Gruppe: Willkommen1.0

Teinehmer/innen:

Yurong Li

Jiangtao Yu

Jiyuan Shan

Haowang Zhao

Guowei Liu

1.1 Univariate, Nonlinear Regression

1.1.1 Adapt the code to conduct a univariate, nonlinear regression of a sine curve similar to the slide deck 'VL__3-7' on ISIS, slides 12-16.

(1) The prediction of since curve used in 3 types of activation functions

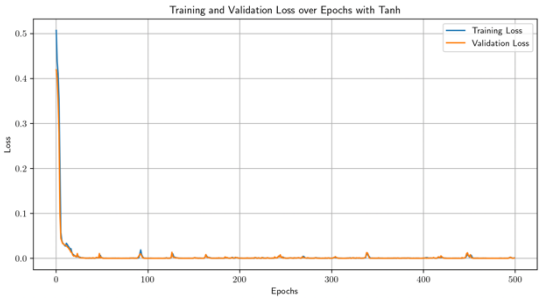


Figure 1.1.1-1 Training and Validation Loss over Epochs with Tanh

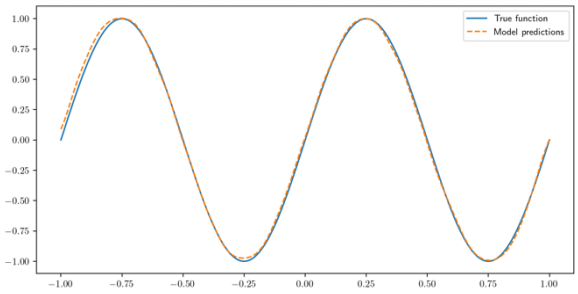


Figure 1.1.1-2 Comparison of True Function and Model Prediction

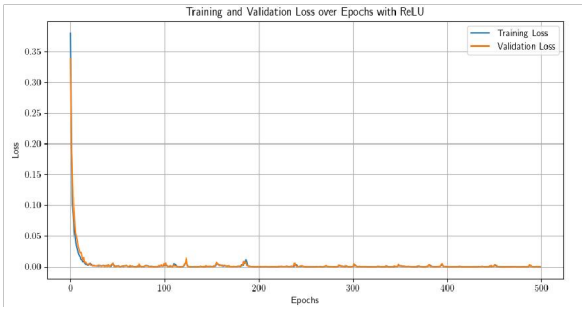


Figure 1.1.1-3 Training and Validation Loss over Epochs with ReLU

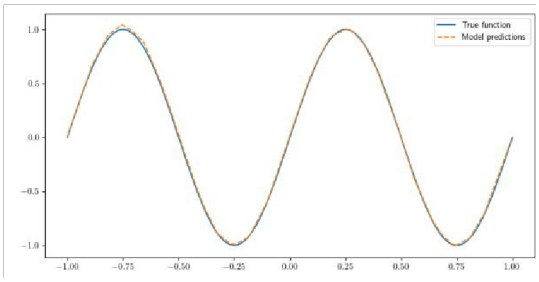


Figure 1.1.1-4 Comparison of True Function and Model Prediction

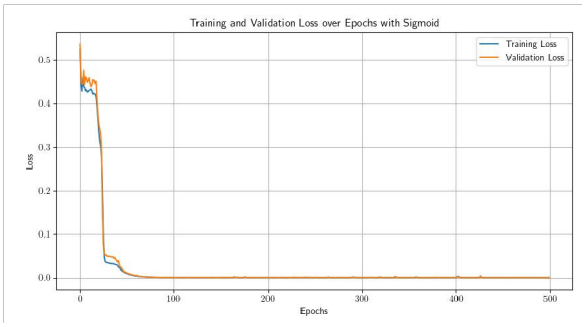


Figure 1.1.1-5 Training and Validation Loss over Epochs with Sigmoid

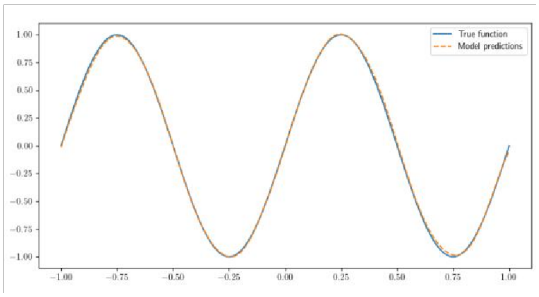


Figure 1.1.1-6 the Comparison of True Function and Model Prediction

(2) The prediction of sine Curve using Tanh activation function with different Learning rates

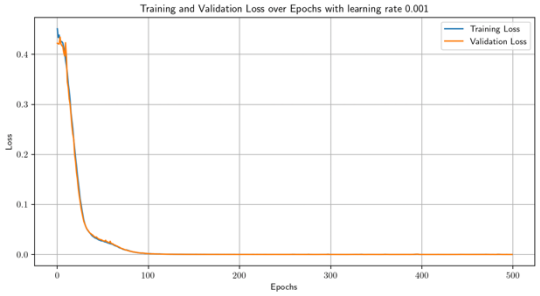


Figure 1.1.1-7 Training and Validation Loss over Epochs with Learning rate 0.001

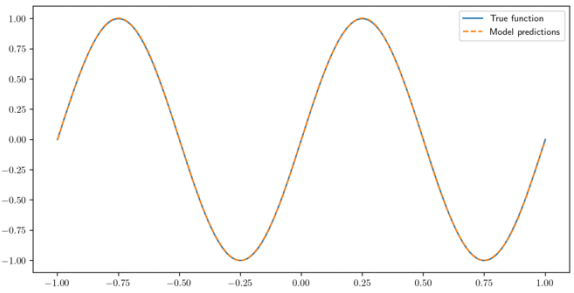


Figure 1.1.1-8 Comparison of True Function and Model Prediction

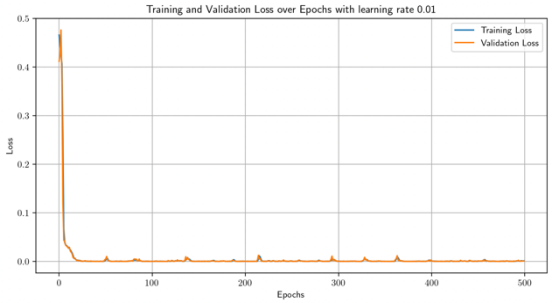


Figure 1.1.1-9 Training and Validation Loss over Epochs with Learning rate 0.01

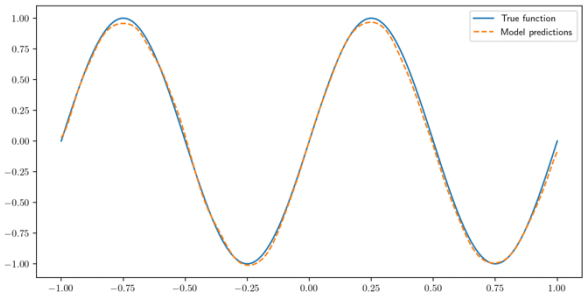


Figure 1.1.1-10 the Comparison of True Function and Model Prediction

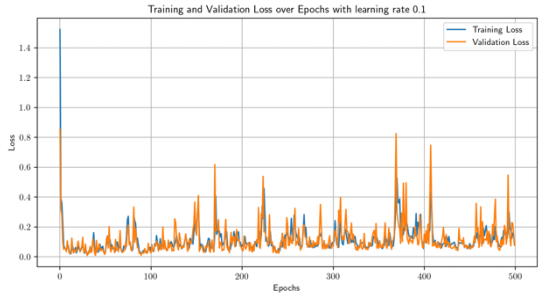


Figure 1.1.1-11 Training and Validation Loss over

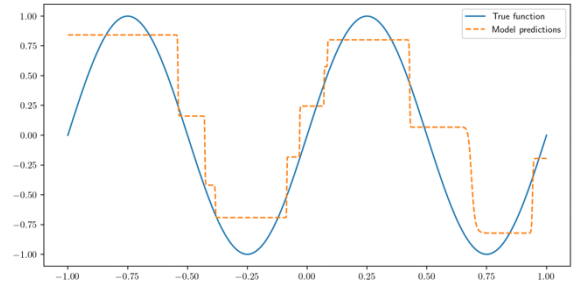


Figure 1.1.1-12 Comparison of True Function and Model Prediction

(3) The prediction of a sine curve using Tanh activation function and Learning rate of 0.01 with L2 regularization(λ=1^{-5} ,0.01)

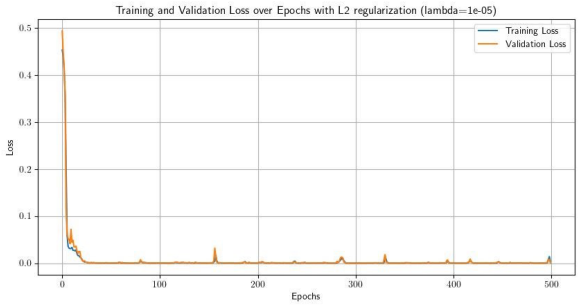


Figure 1.1.1-13 Training and Validation Loss over Epochs with L2 regularization(λ=1^{-5})

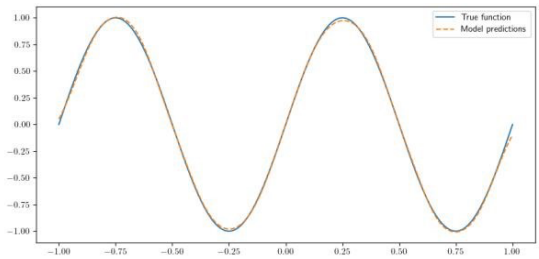


Figure 1.1.1-14 comparison of True Function and Model Prediction

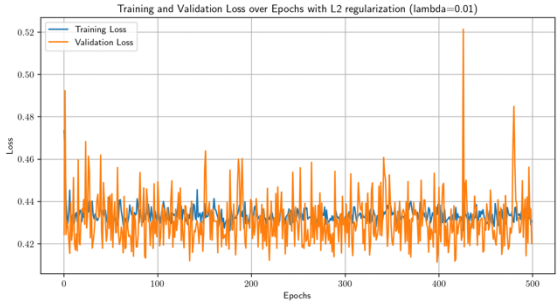


Figure 1.1.1-15 Training and Validation Loss over Epochs with L2 regularization(λ=0.01)

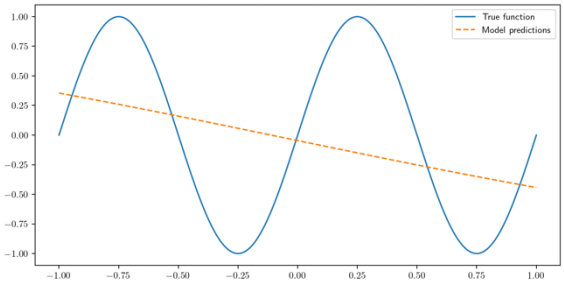


Figure 1.1.1-16 comparison of True Function and Model Prediction

1.1.2 The Influence of Activation Functions, Hyperparameters, Overfitting, and L2 Regularization by Self-Chosen Examples:

$$y=x^3-3x+1$$

(1) Three types of activation functions

① Tanh activation function

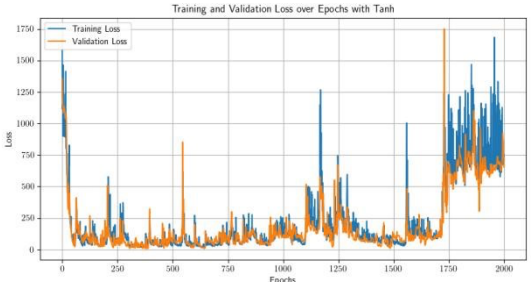


Figure 1.1.2-1 Training and Validation Loss over Epochs with L2 regularization

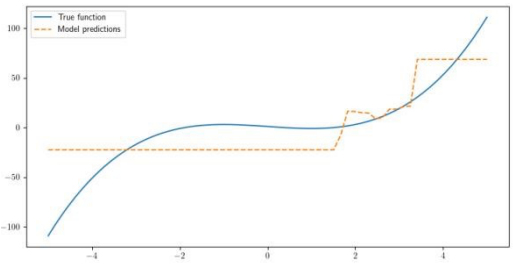


Figure 1.1.2-2 the Comparison between True Function and Model Prediction

Figure 1.1.2-1 shows:

Early loss decline:In the initial phase of training, both the training loss and validation loss decrease rapidly. This indicates that the model is learning quickly at the beginning.

Large fluctuations in the loss curve:As training progresses, the loss curve exhibits significant fluctuations, especially in the later stages of training. This may be due to the vanishing gradient problem associated with the Tanh activation function, leading to unstable training.

Training and validation losses are close:Overall, the training loss and validation loss are quite close, indicating that the model is not significantly overfitting. However, the fluctuating loss curve suggests that the training process may be unstable.

Late-stage validation loss increase:Around the 2000-epoch mark, the validation loss increases significantly, indicating the model is starting to overfit. At this point, the model performs well on the training set but shows decreased performance on the validation set.

Figure 1.1.2-2 shows:

Discontinuity in model predictions:The prediction curve exhibits obvious step-like and discontinuous characteristics. This may be due to the saturation property of the Tanh activation function. When the input values are large, the Tanh function tends to 1 or -1, causing the neuron outputs to change minimally, resulting in discontinuous changes in the model's output.

Difficulty in fitting high-order polynomials:When using the Tanh activation function to fit high-order polynomials (such as $y=^3-3+1$), this kind of discontinuous prediction result is likely to occur. This is because the gradient of the Tanh function becomes smaller for larger inputs, making it difficult for the model to capture the detailed variations of complex high-order functions.

②ReLU activation Function

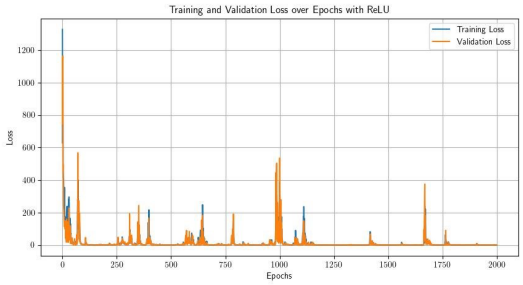


Figure 1.1.2-3 Training and Validation Loss over Epochs with ReLU

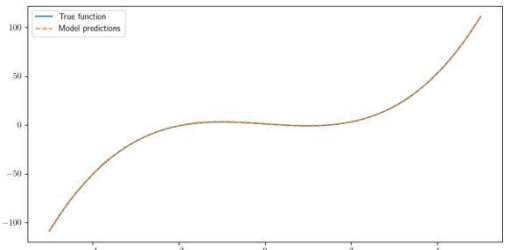


Figure 1.1.2-4 the comparison between True function and Model Prediction

Figure 1.1.2-3 shows:

Early loss decline:In the initial phase of training, both the training loss and validation loss decrease rapidly. This indicates that the model is learning effectively at the beginning.

Large fluctuations in the loss curve:During training, the loss curve shows significant fluctuations, especially in the validation loss. This may be due to the characteristics of the ReLU activation function.

Training and validation losses are close:Overall, the training loss and validation loss are close to each other for most of the training process, indicating that the model does not show significant overfitting.

High validation loss in certain intervals:In certain intervals (such as around 1000 and 1500 epochs), the validation loss is significantly higher than the training loss, suggesting a degree of overfitting, but the overall trend remains relatively stable.

Figure 1.1.2-4 shows:

The predicted results are close to the true function:The model's predictions (orange dashed line) are very close to the true function (blue solid line), indicating that the model performs well in fitting the training data.

The prediction curve is smooth:The prediction curve is quite smooth, with no obvious step-like discontinuities. This suggests that the ReLU activation function performs well in handling this univariate function.

Overall,From the training and validation loss curves, it can be seen that the ReLU function's loss decreases quickly and steadily, indicating that the vanishing gradient problem has been effectively resolved.However,In some training intervals (such as around 1000 and 1500 epochs), the loss fluctuates significantly, which might be related to the "dead ReLU" problem.

However, the overall impact is minimal.

③ Sigmoid activation Function

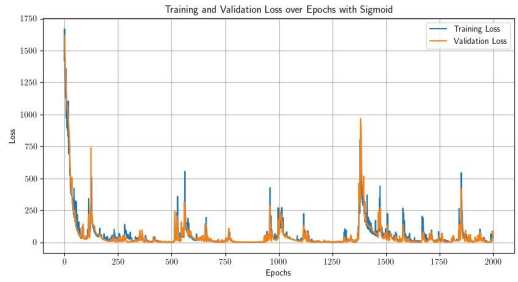


Figure 1.1.2-5 Training and Validation Loss over Epochs with Sigmoid

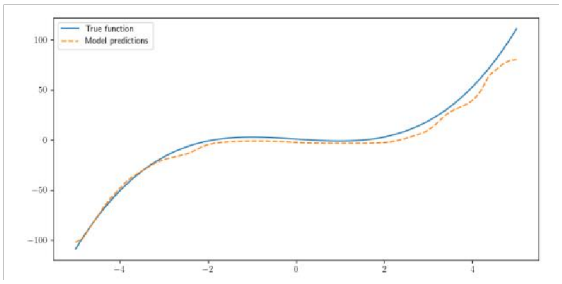


Figure 1.1.2-6 the comparison between True Function and Model Prediction

Figure 1.1.2-5 shows:

Early loss decline:In the initial phase of training, both the training loss and validation loss decrease rapidly. This indicates that the model is learning effectively at the beginning.

Large fluctuations in the loss curve:During training, the loss curve shows significant fluctuations, especially in the validation loss. This may be due to the characteristics of the Sigmoid activation function.

Training and validation losses are close:Overall, the training loss and validation loss are close to each other for most of the training process, indicating that the model does not show significant overfitting.

High validation loss in certain intervals:In certain intervals (such as around 1000 and 1500 epochs), the validation loss is significantly higher than the training loss, suggesting a degree of overfitting, but the overall trend remains relatively stable.

Figure 1.1.2-6 shows:

The predicted results are close to the true function:The model's predictions (orange dashed line) are very close to the true function (blue solid line), indicating that the model performs well in fitting the training data.

The prediction curve is smooth:The prediction curve is quite smooth, with no obvious step-like discontinuities. This suggests that the Sigmoid activation function performs well in handling this univariate function.

Vanishing gradient problem:However, the Sigmoid activation function may suffer from the vanishing gradient problem, which makes it difficult for the model to learn large variations in the data. This is reflected in the smooth but less accurate prediction in regions with steep changes in the true function.

Overfitting and instability:The large fluctuations in the validation loss, especially in later epochs, might indicate a degree of instability and potential overfitting.

Overall, while the Sigmoid activation function provides smooth predictions, it may struggle with capturing sharp transitions in the data due to the vanishing gradient issue.

(2) The Influence of hyperparameters

① different learning rate

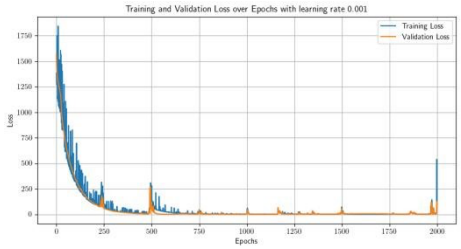


Figure 1.1.2-7 Training and Validation Loss over Epochs with learning rate 0.001

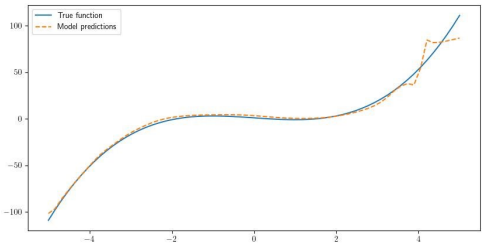


Figure 1.1.2-8 the comparison between True Function and Model Prediction

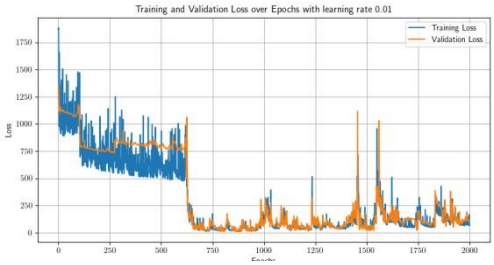


Figure 1.1.2-9 Training and Validation Loss over Epochs with learning rate 0.01

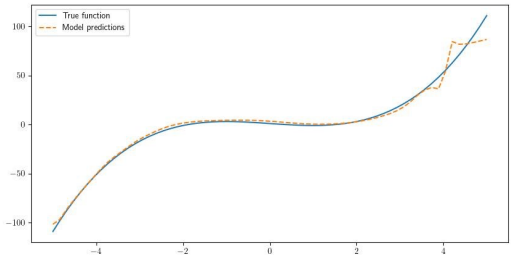


Figure 1.1.2-9 Training and Validation Loss over Epochs with learning rate 0.01

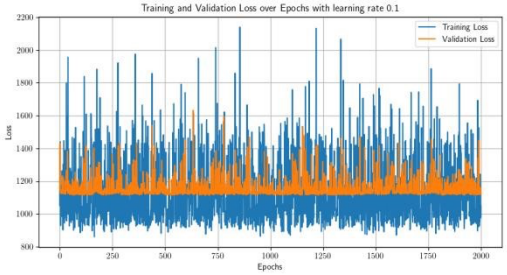


Figure 1.1.2-11 Training and Validation Loss over Epochs with learning rate 0.1

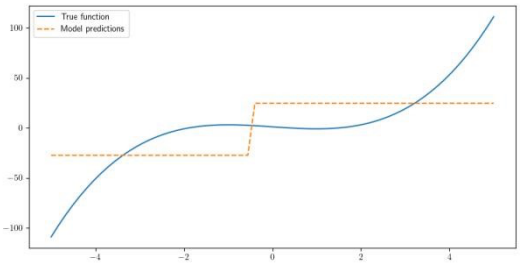


Figure 1.1.2-12 Training and Validation Loss over Epochs with learning rate 0.1

Low Learning Rate(0.001): Slow but stable learning, low risk of overshooting, potentially long training time.

Medium Learning Rate(0.01): Balanced learning speed and stability, some fluctuations, efficient learning.

High Learning Rate(0.1): Rapid learning with significant fluctuations, risk of overshooting, potential overfitting and instability.

② different λ

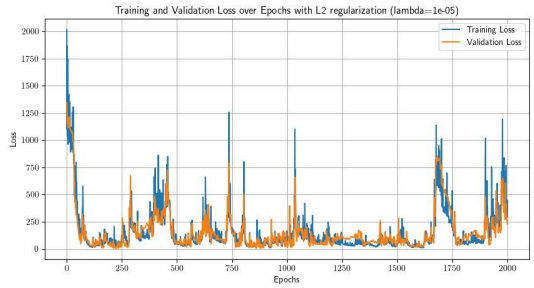


Figure 1.1.2-13 Training and Validation Loss over Epochs with L2 regularization λ=1^{-5}

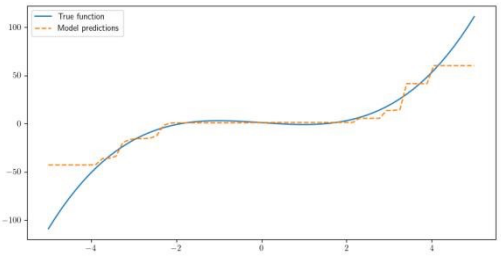


Figure 1.1.2-14 Training and Validation Loss over Epochs with L2 regularization λ=1^{-5}

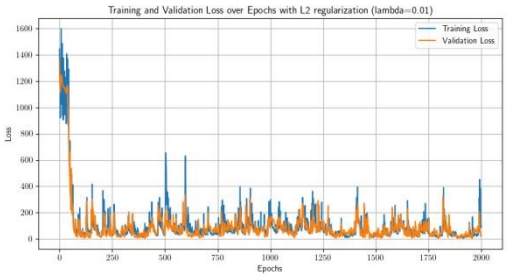


Figure 1.1.2-15 Training and Validation Loss over Epochs with L2 regularization λ=0.01

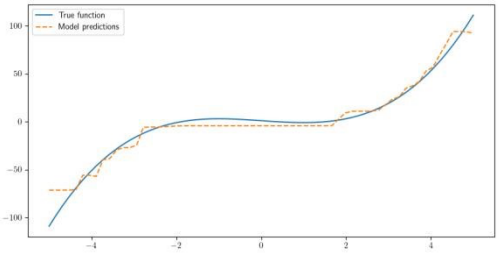


Figure 1.1.2-16 Training and Validation Loss over Epochs with L2 regularization λ=0.01

Low L2 Regularization (lambda = 1e-5): Minimal effect, smooth and similar training and validation loss curves, minimal overfitting prevention.

Medium L2 Regularization (lambda = 1e-2): Noticeable effect, balanced learning, improved stability, smoother predictions.

(3) Overfitting Analysis

Reasons for Overfitting:

Complex Model,Insufficient Training Data,Excessive Training,Lack of Regularization,High Learning Rate,Noise in Data Analysis:

Figure 1.1.2-3 shows a degree of overfitting.Figures 1.1.2-9 and 1.1.2-11 also demonstrate overfitting, which is attributed to the relatively higher learning rates.Additionally, Figure 1.1.2-13 illustrates overfitting near the end due to the lower λ in regularization.However, the phenomenon of overfitting is not very clear in these figures.

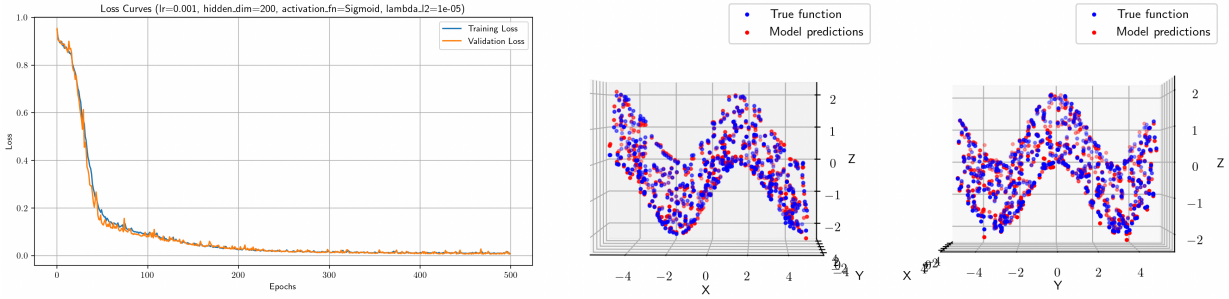
1.2 Multivariate hyperparameter optimization

Choose a suitable, bi-variate function to create training data. Adapt the plot code to visualize the data in a suitable and understandable way for the documentation.

Change the hyperparameters to get sufficiently accurate results. Document this hyperparameter search, give examples for non-suitable hyperparameter combinations which occurred during the search and explain the effects why the new values for the hyperparameters are necessary in this case.

- Setting up bivariate functions z = sin(x) + cos(y) to create training data.
- The code generates 512 random x and y values ranging from -5 to 5.
- The model is trained using stochastic hyperparameter searches (e.g., learning rate, activation function, and L2 regularization coefficients).

1. Iteration



The hyperparameters are set to lr=0.001, hidden_dim=200, activation_fn=Sigmoid, lambda_l2=1e-05

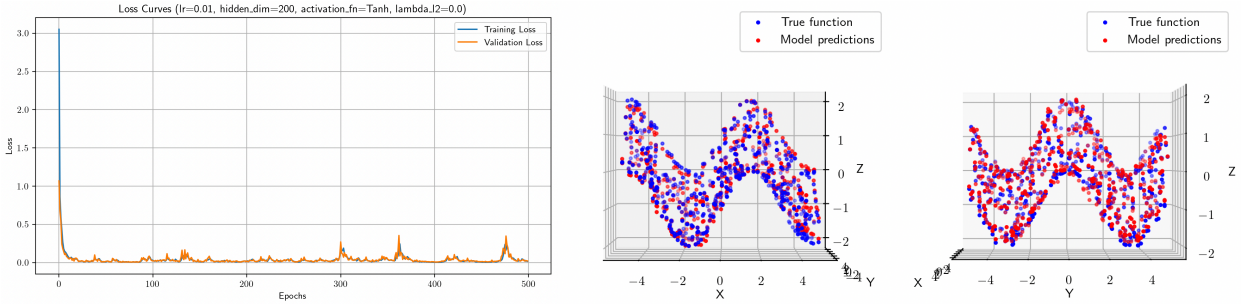
The figure loss curves shows that both the training loss and the validation loss are gradually decreasing as the training progresses, indicating that the model is constantly fitting the data.

Eventually the two curves flatten out, indicating that the model training has reached a better convergence state.

As 3D Scatterplot can be seen from the figure, the model predicted values (red dots) highly overlap with the true values (blue dots), indicating that the model is able to fit the function z = sin(x) + cos(y) better.

These plots show that: The loss profile plots show the decreasing trend of the model training and validation losses, indicating that the model training is effective. The 3D scatter plots show the accuracy of the model predictions, with a high degree of overlap between the predicted and true values, indicating that the model has a good fit.

2. Iteration



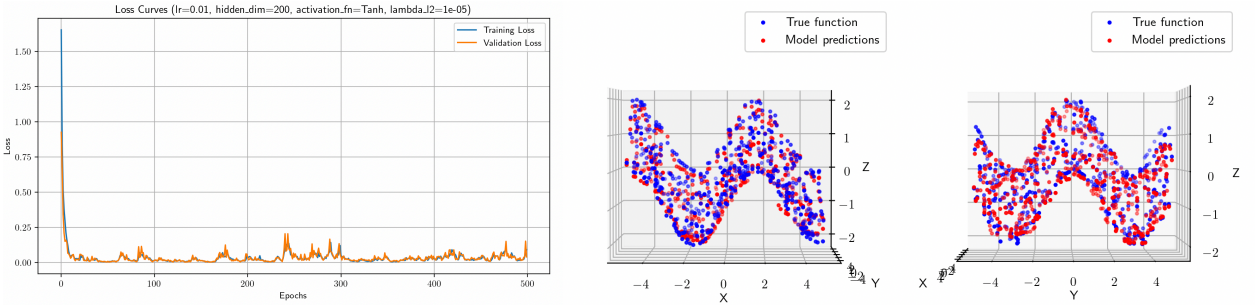
The hyperparameters are set to lr=0.01, hidden_dim=200, activation_fn=Tanh, lambda_l2=0.0

The figure loss curves shows that the training loss and validation loss decrease rapidly in the early stages, but fluctuate somewhat in the middle and late stages. This may be due to the fact that the model encounters different data features in different training cycles, resulting in fluctuating losses.

It can be seen that the model predicted values and the true values basically overlap, indicating that the model fits the function z = sin(x) + cos(y) better, but the distribution of the red and blue points is slightly dispersed compared to the first set of pictures, which may indicate that the model fits slightly less well under this set of hyperparameters.

These plots show that: The loss plot shows that the training and validation loss of the model decreases rapidly in the early stages but fluctuates in the later stages, which may be related to the higher learning rate. The 3D scatter plot shows that the model is able to fit the objective function better, but the predictions are slightly more dispersed as compared to the first set of hyperparameters, which may imply that the model does not fit as well as the first set of hyperparameters with this hyperparameter.

3. Iteration



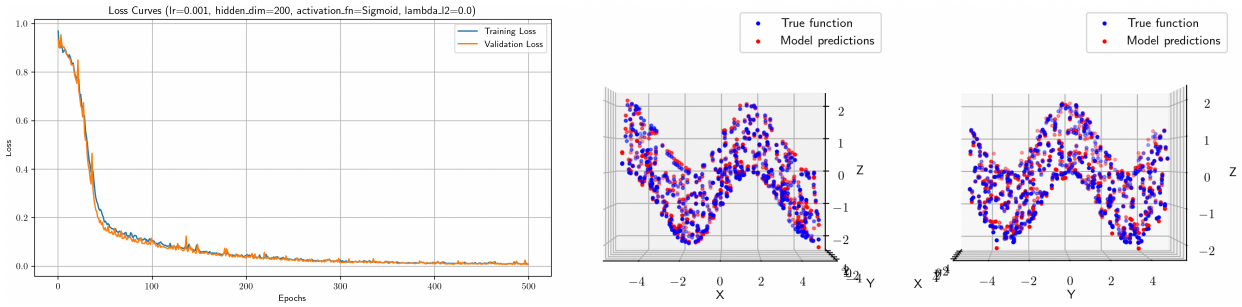
The hyperparameters are set to lr=0.01, hidden_dim=200, activation.fn=Tanh, lambda_2=1e-05

The figure loss curves shows that the training loss and validation loss decrease rapidly in the early stages but fluctuate somewhat in the later stages, which may be due to the fact that the model encounters different data features in different training cycles, resulting in fluctuations in the loss.

It can be seen that the model predicted values and the true values basically overlap, but compared to the previous groups, the distribution of the red and blue points is slightly scattered, indicating that the model's fitting effect under this set of hyperparameters is slightly worse than the first set of hyperparameters

These plots show that: The loss plot shows that the training and validation loss of the model decreases rapidly in the early stages but fluctuates in the later stages, which may be related to the learning rate and the choice of L2 regularization. The 3D scatter plot shows that the model is able to fit the objective function better, but the dispersion of the predicted values is slightly higher compared to the optimal hyperparameter settings, indicating that the model's fit decreases slightly with this set of hyperparameters.

4. Iteration



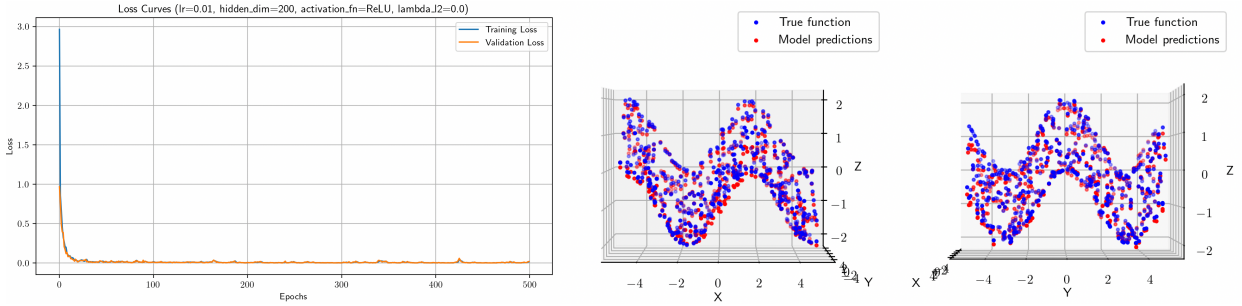
The hyperparameters are set to lr=0.001, hidden_dim=200, activation_fn=Sigmoid, lambda_l2=0.0

The figure loss curves shows that the training loss and validation loss decrease rapidly in the early stages and gradually level off, indicating that the model is fitting the data stably.

It can be seen that the model predicted values and the true values basically coincide, indicating that the model can fit the bivariate function of this setup better under this hyperparameter setting

These plots show that: The loss plot shows that the training and validation loss of the model decreases rapidly in the initial stage and tends to stabilize gradually, which indicates that the model is well trained under the current hyper-parameter setting. The 3D scatter plot shows that the model is able to fit the objective function well, and the predicted values and the true values are highly overlapped, which indicates that the model has a high degree of accuracy under this hyper-parameter setting.

5. Iteration



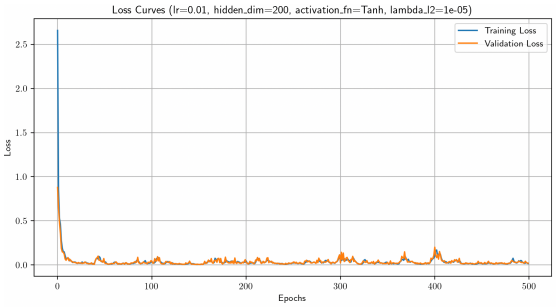
The hyperparameters are set to lr=0.01, hidden_dim=200, activation_fn=ReLU, lambda_2=0.0

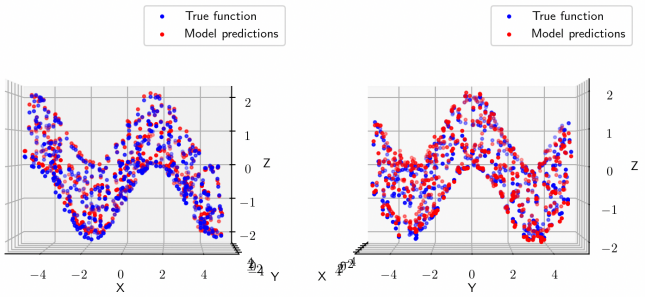
The figure loss curves shows that the training loss and validation loss decrease rapidly at the beginning and gradually level off, indicating that the model is fitting the data stably. Eventually the two curves level off and remain at low loss values, indicating that the model is well trained.

It can be seen that the model predicted values and the true values basically coincide, indicating that the model can fit the bivariate function of this setup better under this hyperparameter setting

These plots show that: The loss plot shows that the training and validation loss of the model decreases rapidly in the initial stage and tends to stabilize gradually, which indicates that the model is well trained under the current hyper-parameter setting. The 3D scatter plot shows that the model is able to fit the objective function well, and the predicted values and the true values are highly overlapped, which indicates that the model has a high degree of accuracy under this hyper-parameter setting.

6. Iteration





The hyperparameters are set to lr=0.01, hidden_dim=200, activation_fn=Tanh, lambda_l2=1e-05

The figure loss curves shows that the training loss and the validation loss drop rapidly in the early stages and remain low throughout the training process, but with some fluctuations halfway through..

It can be seen that the model predicted values and the true values basically overlap, but there is a slight dispersion, indicating that the model is able to fit the bivariate function better with this hyperparameter setting, but not as well as the previous sets of fits.

These plots show that: The training and validation losses of the model are shown to decrease rapidly in the early stages and level off in the later stages, but with some fluctuations in the middle, indicating that the model is trained reasonably well under the current hyperparameter setting, but there is still room for optimization. Shows that the model is able to fit the objective function relatively well, but the predicted values are relatively scattered, indicating that the fit under this hyperparameter setting is slightly less effective than the best combination.

Conclusion

After six iterations using the random hyperparameter approach it can be learned that the model with hyperparameter lr=0.001, hidden_dim=200, activation_fn=Sigmoid, lambda_l2=1e-05 is the best for the first iteration.

homework 2.1.3

May 21, 2024

In this part, we explore the phenomenon of overfitting in a model trained on highly noisy data and the methods to mitigate it.

1 Data Generation

We generated a training dataset containing 512 samples within the range of [-5, 5], adding a significant amount of noise (noise level set to 3.0). The formula is as follows:

$$z = \sin(x) + \cos(y) + \mathcal{N}(0, 3.0)$$

where x and y are uniformly distributed random variables in the interval [-5, 5], and $\mathcal{N}(0, 3.0)$ represents Gaussian noise with a mean of 0 and a standard deviation of 3.0.

2 Model Structure

We used a neural network model with multiple fully connected layers and introduced Dropout layers to increase the model's complexity. The model's structure is as follows:

- Input Layer: 2 nodes
- Hidden Layer 1: 200 nodes, ReLU activation function
- Hidden Layer 2: 400 nodes, ReLU activation function
- Hidden Layer 3: 400 nodes, ReLU activation function
- Hidden Layer 4: 200 nodes, ReLU activation function
- Output Layer: 1 node

Training Process

During training, we first trained a model without regularization to observe overfitting. The hyperparameters are as follows:

- Learning Rate: 0.01
- Epochs: 3000
- No L2 Regularization (lambda = 0.0)
- No Dropout (dropout_prob = 0.0)

Results Analysis

The results of training without regularization are shown in Figure 1:

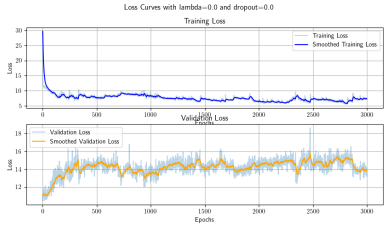


Figure 1: Overfitting

From the figure, it can be seen that the training loss rapidly decreases and stabilizes in the initial stages. However, the validation loss exhibits significant fluctuations and remains at a higher level throughout the training process. This indicates that the model performs well on the training set but poorly on the validation set, signifying overfitting.

Overfitting

Overfitting occurs when a model performs well on the training data but poorly on new data (such as the validation set). The primary cause of overfitting is the model's excessive complexity, which leads to overlearning the noise and details in the training data, resulting in poor generalization to new data.

Introducing Regularization

To reduce overfitting, we introduced L2 regularization and Dropout into the model. The adjusted hyperparameters are as follows:

- L2 Regularization Coefficient (lambda): 0.05

– Dropout Probability (dropout_prob): 0.5

The training results are shown in Figure 2:

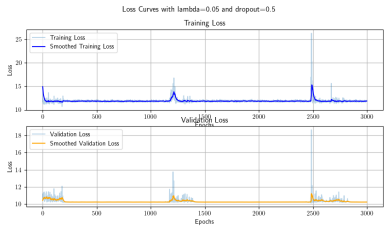


Figure 2: apply L2 and Dropout

From the figure, it can be seen that both the training loss and validation loss remain at lower levels with reduced fluctuations. This indicates that the model's generalization capability has significantly improved after introducing regularization, effectively mitigating overfitting.

Conclusion

The experiment demonstrates that without regularization, the model exhibits significant overfitting when trained on highly noisy data. By introducing L2 regularization and Dropout, overfitting can be effectively reduced, and the model's generalization ability can be enhanced. This highlights the importance of regularization techniques when training neural networks, especially when dealing with noisy data.

Applying autograd to displacement fields

Displacement gradient:

The displacement gradient **grad_u** describes the local deformation of the object during the deformation process. The displacement gradient is a second-order tensor that represents the rate of change of the displacement vector of each point in the object relative to the initial position.

Mathematical definition:

$$\nabla \mathbf{u} = \begin{pmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} \\ \frac{\partial u_y}{\partial x} & \frac{\partial u_y}{\partial y} \end{pmatrix} \tag{1}$$

In our example, the dimension of $\mathbf{u}$ is $(13041, 2)$, and the dimension of the displacement gradient **grad_u** is $(13041, 2, 2)$.

$$\begin{aligned} \text{grad\_u}[:, 0, 0] \quad &\text{is} \quad \frac{\partial u_x}{\partial x} \\ \text{grad\_u}[:, 0, 1] \quad &\text{is} \quad \frac{\partial u_x}{\partial y} \\ \text{grad\_u}[:, 1, 0] \quad &\text{is} \quad \frac{\partial u_y}{\partial x} \\ \text{grad\_u}[:, 1, 1] \quad &\text{is} \quad \frac{\partial u_y}{\partial y} \end{aligned}$$

In our code we show the displacement gradients in the x and y directions respectively.

Deformation Gradient:

In continuum mechanics, the deformation gradient tensor, denoted as $\mathbf{F}$, is a key concept that describes the deformation of a material from its initial configuration to its current configuration. The deformation gradient tensor at each point characterizes how an infinitesimal material element deforms.

Mathematical Definition:

The deformation gradient tensor $\mathbf{F}$ is defined as:

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}$$

where:

- $\mathbf{X}$ is the coordinate in the reference (initial) configuration.
- $\mathbf{x}$ is the coordinate in the current (deformed) configuration.

Relation Between Deformation Gradient and Displacement Gradient:

The deformation gradient tensor $\mathbf{F}$ can be expressed in terms of the identity matrix $\mathbf{I}$ and the displacement gradient tensor $\nabla \mathbf{u}$. The displacement gradient tensor $\nabla \mathbf{u}$ is the spatial derivative of the displacement vector field $\mathbf{u}$:

$$\nabla \mathbf{u} = \frac{\partial \mathbf{u}}{\partial \mathbf{X}}$$

Given that the current position $\mathbf{x}$ is related to the reference position $\mathbf{X}$ by the displacement vector $\mathbf{u}$:

$$\mathbf{x} = \mathbf{X} + \mathbf{u}$$

the deformation gradient can be written as:

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \frac{\partial (\mathbf{X} + \mathbf{u})}{\partial \mathbf{X}} = \frac{\partial \mathbf{X}}{\partial \mathbf{X}} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}} = \mathbf{I} + \nabla \mathbf{u}$$

where $\mathbf{I}$ is the identity matrix.

The Right and Left Cauchy-Green Tensors:

In continuum mechanics, the right and left Cauchy-Green deformation tensors are used to describe the deformation of a material body. These tensors provide a measure of the strain experienced by the material.

Right Cauchy-Green Tensor:

The right Cauchy-Green deformation tensor, denoted as $\mathbf{C}$, is defined as:

$$\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F}$$

where:

- $\mathbf{F}$ is the deformation gradient tensor.
- $\mathbf{F}^{\mathrm{T}}$ is the transpose of the deformation gradient tensor.

The right Cauchy-Green tensor $\mathbf{C}$ describes the deformation in the reference configuration. It provides a measure of the stretch experienced by the material without considering rotations.

Left Cauchy-Green Tensor:

The left Cauchy-Green deformation tensor, denoted as $\mathbf{B}$, is defined as:

$$\mathbf{B} = \mathbf{F}\mathbf{F}^{\mathrm{T}}$$

where:

- $\mathbf{F}$ is the deformation gradient tensor.
- $\mathbf{F}^{\mathrm{T}}$ is the transpose of the deformation gradient tensor.

The left Cauchy-Green tensor $\mathbf{B}$ describes the deformation in the current configuration. It provides a measure of the stretch experienced by the material, considering rotations.

The Green-Lagrange Strain Tensor:

In continuum mechanics, the Green-Lagrange strain tensor is a measure of strain that describes the deformation of a material body. It is particularly useful in finite deformation analysis as it accounts for large deformations.

Mathematical Definition:

The Green-Lagrange strain tensor, denoted as $\mathbf{E}$, is defined as:

$$\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I})$$

where:

- $\mathbf{C}$ is the right Cauchy-Green deformation tensor, defined as $\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F}$.
- $\mathbf{I}$ is the identity matrix.
- $\mathbf{F}$ is the deformation gradient tensor.

To compute the Green-Lagrange strain tensor, we first need the deformation gradient tensor $\mathbf{F}$ and the right Cauchy-Green deformation tensor $\mathbf{C}$. Here is an example in Python using NumPy and PyTorch:

Resulting Plots:

Below are the resulting plots of this simulation:
The displacement gradient, deformation gradient, right Cauchy-Green tensor and Green-Lagrange strain tensor have no equivalent values in the x direction, and all variables become larger at the far end in the y direction.

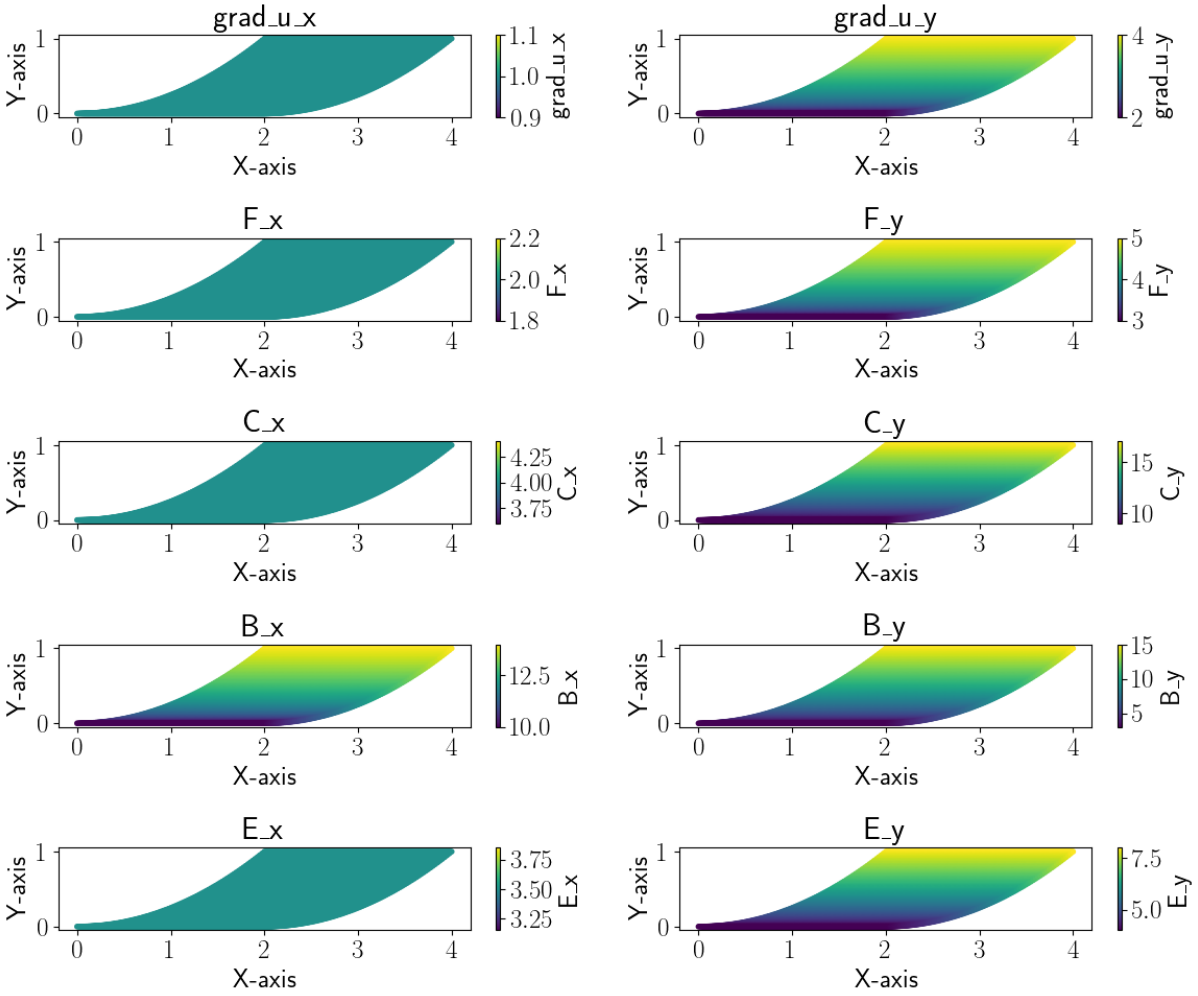


Figure 1: Displacement gradient,Deformation gradient,The Right and Left Cauchy-Green Tensors and The Green-Lagrange Strain Tensor in x and y directions