AMAL Homework-1 :

Essential elements of deep learning and Pytorch

Guillaume Potel

ZHAO Yunfei

October 19, 2021

In this homework we aim at demonstrating some basic elements in deep learning, such as forward propagation, backward propagation, chain rule, gradient descent, stochastic gradient descent, data loader with some implementations from scratch as well as some manipulation with Pytorch.

Contents

List of Figures

Chapter 1

TP1

1.1 Function in a graph

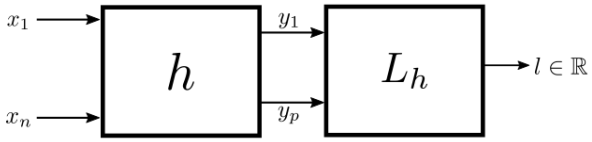

Figure 1.1: Illustration of function h and L in a graph

Question 1

Calculate the partial derivative of an input of our composite function

$$\frac{\partial L \circ h}{\partial x_i}(x)$$

in terms of

$$\frac{\partial h_j}{\partial x_i}(x)$$

and

$$(\nabla L)_j = \frac{\partial L}{\partial y_j}(h(x))$$

$$\begin{aligned}
\frac{\partial L \circ h}{\partial x_i}(x) &= \sum_{j=1}^{p} \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial x_i} \\
&= \sum_{j=1}^{p} \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial x_i}(x) \\
&= \sum_{j=1}^{p} (\nabla L)_j \frac{\partial h_j}{\partial x_i}(x) \\
&= \nabla L \left(\frac{\partial h_1}{\partial x_i}, ..., \frac{\partial h_p}{\partial x_i} \right)^\top
\end{aligned} \tag{1.1}$$

1.2 Application: Linear Regression

1.2.1 gradient calculation (scalar)

Our linear regression function has two composition:

- A linear function for making predictions: $\hat{y} = f(\mathbf{x}, \mathbf{w}, b) = \mathbf{x} \cdot \mathbf{w} + b$
- A Mean squared error (MSE) loss function : $mse(\hat{y}, y) = (\hat{y} - y)^2$

The loss function:

$$C(\mathbf{x}, \mathbf{w}, b, y) = mse(f(\mathbf{x}, \mathbf{w}, b), y) \tag{1.2}$$

Question 2

We are going to calculate the gradients of C regarding $\mathbf{w}$ and b to get the optimal value by gradient descent by the results from equation 1.1.

- The derivatives of loss function MSE.

$$\frac{\partial L \circ mse}{\partial y}(\hat{y}, y)$$

and

$$\frac{\partial L \circ mse}{\partial \hat{y}}(\hat{y}, y)$$

L here is identity function.

- The derivatives of linear function f as regards to theirs inputs $\mathbf{x}$ (We replace it by $\mathbf{w}$ and b here, because we want to do gradient descent on these two terms.)

$$\frac{\partial L \circ f}{\partial x_j}(\mathbf{x}, \mathbf{w}, b)$$

L here is the MSE loss.

In real case, L can be a composition of very complex functions.

- $$\begin{aligned} \frac{\partial L \circ mse}{\partial y}(\hat{y}, y) &= \nabla L \cdot -2(\hat{y} - y) \\ &= -2(\hat{y} - y) \\ \frac{\partial L \circ mse}{\partial \hat{y}}(\hat{y}, y) &= 2(\hat{y} - y) \end{aligned} \tag{1.3}$$

- $$\begin{aligned} \frac{\partial mse \circ f}{\partial x_j}(\mathbf{x}, \mathbf{w}, b) &= \nabla mse \times \left(\frac{\partial y}{\partial x_j}, \frac{\partial \hat{y}}{\partial x_j} \right)^\top \\ &= \left(\frac{\partial mse}{\partial y}, \frac{\partial mse}{\partial \hat{y}} \right) \times \left(\frac{\partial y}{\partial x_j}, 0 \right)^\top \\ &= -2(\hat{y} - y) \cdot w_j \\ \frac{\partial mse \circ f}{\partial w_j}(\mathbf{x}, \mathbf{w}, b) &= -2(\hat{y} - y) \cdot x_j \\ \frac{\partial mse \circ f}{\partial b}(\mathbf{x}, \mathbf{w}, b) &= -2(\hat{y} - y) \end{aligned} \tag{1.4}$$

1.2.2 gradient calculation (matrix)

Now we are analysing a more general case where we have a batch of q samples and the output is of dimension p instead of 1. We have then:

$$\hat{\mathbf{Y}} = f(\mathbf{X}, \mathbf{W}, \mathbf{b}) = \mathbf{X}\mathbf{W} + \mathbf{b} \tag{1.5}$$

Where $\mathbf{X} \in \mathbb{R}^{q \times n}$, $\mathbf{W} \in \mathbb{R}^{n \times p}$, $\hat{\mathbf{Y}} \in \mathbb{R}^{q \times p}$, $\mathbf{b} \in \mathbb{R}^{1 \times p}$

The MSE is defined as below and we divide it by p so as to remove the influence of samples number on magnitude of gradient.

$$MSE(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{q}\|\hat{\mathbf{Y}} - \mathbf{Y}\|^2 \tag{1.6}$$

The definition of cost function is:

$$C(\mathbf{X}, \mathbf{W}, \mathbf{b}, \mathbf{Y}) = mse(f(\mathbf{X}, \mathbf{W}, \mathbf{b}), Y) \tag{1.7}$$

Question 3

We are going to apply the result we achieve from section 1.2.1 to this part.

- The derivatives of loss function MSE.

$$\frac{\partial L \circ mse}{\partial \mathbf{Y}_{ij}}(\hat{\mathbf{Y}}, \mathbf{Y})$$

and

$$\frac{\partial L \circ mse}{\partial \hat{\mathbf{Y}}_{ij}}(\hat{\mathbf{Y}}, \mathbf{Y})$$

L here is identity function.

- The derivatives of linear function f as regards to theirs inputs $\mathbf{x}$ (We replace it by $\mathbf{w}$ and b here, because we want to do gradient descent on these two terms.)

$$\frac{\partial L \circ f}{\partial \mathbf{X}_{ij}}(\mathbf{X}, \mathbf{W}, \mathbf{b})$$

$$\frac{\partial L \circ f}{\partial \mathbf{W}_{ij}}(\mathbf{X}, \mathbf{W}, \mathbf{b})$$

$$\frac{\partial L \circ f}{\partial \mathbf{b}_{i}}(\mathbf{X}, \mathbf{W}, \mathbf{b})$$

L here is the MSE loss.

- $$\begin{aligned} \frac{\partial L \circ mse}{\partial \mathbf{Y}_{ij}}(\hat{\mathbf{Y}}, \mathbf{Y}) &= -\frac{2}{q}\nabla L(\hat{Y}_{ij} - Y_{ij}) \\ \frac{\partial L \circ mse}{\partial \hat{\mathbf{Y}}_{ij}}(\hat{\mathbf{Y}}, \mathbf{Y}) &= \frac{2}{q}\nabla L(\hat{Y}_{ij} - Y_{ij}) \end{aligned} \tag{1.8}$$

- $$\begin{aligned} \frac{\partial L \circ f}{\partial \mathbf{X}_{ij}}(\mathbf{X}, \mathbf{W}, \mathbf{b}) &= \sum_{i=1}^{q}\sum_{j=1}^{p}\frac{\partial L}{\partial \mathbf{Y}_{ij}}(f(\mathbf{X}, \mathbf{W}, \mathbf{b}))\frac{\partial f_{ij}}{\partial \mathbf{X}_{ij}}(\mathbf{X}, \mathbf{W}, \mathbf{b}) \\ &= \sum_{i=1}^{q}\sum_{j=1}^{p}(\nabla L)_j W_{j,i} \\ \frac{\partial L \circ f}{\partial \mathbf{W}_{ij}}(\mathbf{X}, \mathbf{W}, \mathbf{b}) &= \sum_{i=1}^{q}\sum_{j=1}^{p}\mathbf{X}_{ji}(\nabla L)_i \\ \frac{\partial L \circ f}{\partial \mathbf{b}_{i}}(\mathbf{X}, \mathbf{W}, \mathbf{b}) &= \sum_{i=1}^{q}\sum_{j=1}^{p}\frac{\partial L}{\partial \mathbf{Y}_{ij}}(f(\mathbf{X}, \mathbf{W}, \mathbf{b})) \cdot 1 \end{aligned} \tag{1.9}$$

To make use of the parallel architectures (GPU), we are going to express as much as possible the calculations in a matrix manner.

Question 4

We present the result we achieve in the previous question in a matrix way.

- $$\begin{aligned} \frac{\partial L \circ mse}{\partial \mathbf{Y}}(\hat{\mathbf{Y}}, \mathbf{Y}) &= -\frac{2}{q}\nabla L(\hat{Y} - Y) \\ \frac{\partial L \circ mse}{\partial \hat{\mathbf{Y}}}(\hat{\mathbf{Y}}, \mathbf{Y}) &= \frac{2}{q}\nabla L(\hat{Y} - Y) \end{aligned} \tag{1.10}$$

  Where ∇L is a scalar

- $$\begin{aligned} \frac{\partial L \circ f}{\partial \mathbf{X}}(\mathbf{X}, \mathbf{W}, \mathbf{b}) &= \nabla L \cdot \mathbf{W}^\top \\ \frac{\partial L \circ f}{\partial \mathbf{W}}(\mathbf{X}, \mathbf{W}, \mathbf{b}) &= \mathbf{X}^\top \cdot \nabla L \\ \frac{\partial L \circ f}{\partial \mathbf{b}}(\mathbf{X}, \mathbf{W}, \mathbf{b}) &= \nabla L \end{aligned} \tag{1.11}$$

Question 5

Calculate $\frac{\partial C}{\partial \mathbf{W}}$ and $\frac{\partial C}{\partial \mathbf{b}}$

$$\begin{aligned} \frac{\partial C}{\partial \mathbf{W}} &= \frac{\partial C}{\partial \hat{\mathbf{Y}}}\frac{\partial \hat{\mathbf{Y}}}{\partial \mathbf{W}} \\ &= \mathbf{X}^\top(\frac{2}{q}\nabla L(\hat{Y} - Y)) \\ \frac{\partial C}{\partial \mathbf{b}} &= \frac{\partial C}{\partial \hat{\mathbf{Y}}}\frac{\partial \hat{\mathbf{Y}}}{\partial \mathbf{b}} \\ &= \frac{2}{q}\nabla L(\hat{Y} - Y) \end{aligned} \tag{1.12}$$

For the rest of this part, we implement **MSE** and **Linear** class as what Pytorch do in **TP01/tp1.py** and it is also verified by a numerical method in **TP01/tp1_gradcheck.py** by **torch.autograd.gradcheck**. Finally, we make a simple gradient descent for linear regression in **TP01/tp1_descente.py**

Chapter 2

TP2

In this part, the work is more experimental. We explore some functions and in Pytorch and make a comparison between gradient descent and stochastic gradient descent on **Boston house dataset**. We explored **Autograd** and **Optimiser** and **Container**.

We find that gradient descent with mini batch converge quicker than gradient descent on whole data. It is due to that mini batch make more gradient descent operation during on epoch.

All implementations for this part can be found in **TP02**.

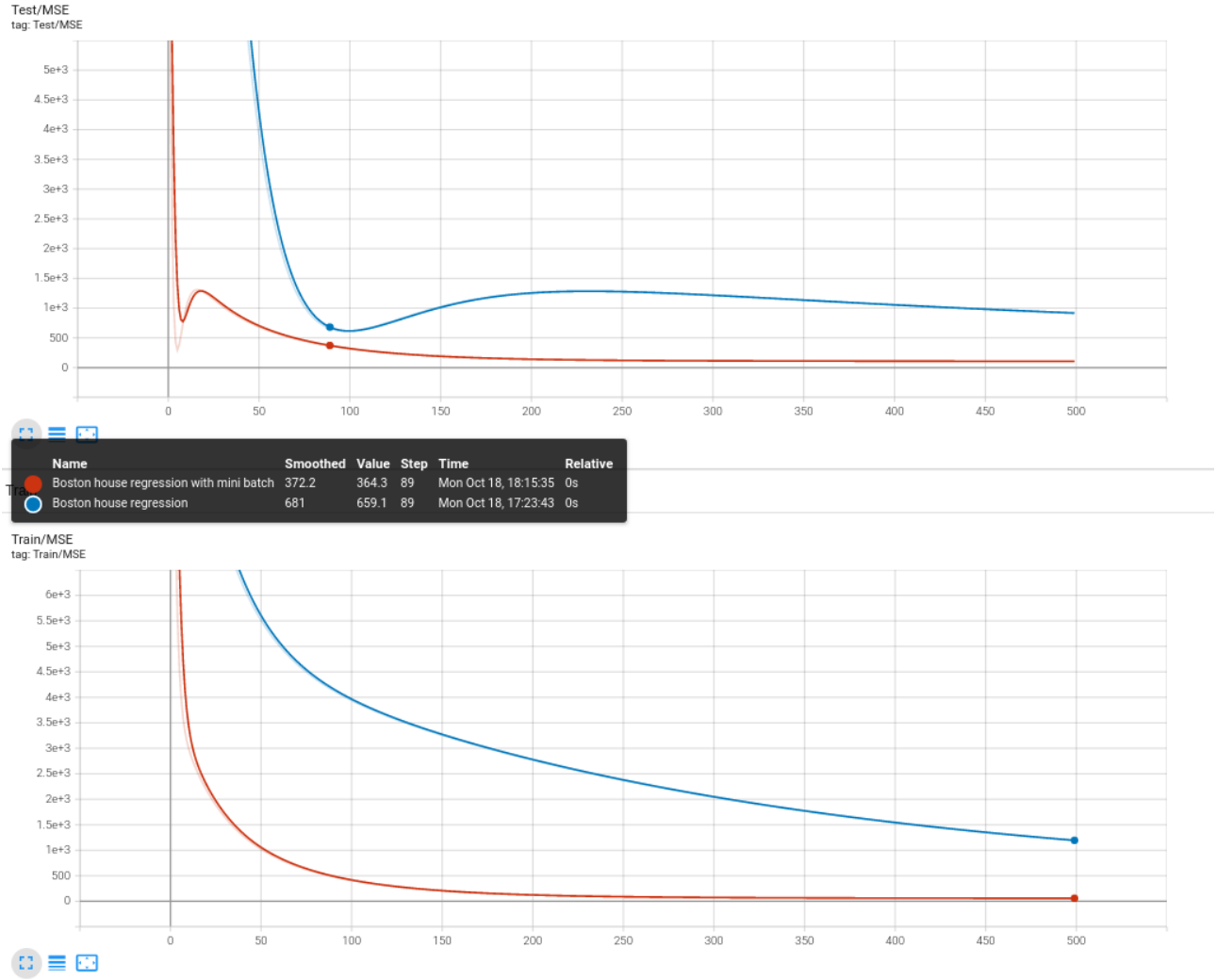

Figure 2.1: gradient descent on Boston house dataset. red: mini batch with size 30, blue all data

Chapter 3

TP3

For this part, we implement a simple auto encoder to present images in a low dimension. We use all techniques we learned for now such as **Dataloader**, **GPU**, **Checkpointing**.

3.1 Comparison

3.2 Why we use W.T from encoder as weight of decoder

May be we suppose that like in PCA, a matrix of positive definite of covariance can be used to extract important dimensions. And in this way, we have a prior that these two matrix are the inverse of each other. During the training, this converge surely quicker. We can conclude that

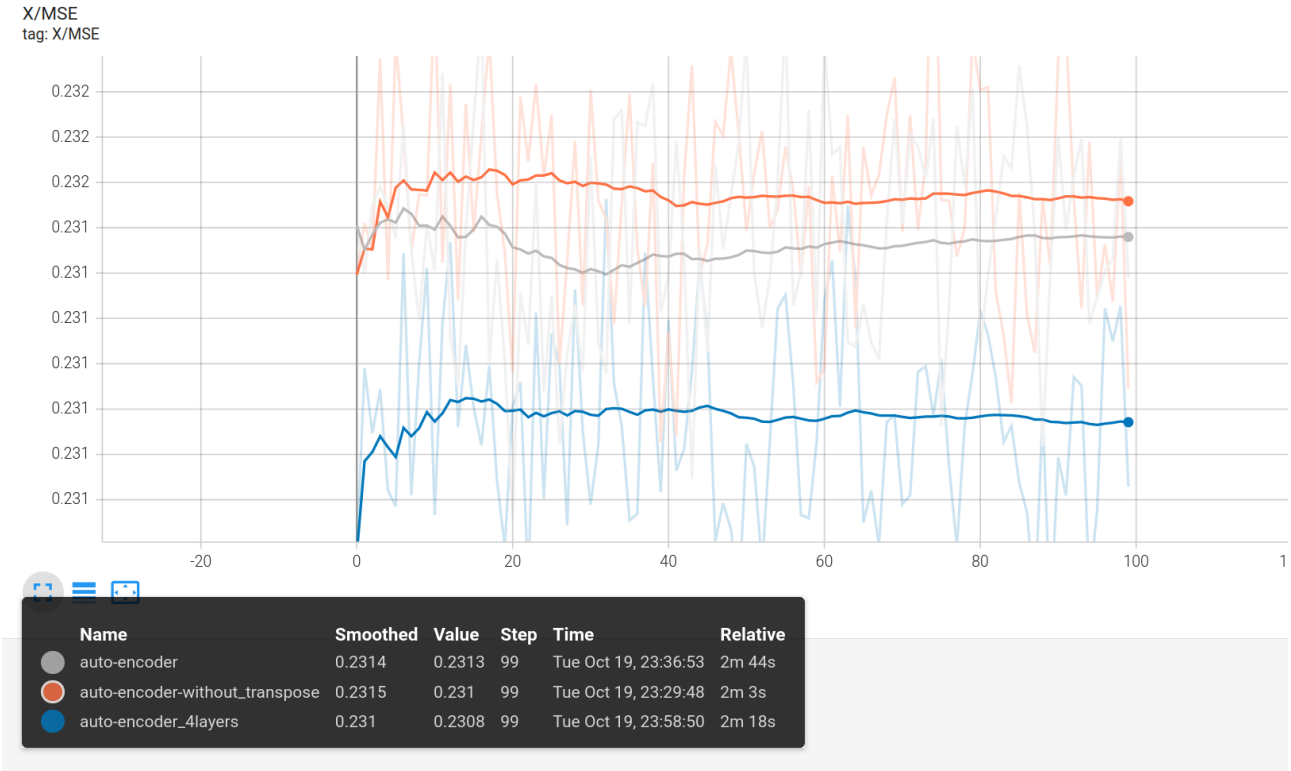

Figure 3.1: Comparison of MSE during training for different models

if we take the transpose of layer weight from encoder to decoder, our model converge quicker

with a better result. And the four layers model converge better.

3.3 Speed of GPU

During training, we compared also the speed for CPU version and GPU version, but we found an anti intuition phenomenon is that the CPU version is not much quicker than CPU version, we think that our net is too small and the parallel power is not well exploited because the memory of GPU is well used while the computation resource is only used 5% when we launched **nvidia-smi**

3.4 Embedding visualisation

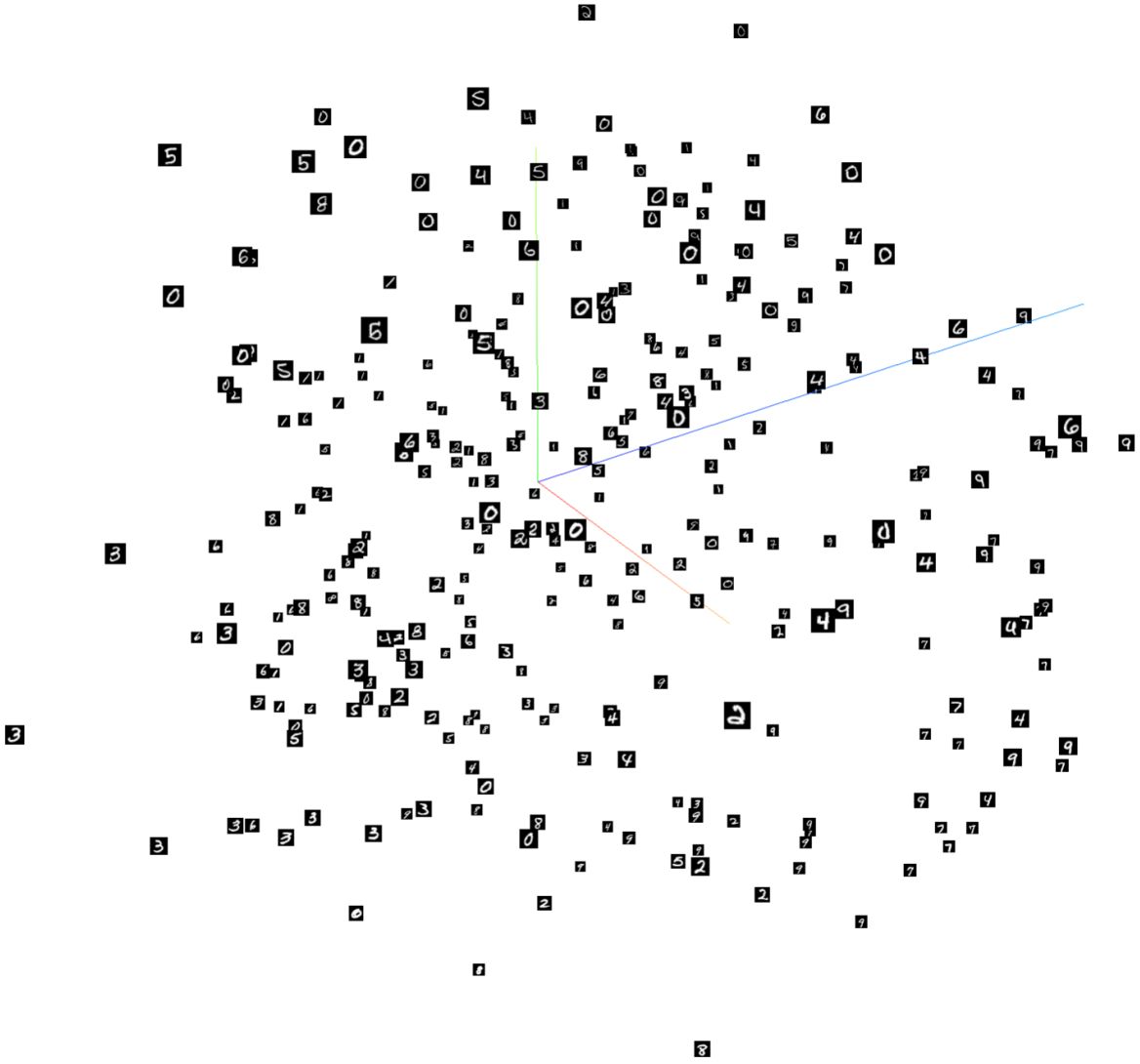

Figure 3.2: Visualisation of a batch of 300 images in our embedding dimension with four layers Net

We show images in our embedding dimensions (The three most important dimensions), We see that the similar images tend to locate in the same region. This presentation describe only 27.0% of total variance.