. The system identification problem is related to the simulation of Nankai trough earthquakes in which the asperity between oceanic plates on the west side of Japan is mathematically modeled

1. For simplicity, we use a scalar value for the output of the target function $f(\mathbf{x})$—we can easily extend it to be a vector value.

in Hori et al. (2004); Hori (2006); Hyodo et al. (2016). It is required to accurately estimate the asperity parameters to reproduce the historical sequence of earthquakes in simulation to forecast next megaquakes. This system identification problem requires the fine-grained regression and we show the effectiveness of our proposed method.

2. Related Works

2.1. Anchor-based regression in object detection

Anchor-based regression has greatly advanced the performance of object detection tasks, e.g., in Faster R-CNN by Ren et al. (2015) and Yolov2 by Redmon and Farhadi (2017). In Faster R-CNN, a variety of candidates for bounding boxes (BBs) with different shapes and sizes are manually prepared. The anchors of BBs, 4 dimensional vectors of row-column coordinates, width, and height, are given a score on how likely these contain target objects in *region proposal network* and then following neural networks predict the residual between highly scored anchors and the ground-truth BBs. Meanwhile, in Yolov2, the anchors of BBs have generated automatically though k-means clustering, and then the confidence and residual of each anchor are estimated at the same time through a single convolutional neural network. The anchor-based regression for object detection is further extended to the distance estimation in Hachiya et al. (2018) using $2.5D$ anchors where the depth candidate, the distance from the camera, is added to 2D anchors for BBs, i.e., 5 dimensional vectors.

Overall, many studies are showing anchor-based regression is a promising approach for object detection and 3D localization from images. However, as discussed in Sec. 1, the anchor-based regression tends to fail in the ill-posed regression problem and fine-grained problem.

2.2. Piece-wise Linear Regression Method

The piece-wise linear model has been long studied to solve a nonlinear fitting problem in Dantzig (1963); Choi and Farrell (2000); Trecate and Muselli (2002). Recently, piece-wise linear method neural networks are shown to reduce the variance of target variables by dividing a nonlinear model into several linear models in Bagnall et al. (2015); Ji et al. (2018). For example, the target variable y could be approximated by the sum of linear sub-functions $\widehat{y} = \Sigma_{i=1}^{n} w_i f_i(x_i)$, where w_i is the weight for i-th sub-function $f_i(x_i)$ and the domain $\mathbf{x} \in \mathbb{R}_x^d$ assumes to be known. However, as discussed in Sec. 1, even in a piece-wise linear model which has a similar concept with anchor-based regression, tends to fail in the ill-posed regression problem and fine-grained problem.

3. Proposed Method: ATR-Nets

To overcome two problems: ill-posed and fine-grained, in this section, we firstly define a generalized anchor-based regression and secondly extend it to adaptively truncated residual-networks (ATR-Nets).

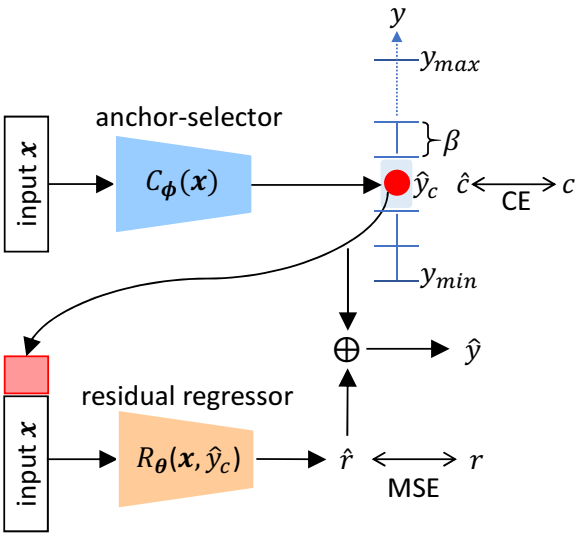


Figure 1: Diagram of the structure of a generalized anchor-based regression consisting of anchor-selector $C_\phi(\mathbf{x}) \in \mathbb{R}^{d_c}$ and residual regressor $R_\theta(\mathbf{x}, \widehat{y_c}) \in \mathbb{R}$. CE and MSE indicate the cross-entropy loss $\mathcal{L}_{\text{cls}}$ in Eq. 6. the mean squared error $\mathcal{L}_{\text{res}}$ in Eq. 2.

3.1. Generalized anchor-based regression

Fig. 1 depicts the structure of a generalized anchor-based regression. As the figure shows our generalized anchor-based regression consists of two-networks, anchor selector $C_\phi(\mathbf{x}) \in \mathbb{R}^{d_c}$ and residual regressor $R_\theta(\mathbf{x}, \widehat{y_c}) \in \mathbb{R}$. More specifically, anchors here are designed to be the center values of intervals of the target variable y defined as

$$y_c = y_{\min} + (c-1)\beta + \frac{\beta}{2} \tag{4}$$

where y_c is the center value of the c-th interval, $y_{\min}$ is the minimum value of the target variable y, and β is a hyper parameter representing the fixed width of the interval. β is usually set to a divisible number of $y_{\max} - y_{\min}$ where $y_{\max}$ is the maximum value of y. Thus, the number of intervals is $d_c = \frac{y_{\max} - y_{\min}}{\beta}$.

Given an input vector $\mathbf{x}$, the anchor-selector $C_\phi(\mathbf{x})$ computes the scores of d_c intervals and the interval $\widehat{c}$ with the maximum score is selected as

$$\widehat{c} = \underset{c}{\text{argmax}}\ C_\phi(\mathbf{x})[c] \tag{5}$$

where $C_\phi(\mathbf{x})[c]$ indicates the c-th element of the output vector of $C_\phi(\mathbf{x})$. Then, the anchor $\widehat{y_c}$ of the target output y is computed using Eq. 4. $C_\phi(\mathbf{x})$ is trained so as to minimize the cross-entropy loss as

$$\min_{\phi}(\phi)\mathcal{L}_{\text{cls}} \equiv \mathbb{E}_{\mathbf{x},c}\big[-\sum_{c=1}^{d_c} p(c|\mathbf{x}) \log(C_\phi(\mathbf{x})[c])\big] \tag{6}$$

Given the anchor $\widehat{y_c}$, the residual regressor $R_\theta(\mathbf{x}, \widehat{y_c})$ in Eq. 2 predicts $\hat{r}$ for the ground-truth residual r defined in Eq. 3.

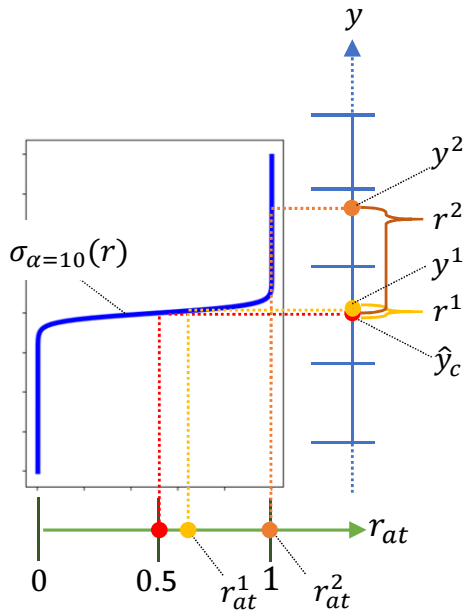


Figure 2: Diagram of the process of residual encoding. Vertical (in blue) and horizontal (in green) axes correspond to the output y and encoded residual r_{at} respectively. The red point at y-axis indicates the selected anchor $\widehat{y_c}$ and yellow and orange points correspond to two different ground-truth targets with smaller residual r^1 and with bigger residual r^2 respectively. The steepness parameter in this example is set to $\alpha = 10$.

3.2. Adaptive encoding of residuals

We extend the generalized anchor-based regression in Fig. 1 by introducing an adaptive encoding of residual r to alleviate the influence of multiple large residuals caused by the ill-posed problem and to magnify the fine-grained residuals as discussed in section 1. To this purpose, we encode the residual using a sigmoidal function $\sigma_\alpha(r)$ as follows:

$$r_{\text{at}} = \sigma_\alpha(r) = \frac{1}{1 + \exp(-\alpha r)} \tag{7}$$

where α is the steepness parameter and r_{at} is the encoded residual.

To more visually explain, Fig. 2 depicts the diagram of the process of residual encoding. In this figure, a sigmoidal function $\sigma_\alpha(r)$ is located at the selected anchor $\widehat{y_c}$ indicated by the red dot on the vertical output y axis. For a ground-truth value y^1 which is by accident close to the anchor $\widehat{y_c}$, the small residual, e.g., $r^1 = 0.01$ is encoded to a magnified residual value, e.g., $r^1_{\text{at}} = 0.525$ where the third decimal change is magnified to the second one and thus the residual regression by $R_\theta(\mathbf{x}, \widehat{y_c})$ could be alleviated. Meanwhile, for a ground-truth value y^2 which is far from the anchor $\widehat{y_c}$, the large residual $r^2 = 3$ is encoded to truncated residual value, e.g., $r^2_{\text{at}} \approx 1$. That is, the influence of the large residual r^2 in the training $R_\theta(\mathbf{x}, \widehat{y_c})$ could be limited. The steepness parameter α controls the range of the training

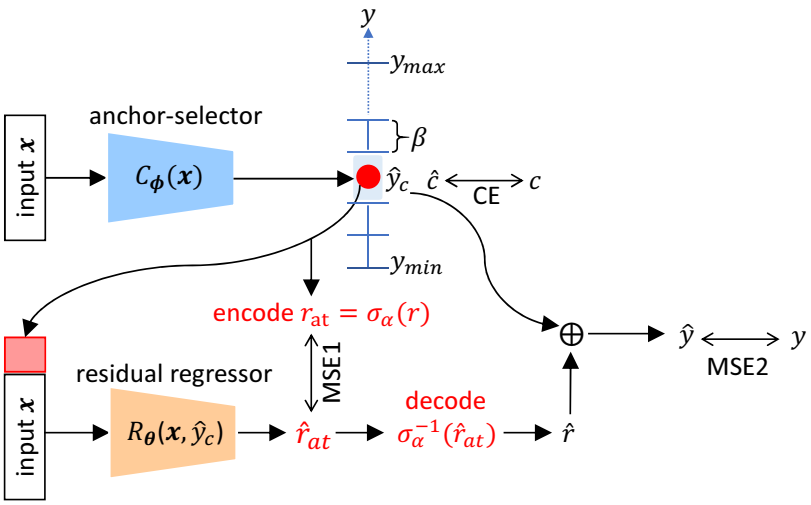


Figure 3: Diagram of structure of proposed method, adaptive truncated residual-networks (ATR-Nets) consisting of anchor-selector $C_\phi(\mathbf{x})$, adaptive residual encoder $\sigma_\alpha(r)$, residual regressor $R_\theta(\mathbf{x}, \widehat{y_c})$ and residual decoder $\sigma_\alpha^{-1}(\widehat{r_{\text{at}}})$. CE indicates the cross-entropy loss $\mathcal{L}_{\text{cls}}(\phi)$ in Eq. 6. MSE1 and MSE2 indicate the mean squared error for the encoded residual $\mathcal{L}_{\text{enc}}(\theta)$ and the output $\mathcal{L}_y(\phi, \theta, \alpha)$ in Eq. 10.

target of $R_\theta(\mathbf{x}, \widehat{y_c})$. That is, the smaller α is, the more gradual slope of sigmoidal function in Fig. 2—the range of the output y encoded r_{at} in $[0, 1]$ becomes wider and vice-versa.

After the adaptvely encoded residual $\widehat{r_{\text{at}}}$ is predicted by $R_\theta(\mathbf{x}, \widehat{y_c})$, it is decoded back to the original residual r using decode function $\sigma_\alpha^{-1}(\widehat{r_{\text{at}}})$ defined as

$$r = \sigma_\alpha^{-1}(\widehat{r_{\text{at}}}) = -\frac{1}{\alpha} \log\Big(\frac{1}{\widehat{r_{\text{at}}}} - 1\Big). \tag{8}$$

3.3. Architecture of ATR-Nets

Our proposed method, called adaptive truncated residual-networks (ATR-Nets) introduces adaptive residual-encoder $\sigma_\alpha(r)$ into anchor-based regression in Fig. 1. Fig. 3 depicts the entire structure of ATR-Nets, which consists of anchor-selector $C_\phi(\mathbf{x})$, adaptive-truncator $\sigma_\alpha(r)$, residual regressor $R_\theta(\mathbf{x}, \widehat{y_c})$ and decoder $\sigma_\alpha^{-1}(r_{\text{at}})$. To optimize the entire networks with adaptive truncation, we minimize two losses: the cross-entropy loss $\mathcal{L}_{\text{cls}}(\phi)$, i.e., CE in Fig. 3 and the following encoded residual loss $\mathcal{L}_{\text{enc}}(\theta)$, i.e., MSE1 in Fig. 3:

$$\min_\theta \mathcal{L}_{\text{enc}}(\theta) \equiv \mathbb{E}_{\mathbf{x},y}\big[\big(R_\theta(\mathbf{x}, y_c) - \sigma_\alpha(r)\big)^2\big]. \tag{9}$$

The final output of the ATR-Nets is $\widehat{y} = \widehat{y_c} + \sigma_\alpha^{-1}(\widehat{r_{\text{at}}})$ and its mean squared error, i.e., MSE2 in Fig. 3 is defined as follows:

$$\mathcal{L}_y(\phi, \theta, \alpha) \equiv \mathbb{E}_{\mathbf{x},y}\big[\big(y - \widehat{y_c} - \sigma_\alpha^{-1}(\widehat{r_{\text{at}}})\big)^2\big]. \tag{10}$$

Table 1: Detailed information of layers of neural networks. d_x is the dimension of input $\mathbf{x}$, d_h is the number of nodes in fully connected (fc) layers, d_c is the number of intervals of the output y splitted by the width β and L is number of layers.

Table 2: Layers of $C_\phi(\mathbf{x})$.

Layer Name	Num. of channels	activation
input	d_x	-
fc1	d_h	relu
fc2	d_h	relu
$\vdots$	$\vdots$	$\vdots$
fcL	d_c	-

We tune the steepness parameter α adaptively in the process of the training of the entire networks based on the final output error $\mathcal{L}_y$ as

$$\alpha \propto \mathcal{L}_y(\phi, \theta, \alpha) \tag{11}$$

where the larger final error leads to a steep sigmoidal function, i.e., large α, like in Fig. 2 and active magnification and truncation of residuals—even small residuals would be encoded to $[0, 1]$. In this situation, the contribution of the residual regressor $R_\theta(\theta)$ becomes low and the entire system of ATR-Nets relies on the anchor $\widehat{y_c}$ selected by $C_\phi(\phi)$.

Meanwhile, the smaller final error leads to a gradual sigmoidal function, i.e., small α and inactive magnification and truncation of residuals—even large residuals would be encoded to $[0, 1]$. In this situation, the residual regressor $R_\theta(\theta)$ actively corrects the residual of the anchor $\widehat{y_c}$.

Overall, our proposed ATR-Nets, adaptively tunes the steepness parameter α based on the final training error $\mathcal{L}_y$ and controls the contribution of the residual regressor; in other words, controls the combination between classifier (anchor-selector) and regressor (residual regressor).

3.4. Training of ATR-Nets

We design L-layer fully-connected (fc) neural networks for both $C_\phi(\mathbf{x})$ and $R_\theta(\mathbf{x}, \widehat{y_c})$ as shown in Table 2 and 3.

To train the neural networks, we prepare training data $\mathcal{D}_{\text{tr}}$ as follows:

$$\mathcal{D}_{\text{tr}} = \left\{\mathbf{x}^n, y^n, c^n\right\}_{n=1}^N \tag{12}$$

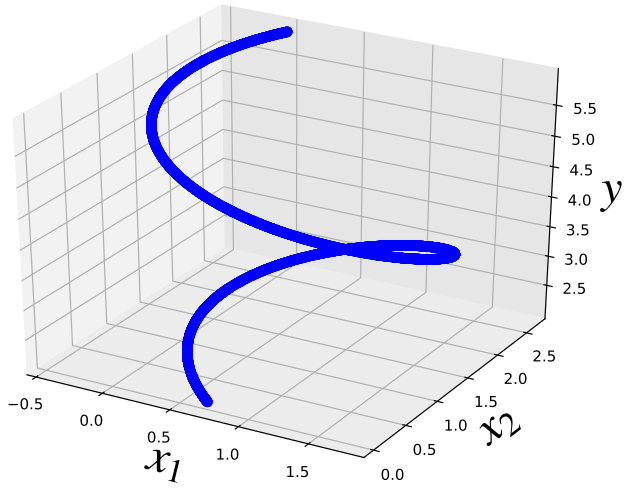


Figure 4: An example of toy-data, 3-dimensional spiral staircase data, when the number of rolling $m = 2$. The horizontal axes x_1 and x_2 are input variables and the vertical axis y is the output variable.

where N is the number of training data, $\mathbf{x}^n$, y^n and c^n are the n-th input vector, target output and β-width interval where y^n is located. We also prepare test data $\mathcal{D}_{\text{te}}$ which are not overlapped to the training data $\mathcal{D}_{\text{tr}}$ for the evaluation purpose.

4. Experimental evaluation on toy data

To evaluate our proposed method, ATR-Nets on the ill-posed problems discussed in Section 1 in comparison with ordinary regression in Eq. 1 and anchor-based regression in Fig. 1, we use artificial toy data, 3-dimensional spiral staircase data, as shown in Fig. 4.

4.1. Setting of toydata

The toy data in Fig. 4 are generated as follows:

$$\begin{aligned} y^n &\overset{\text{i.i.d}}{\sim} U(y_{\min}, y_{\max}) \\ x_1^n &= \sin(m \times y^n) + \frac{1}{\log(y^n)} \\ x_2^n &= \cos(m \times y^n) + \log(y^n) \end{aligned} \tag{13}$$

where (x_1^n, x_2^n) are the n-th two dimensional input vector, y^n is the n-th output, $U(y_{\min} = 2, y_{\max} = 6)$ is the uniform distribution in the range $[y_{\min}, y_{\max}]$ and m = 2 is the number of rolling of spiral staircase.
We randomly generated 10000 data $\{x_1^n, x_2^n, y^n, c^n\}$ using Eq. 13 and randomly select 80% of the data as training data $\mathcal{D}_{\text{tr}}$ and the rest as test data $\mathcal{D}_{\text{te}}$. We add random noise to the

output of y^n in the training data $\mathcal{D}_{\text{tr}}$ as

$$y^n = y^n + \mathcal{N}(0, \sigma^2), \quad y^n \in \mathcal{D}_{\text{tr}} \tag{14}$$

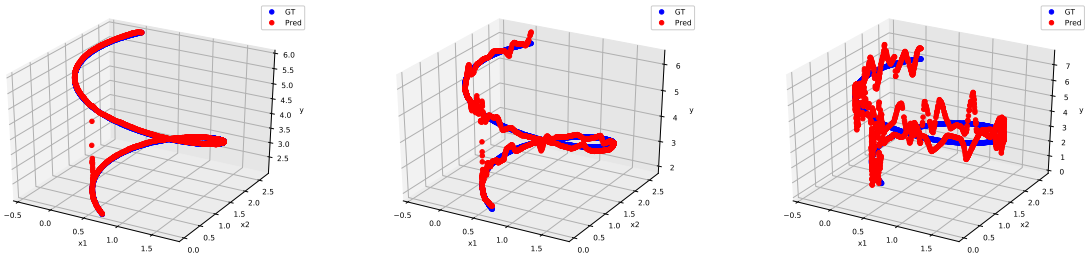
where σ is the manually tuned standard deviation. We randomly split training data $\mathcal{D}_{\text{tr}}$ into 10-minibatch each size of which is 800 every epoch in the training process.

4.2. Evaluation on toydata

We compare three methods: ordinary regression, anchor-based regression and our proposed ATR-Nets with the number of layers $L = 3$, the dimension of input $d_x = 2$, and the number of nodes $d_h = 128$. We train all methods using the same training data $\mathcal{D}_{\text{tr}}$ with different noise levels $\sigma \in \{0.0, 1.0, 5.0\}$ for 50000 iterations and evaluate using test data $\mathcal{D}_{\text{te}}$. As for anchor-based regression and ATR-Nets, we prepare two different numbers of intervals $d_c \in \{10, 20\}$ corresponding width-parameters $\beta \in \{0.4, 0.2\}$. As for the steepness parameter of ATR-Nets, we updated every 500 iteration using the final error $\widehat{\mathcal{L}_y}$ computed using a corresponding minibatch training data as

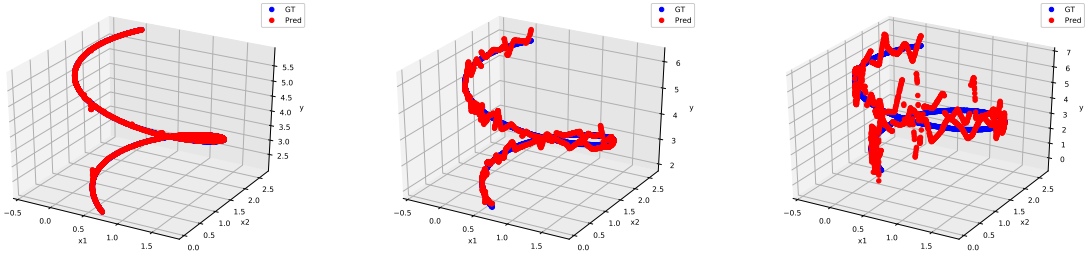
$$\widehat{\alpha}_{t+1} = \alpha_{\text{base}} \times \widehat{\mathcal{L}_y}(\phi_t, \theta_t, \alpha_t) \tag{15}$$

where t indicates the iteration number and α_{base} is a magnification ratio set at 1 or 10.



$\sigma = 0.0$ $\sigma = 1.0$ $\sigma = 5.0$

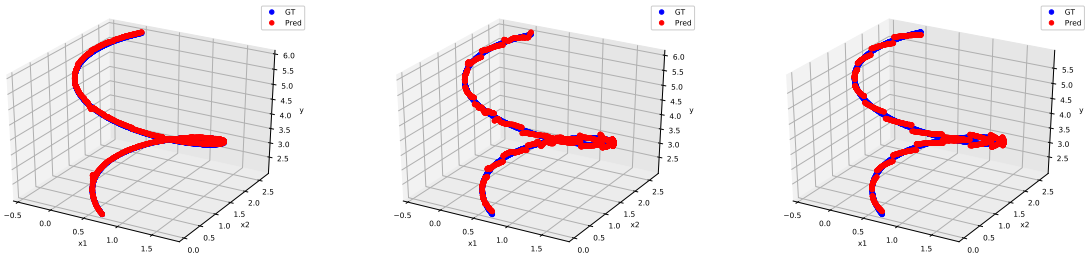
Figure 5: Examples of results by ordinary regression



$\sigma = 0.0$ $\sigma = 1.0$ $\sigma = 5.0$

Figure 6: Examples of results by anchor-based regression with $d_c = 20$

Tab. 4 depicts the mean squared error computed using test data $\mathcal{D}_{\text{te}}$ for three methods in three cases that noise levels in the training data $\mathcal{D}_{\text{tr}}$ are changed, $\sigma \in \{0.0, 1.0, 5.0\}$. The



$\sigma = 0.0$ $\sigma = 1.0$ $\sigma = 5.0$

Figure 7: Example of results of ATR-Nets with $d_c = 20$ and $\alpha_{\text{base}} = 10$

Table 4: Mean squared test error $\mathcal{L}_y$ (variance) on toy data for three methods: ordinary regression, anchor-based regression and our proposed method, ATR-Nets. The errors in bold indicate the minimum error.

noise level	$\sigma = 0.0$	$\sigma = 1.0$	$\sigma = 5.0$
ordinary regression	0.0141 (0.0404)	0.0178 (0.0225)	0.4942 (0.6251)
anchor-based $d_c = 10$	0.0248 (0.0131)	0.0252 (0.0671)	0.4868 (1.2745)
anchor-based $d_c = 20$	0.0036 (0.0178)	0.0245 (0.0575)	0.4595 (0.9096)
ATR-Nets $d_c = 10$, $\alpha_{\text{base}} = 10$	0.0173 (0.0920) $\alpha = 0.10$	0.0172 (0.0541) $\alpha = 9.86$	0.0171 (0.0274) $\alpha = 232.7$
ATR-Nets $d_c = 20$, $\alpha_{\text{base}} = 1$	0.0105 (0.0533) $\alpha = 0.10$	0.0230 (0.0647) $\alpha = 0.96$	0.0068 (0.0233) $\alpha = 23.28$
ATR-Nets $d_c = 20$, $\alpha_{\text{base}} = 10$	**0.0015** (0.0721) $\alpha = 0.10$	**0.0133** (0.0683) $\alpha = 10.34$	**0.0067** (0.0235) $\alpha = 232.9$

table shows that as the noise level changes, the performance of ordinary and anchor-based regression methods also drastically change. Especially, in the case that noise level is high, e.g., $\sigma = 5.0$, the error of these two methods become prohibitively large.

Fig. 5, 6 and 7 depict examples of predicts of three methods for test data $\mathcal{D}_{\text{te}}$. Fig. 5 shows the ordinary regression surfers ill-posed problem even without noise, $\sigma = 0.0$ where spiral stairs crosses at $(x_1 = 0.5, x_2 = 0.5)$ (see the left figure in Fig. 5 and thus multiple outputs y exists. Even if it does not affect much the numerical error in Tab. 4, this is a critical issue in a real-world application. In addition, both ordinary and anchor-based regression methods have unstable predictions in large noises, i.e., $\sigma \in \{1.0, 5.0\}$ due to the influence of ill-posed situation caused by training noises.

Meanwhile, our proposed method, ATR-Nets, shows stable and accurate performance for all noise levels as shown in Fig. 7 and Tab. 4, indicating that adaptive encoding of residuals well controls the balance of influences between anchor-selector $C_\phi(\phi)$ and residual regressor $R_\theta(\theta)$.

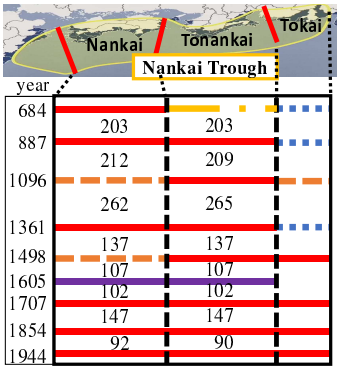


Figure 8: Historical records of megaquakes in Nankai Trough.

5. Application on system identification

We apply our proposed method, ATR-Nets on the fine-grained regression problem discussed in Section 1 for the system identification of the Nankai trough plate model.

5.1. Nankai Trough earthquake simulation

In southwest Japan, the Philippine Sea plate is subducting along the Nankai Trough, whereby megaquakes have repeatedly occurred, causing great disasters over southwest Japan. Fig. 8 is the sequence of megaquakes recorded in Nankai Trough consisting of three regions Nankai, Tonankai and Tokai over 1400 years. Based on friction laws between plates and interaction between cells, the megaquake cycles have been simulated to reproduce the historical record toward forecasting the occurrence of the next Nankai megaquake in Hori et al. (2004); Hori (2006); Kodaira et al. (2006); Nakata et al. (2014); Hyodo et al. (2016).

However, the Nankai megaquakes have occurred at irregular intervals with large variations and even the latest studies have not successfully reproduced such irregular recurrence intervals in Hyodo et al. (2016). In these studies, researchers have so far manually adjusted the frictional parameters controlling the recurrence intervals to reproduce such complex megaquake cycles in the simulator, which seems to have some limitations.

5.2. System identification using machine learning

In this study, we apply a machine learning approach to estimate the frictional parameters to reproduce the sequence of megaquakes. Firstly, let us express the megaquake cycle simulator by a function $F(\cdot)$ as

$$\{V_i\}_{i=1}^{C} = F\big(\{a_i, b_i, L_i\}_{i=1}^{C}\big) \tag{16}$$

where V_i is the generated sequence of slip velocities at cell i, C is the number of cells on the plate model, a_i, b_i and L_i are the friction parameters respectively. Then, we consider the inverse function $F^{-1}(\cdot)$ as

$$\{a_i, b_i, L_i\}_{i=1}^{C} = F^{-1}\big(\{V_i\}_{i=1}^{C}\big) \tag{17}$$

We approximate this inverse function using machine learning, e.g., neural networks.

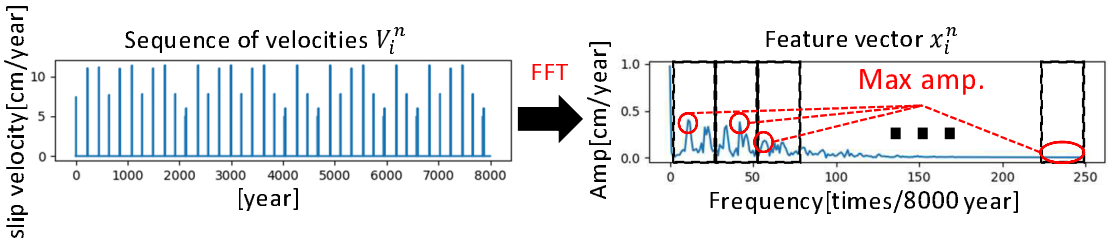


Figure 9: Feature extraction from time-series plate slip velocities.

5.3. Experimental setting used training data

In usual earthquake cycle simulations, the plate interface is divided into fine cells with the sizes less than critical nucleation sizes (Rice, 1993); however, it demands heavy computational costs and resources. Therefore, instead, as a developing stage of the method, we use a discrete cell model, which consists of 3 discrete cells corresponding to the fault rupture segments on the plate interface as Nankai, Tonankai and Tokai in Fig. 8. In addition, we consider only a frictional parameter b_i for each cell i, i.e., b_1, b_2 and b_3, and keeping other parameters fixed.

5.4. Data generation and feature extraction

We use 3-cell simulator of the Nankai megaquakes written in C-language and running on Windows workstation and generate pairs of the parameters and slip velocity as

$$\{\left(\{b_i^n\}_{i=1}^C, \{V_i^n\}_{i=1}^C\right)\}_{n=1}^N \tag{18}$$

Every each sample, one of b_1, b_2 and b_3 is incremented by 0.00005 and the sampling range of parameters are $[0.0125, 0.0170]$, $[0.0120, 0.0165]$ and $[0.0120, 0.0165]$ respectively—as a result, we generated $N = 753571$ samples. Then, we randomly split into $N = 678214$ training data $\mathcal{D}_{\text{tr}}$ and $N = 75357$ test data $\mathcal{D}_{\text{te}}$. Let us note that we use the same training and test data for all methods for a fair comparison.

As Fig. 9 depicts, a sequence of slip velocities V_i^n in each cell are converted to its frequency spectrum by fast Fourier transform (FFT), and then the maximum amplitude of FFT every 25Hz window up to 250(Hz). That is, 10 dimensional vector $\mathbf{x}_i^n$ is extracted from each cell and then vectors extracted from three cells are concatenated as input vector $\mathbf{x}^n$, i.e., $d_x = 30$. In addition, the output variable here is 3-dimensional vector, i.e., $\mathbf{y} = (b_1, b_2, b_3)$.

5.5. Evaluation on system identification

We compare three methods: ordinary regression, anchor-based regression and our proposed method, ATR-Nets.

We compare three methods, ordinary regression, anchor-based method and the proposed method, ATR-Nets. We use $d_c = 5$ layers on regression networks, e.g., $R_\theta(\mathbf{x}, \widehat{y}_c)$ and $f_\theta(\mathbf{x})$, and $d_c = 4$ layers on anchor-selector $C_\phi(\mathbf{x})$. Besides, for simplicity, we set a static value for $\alpha = 20$ for the residual encoder.

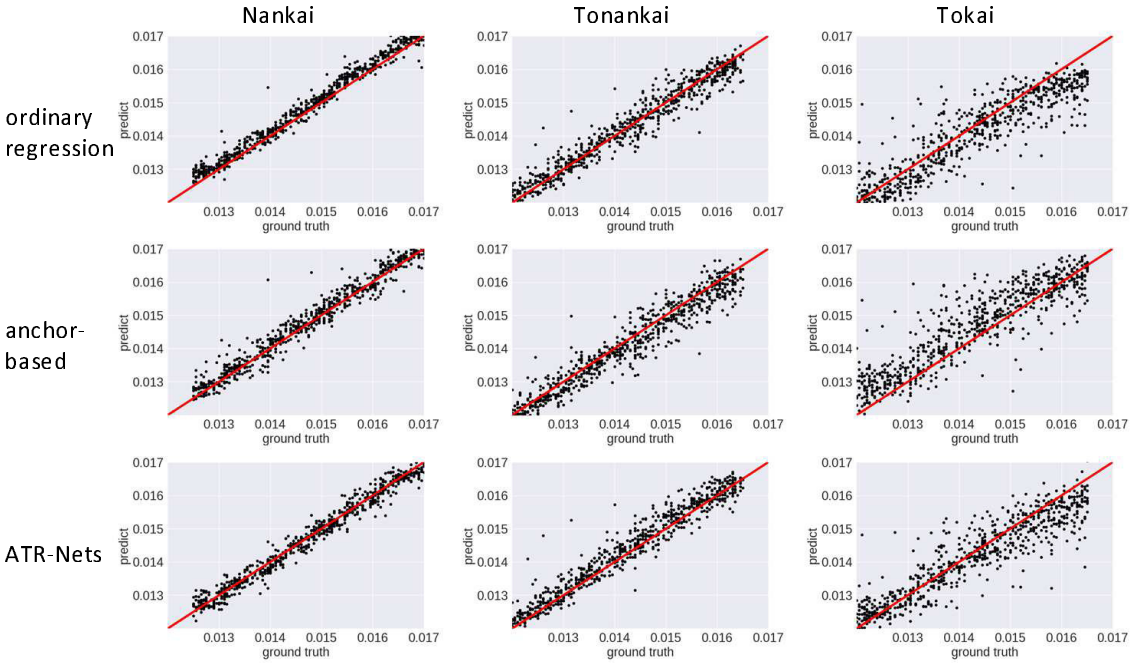


Figure 10: Ground-truth (horizontal) vs. predicted (vertical) friction values for test data $\mathcal{D}_{\text{te}}$ at each cell on Nankai megaquakes data. From the left, each column of graphs corresponds to the prediction by ordinary regression, anchor-based method with $d_c = 20$ and ATR-Nets with $d_c = 20$, respectively. For the clear visualization, only 100 data in $\mathcal{D}_{\text{te}}$ are plotted.

Table 5: Mean squared error and mean variance for the test data $\mathcal{D}_{\text{te}}$ on the Nankai megaquakes data for three methods: ordinary regression, anchor-based regression and our proposed method, ATR-Nets. The errors in bold indicate the best.

Method	Mean squared error (Mean variance)
ordinary regression	$4.96 \times 10^{-7} (1.48 \times 10^{-13})$
anchor-based $d_c = 20$	$6.30 \times 10^{-7} (2.36 \times 10^{-13})$
ATR-Nets $d_c = 20$	$\mathbf{4.29 \times 10^{-7}}$ (1.45×10^{-13})

Tab. 5 depicts the mean-squared error and mean variance for the predicted 3-dimensional output $\widehat{\mathbf{y}}$, and Fig. 10 depicts the ground-truth (horizontal axis) vs. the predicted (vertical axis) frictional value b_1, b_2 and b_3 for each cell and method.

As Tab. 10 shows ATR-Nets provides good performances similarly with the toy data sets.

6. Conclusion

In this paper, firstly we provided a general framework of anchor-based regression consisting of anchor-selector and residual regressor, which is expected to mitigate the regression problem with a large variance in the target variable. However, the anchor-based regression has two potential issues on ill-posed and fine-grained regression problems. To overcome these issues, we extended the anchor-based regression by introducing adaptive residual encoding and decoding, to adaptively magnify and truncate the residuals to be regressed based on the performance of the entire regression. Through experiments with toy data and system identification task of the earthquake plate models, we show the effectiveness of our proposed method, ATR-Nets.

References

Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with cote: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.

Jin Young Choi and Jay Farrell. Nonlinear adaptive control using networks of piecewise linear approximators. *IEEE Transactions on Neural Networks*, 11(2):390–401, 2000.

George Bernard Dantzig. *Linear programming and extensions*. Princeton University Press, 1963.

Hirotaka Hachiya, Yuki Saito, Kazuma Iteya, Masaya Nomura, and Takayuki Nakamura. 2.5d faster r-cnn for distance estimation. *ROBOMECH journal*, 5(22):1–13, 2018.

Takane Hori. Mechanisms of separation of rupture area and variation in time interval and size of great earthquakes along the nankai trough, southwest japan. *Journal of the Earth Simulator*, 5:8–19, 2006.

Takane Hori, Naoyuki Kato, Kazuro Hirahara, Toshitaka Baba, and Yohiyuki Kaneda. A numerical simulation of earthquake cycles along the nankai trough in southwest japan: lateral variation in frictional property due to the slab geometry controls the nucleation position. *Earth and Planetary Science Letters*, 228(3-4):215–226, 2004.

Mamoru Hyodo, Takane Hori, and Yoshiki Kaneda. A possible scenario for earlier occurrence of the next nankai earthquake due to triggering by an earthquake at hyuga-nada, off southwest japan. *Earth, Planets and Space*, 68(1):6, 2016.

Cun Ji, Shijun Liu, Chaenglei Yang, Li Pan, Lei Wu, and Xiangxu Meng. A shapelet selection algorithm for time series classification: new directions. *Procedia Computer Science*, 129:461–467, 2018.

Shuichi Kodaira, Takane Hori, Aki Ito, Seiichi Miura, Gou Fujie, Jin-Oh Park, Toshitaka Baba, Hide Sakaguchi, and Yoshiyuki Kaneda. A cause of rupture segmentation and synchronization in the nankai trough revealed by seismic imaging and numerical simulation. *Journal of Geophysical Research: Solid Earth*, 111(B9), 2006.

Ryoko Nakata, Mamoru Hyodo, and Takane Hori. Possible slip history scenarios for the hyuga-nada region and bungo channel and their relationship with nankai earthquakes in southwest japan based on numerical simulations. *Journal of Geophysical Research: Solid Earth*, 119(6):4787–4801, 2014.

Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

Giancarlo Ferrari Trecate and Marco Muselli. A new learning method for piecewise linear regression. In *International conference on artificial neural networks*, pages 444–449. Springer, 2002.