RĪGAS TEHNISKĀ UNIVERSITĀTE
Datorzinātnes un informācijas tehnoloģijas fakultāte
Lietišķo datorsistēmu institūts
Mākslīgā intelekta un sistēmu inženierijas katedra

Oļegs Korsaks
Bakalaura studiju programmas „Datorsistēmas”
students, stud. apl. nr. 051RDB146

SALIDZINOŠĀ ANALĪZE DATU KOPU FORMĀTIEM PYTORCH ATTĒLU KLASIFIKĀCIJAS UZDEVUMIEM

Atskaite par bakalaura darbu

Zinātniskais vadītājs
Mg.sc.ing, Pētnieks
ĒVALDS URTĀNS

Rīga 2022

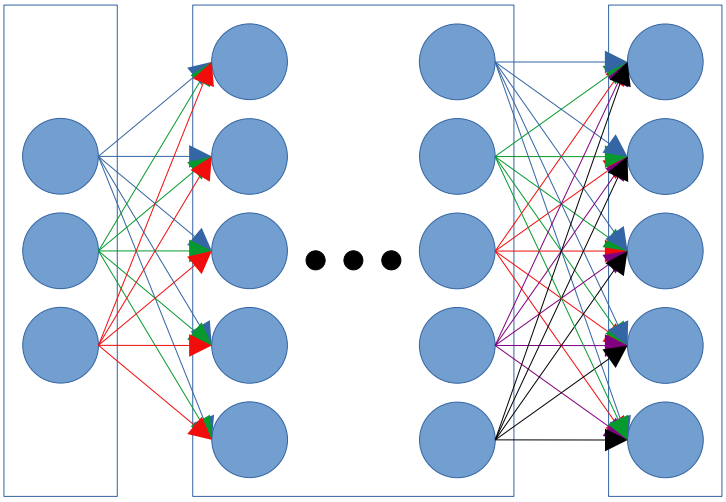
Table of Contents

1. Ievads

1.1. Dziļā mašinmācīšanās

Pamata arhitektūras

Makslīgais neironu tīkls sastāv no viena ieejas slāņa, viena vai dažiem slēptiem slāņiem, ka arī viena izejas slāņa. Katrs slānis satur vienu vai vairākus neironus. Tie ir saistīti ar blakusslāņu neironiem.



Eksistē tīkli, kuru neironi tiek pilnīgi saistīti ar visiem neironiem blakusslāņos, bet var būt saistīti tikai daļēji. Katrai saitei ir savs svars un nobīde. Tie ir apmācamie parametri. Apmācīšanas procesa mērķis ir atrast tādus svarus un nobīdes visām saitēm, ar kuriem tīkls veidotu pareizus rezultātus izejas slānī.

Linārie slāņi

Although there is no strong formal theory on how to select the neural network layers and configuration, and although the only way to tune **some** hyper-parameters is just by trial and error (meta-learning for instance), there are still some heuristics, guidelines, and theories that can still help us reduce the search space of suitable architectures considerably. In a previous blog post, we introduced the inner mechanics of neural networks. In this series of blog posts we will talk about the basic layers, their rationale, their complexity, and their computation capabilities.

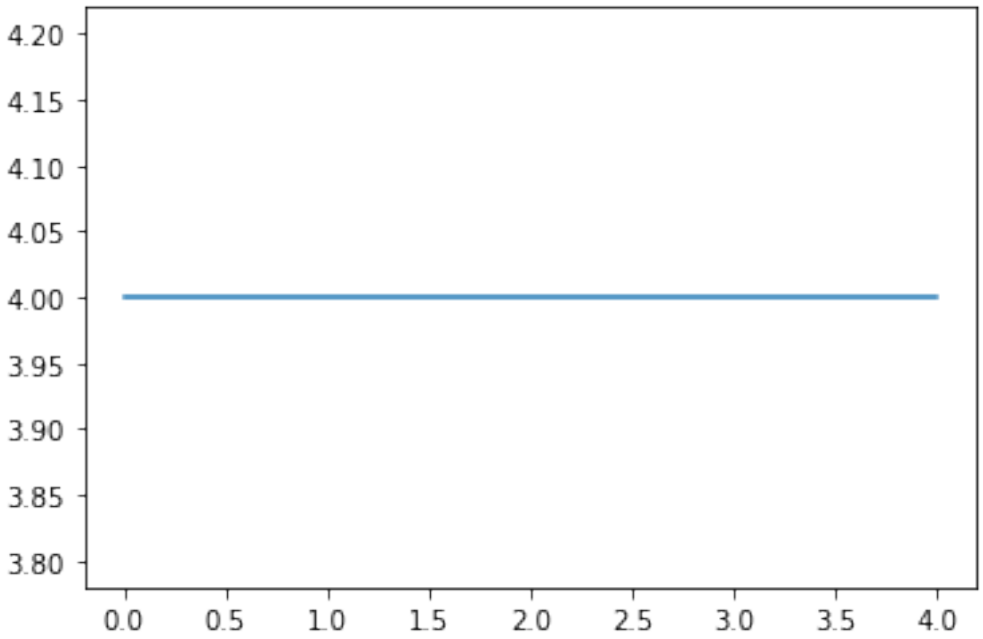
Lineārais nobīdes slānis

$$y = b$$

This layer is basically learning a constant. It’s capable of learning an offset, a bias, a threshold, or a mean. If we create a neural network only from this layer and train it over a dataset, the mean square

error (MSE) loss will force this layer to converge to the mean or average of the outputs.

For instance, if we have the following dataset {2,2,3,3,4,4}, and we're forcing the neural network to compress it to a unique value ***b***, the most logical convergence will be around the value **b=3** (which is the average of the dataset to reduce the losses to the maximum. We can see that learning a constant is kind of learning a DC value component of an electric circuit, or an offset, or a ground truth to compare to. Any value above this offset will be positive, any value below it will be negative. It's like redefining where the offset 0 should start from.

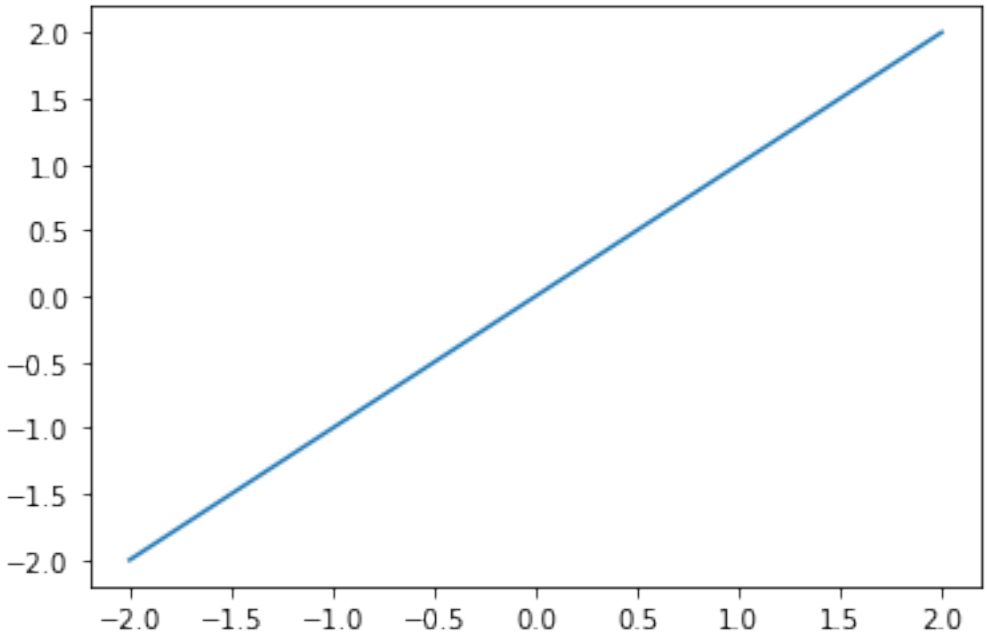


Lineārais slānis

$$y = W \cdot x$$

A linear layer without a bias is capable of learning an average rate of correlation between the output and the input, for instance if x and y are positively correlated => w will be positive, if x and y are negatively correlated => w will be negative. If x and y are totally independent => w will be around 0.

Another way of perceiving this layer: Consider a new variable A=y/x. and use the "bias layer" from the previous section, as we said before, it will learn the average or the mean of A. (which is the average of output/input thus the average of the rate to which the output is changing relatively to the input).



Lineārais padeves slānis

$y = W \cdot x + b$

A Feed-forward layer is a combination of a linear layer and a bias. It is capable of learning **an offset and a rate of correlation**. Mathematically speaking, *it represents an equation of a line*. In term of capabilities:

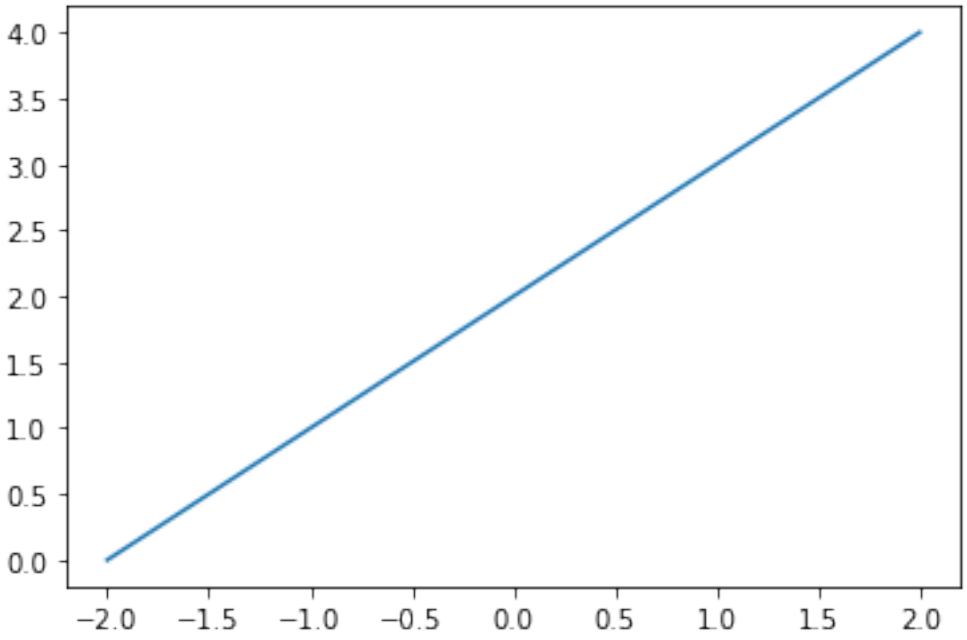
- This layer is able to replace both a linear layer and a bias layer.
- By learning that **w=0** => we can reduce this layer to a pure bias layer.
- By learning that **b=0** => we can reduce this layer to a pure linear layer.
- A linear layer with bias can represent **PCA** (for dimensionality reduction). Since PCA is actually just combining linearly the inputs together.
- A linear feed-forward layer **can learn scaling** automatically. Both a MinMaxScaler or a StandardScaler can be modeled through a linear layer.

By learning **w=1/(max-min)** and **b=-min/(max-min)** a linear feed-forward is capable of simulating a MinMaxScaler

$$y = \frac{x - min}{max - min} => y = \frac{1}{max - min}x - \frac{min}{max - min} => y = w.x + b$$

Similarly, by learning w=1/std and b=-avg/std, a linear feed-forward is capable of simulating a StandardScaler

$$y = \frac{x - avg}{std} => y = \frac{1}{std}x - \frac{avg}{std} => y = w.x + b$$



Aktivizācijas funkcijas

Softmax (priekš klasifikācijas)

$$L(y, y') = -\frac{1}{N}\sum y \cdot \log(y')$$

$$\frac{\partial L(y, y')}{\partial y'} = -y \cdot \frac{1}{y'} = -\frac{y}{y'}$$

$$SoftMax(y = j|x) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$$

$$\begin{bmatrix} a_0(1-a_0) & -a_0 a_1 & -a_0 a_2 \\ -a_1 a_0 & a_1(1-a_1) & -a_1 a_2 \\ -a_2 a_0 & -a_2 a_1 & a_2(1-a_2) \end{bmatrix}$$

Kļūdas funkcijas

Kļūdas funkcija palīdz noteikt cik tālu tekoša prognozēta vērtība ir no patiesas. Un ja to pielietot visiem datu eksemplāriem – ar to var noteikt, cik labi tekošais modelis var prognozēt rezultātus kopumā. Ideālā gadījumā kļūdai jābūt vienādai nullei, gan apmācības datu eksemplāriem, gan pārbaudes. Tātad apmācība cenšas kļūdu samazināt.

MAE

Mean absolute error vai vidēja absolūta kļūda:

$$L_{MAE} = \frac{1}{N} \cdot \sum_{i=0}^{N} \left| (h_\theta(x_i) - y_i) \right|$$

Tā ir vidēja absolūta starpība starp pareizas un prognozētas vertībām. Kļūda pieaug lineāri un tiek pielietota regresijas uzdevumiem, kuru rezultāts ir viena vērtība.

MSE

Mean squared error vai vidēja kvadrātiskā kļūda:

$$L_{MSE} = \frac{1}{N} \cdot \sum_{i=0}^{N} (h_\theta(x_i) - y_i)^2$$

Tā ir vidēja starpība starp pareizas un prognozētas vertībām, kas tiek pacelta kvadrātā. Kvadrāts palīdz izvairīties no negatīvām vērtībām. Kā arī kļūdas vērtība pieaug straujāk, salīdzinot ar MAE. Tiek pielietota regresijas uzdevumiem.

CCE

Categorical cross-entropy vai kategoriska krustentropija.

Pielieto klasifikācijas uzdevumiem, kad vajag izvelēties tikai vienu klasi no dažādam.

BCE

Binary cross-entropy vai bināra krustentropija.

Pielieto klasifikācijas uzdevumiem, kad modelim atļauts izvēlēties par pareizo vairāk nekā vienu klasi.

Atpakaļizplatīšanās algoritms

Šīs algoritms cenšas mainīt tīkla svarus un nobīdes tā, lai kļūda būtu 0.

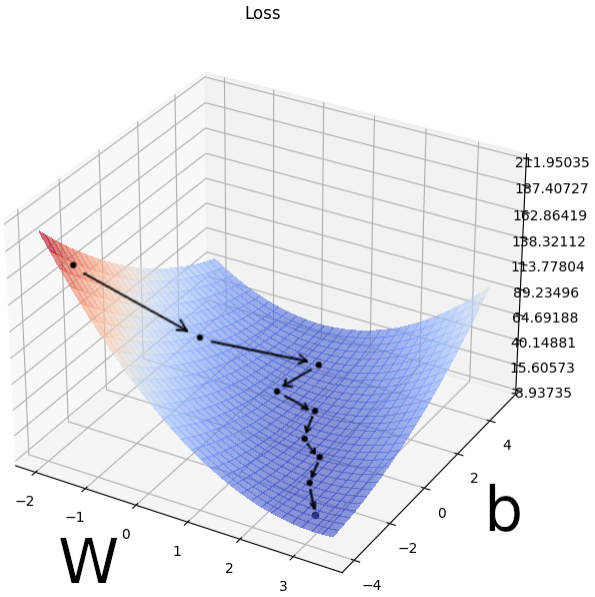
$$\theta_0 := \theta_0 - \alpha \cdot \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$b := b - \alpha \cdot \frac{\partial}{\partial b} J(b, W)$$

$$\theta_1 := \theta_1 - \alpha \cdot \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$W := W - \alpha \cdot \frac{\partial}{\partial W} J(b, W)$$

Pieņemsim, ka modelis ir:

$$y' = M(x) = Linear(W_1, b_1, W_2, b_2, x) = Linear(W_2, b_2, ReLU(Linear(W_1, b_1, x)))$$
$$Linear(W_i, b_i, x_i) = W_i \cdot x_i + b_i$$
$$ReLU(x_i) = \begin{cases} x_i, x_i \geq 0 \\ 0, x_i < 0 \end{cases}$$

Kur, piemēram, $J(\theta_0, \theta_1) = L_{MAE} = \frac{1}{N} \cdot \sum_{i=0}^{N} (h_\theta(x_i) - y_i)$ ir MAE kļūdas funkcija un α ir apmācības koeficients, kurš noteic, cik strauji svars W un nobīde b tiek mainīti. Kļūdas funkcijas atvasinājums noteic vai parametru ir jāpalielina, vai jāsāmazina un uz kādu lielumu.



Apmācāmo parametru optizācijas algoritmi

Stohastiskā gradienta nolaišanās algoritms (SGD)

The goal of this is to update θ_0, b and θ_1, W to minimize cost function result.

$$\theta_0 := \theta_0 - \alpha \cdot \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$b := b - \alpha \cdot \frac{\partial}{\partial b} J(b, W)$$

$$\theta_1 := \theta_1 - \alpha \cdot \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$W := W - \alpha \cdot \frac{\partial}{\partial W} J(b, W)$$

but to simultaneously update both variables we need to assign them to temporary variables first, so b doesn't affect calculation of the W.

α - is a learning rate, which defines how fast we are going to change b and W.

By intuition what is going to happen – partial derivative of the cost function will be a positive slope (will return a positive number) if the mse will on the right side of the local minima, as a result we will subtract a slope value multiplied by learning rate from W (or b). Otherwise we will add it. In ideal situation once MSE is 0 – we don't move anymore.

So let's try to figure out partial derivatives:

For θ_0 or b :

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_0} \cdot \frac{1}{N} \cdot \sum_{i=0}^{N} (h_\theta(x_i) - y_i)^2 = \frac{\partial}{\partial \theta_0} \cdot \frac{1}{N} \cdot \sum_{i=0}^{N} (\theta_0 + \theta_1 x_i - y_i)^2 = \frac{2}{N} \sum_{i=0}^{N} (\theta_0 + \theta_1 x_i - y_i) \cdot 1$$

equals to:

$$\frac{2}{N} \sum_{i=0}^{N} (\theta_0 + \theta_1 x_i - y_i) \cdot 1 = \frac{2}{N} \sum_{i=0}^{N} (b + W \cdot x_i - y_i) \cdot 1$$

and for θ_1 or W :

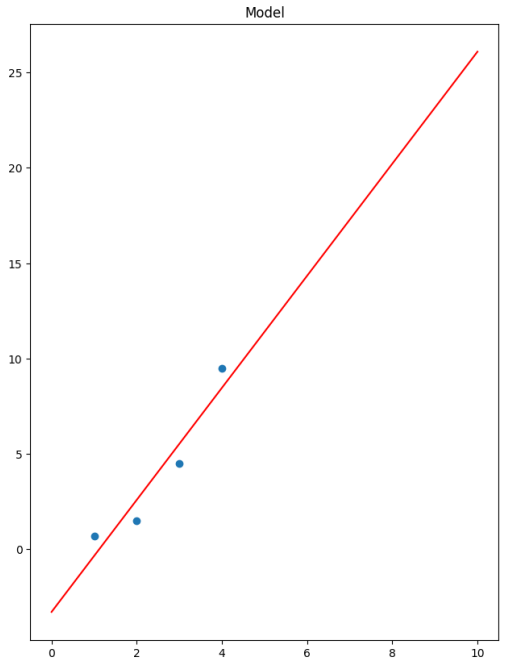
$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_1} \cdot \frac{1}{N} \cdot \sum_{i=0}^{N} (h_\theta(x_i) - y_i)^2 = \frac{\partial}{\partial \theta_1} \cdot \frac{1}{N} \cdot \sum_{i=0}^{N} (\theta_0 + \theta_1 x_i - y_i)^2 = \frac{2}{N} \sum_{i=0}^{N} (\theta_0 + \theta_1 x_i - y_i) \cdot x_i$$

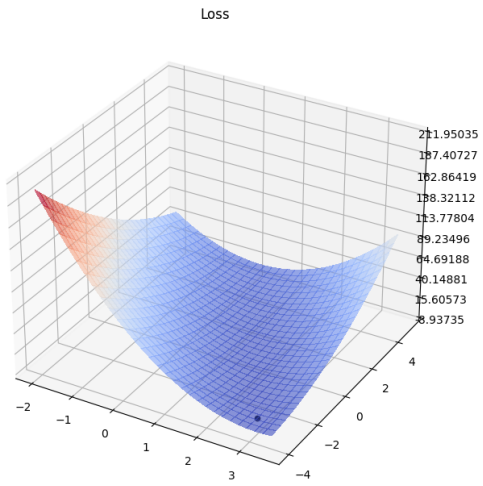
equals to:

$$\frac{2}{N} \sum_{i=0}^{N} (\theta_0 + \theta_1 x_i - y_i) \cdot x_i = \frac{2}{N} \sum_{i=0}^{N} (b + W \cdot x_i - y_i) \cdot x_i$$

For linear model:

w=2.939305864784624 b=-3.2979591578193053 loss=1.103000695674468 learning_rate=0.0001490000000000007





For sigmoid model:

$$\frac{1}{1+e^{-x}}$$

We need to find a gradient descent:

$$\theta_0 := \theta_0 - \alpha \cdot \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$b := b - \alpha \cdot \frac{\partial}{\partial b} J(b, W)$$

$$\theta_1 := \theta_1 - \alpha \cdot \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$W := W - \alpha \cdot \frac{\partial}{\partial W} J(b, W)$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_0} \cdot \frac{1}{N} \cdot \sum_{i=0}^{N} (h_\theta(x_i) - y_i)^2$$

where $h_\theta(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 \cdot x)}} = \frac{1}{1+e^{-(b+W \cdot x)}}$

let's substitute $a = b + W \cdot x$ so $h_\theta(x) = \frac{1}{1+e^{-a}}$

$$\frac{\partial}{\partial a}\frac{1}{1+e^{-a}}=\frac{\partial}{\partial a}(1+e^{-a})^{-1}=-(1+e^{-a})^{-2}\cdot\frac{\partial}{\partial a}(1+e^{-a})=-(1+e^{-a})^{-2}\cdot(\frac{\partial}{\partial a}1+\frac{\partial}{\partial a}e^{-a})=$$

$$=-(1+e^{-a})^{-2}\cdot(0+e^{-a}\cdot\frac{\partial}{\partial a}[-a])=-(1+e^{-a})^{-2}\cdot(e^{-a}\cdot-1)=\frac{e^{-a}}{(1+e^{-a})^2}=$$

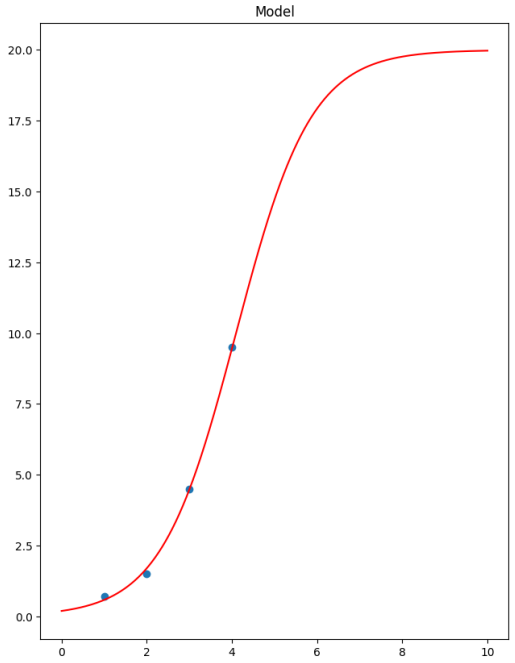
$$=\frac{e^{-a}}{(1+e^{-a})\cdot(1+e^{-a})}=\frac{1\cdot e^{-a}}{(1+e^{-a})\cdot(1+e^{-a})}=\frac{1}{1+e^{-a}}\cdot\frac{e^{-a}}{1+e^{-a}}=$$

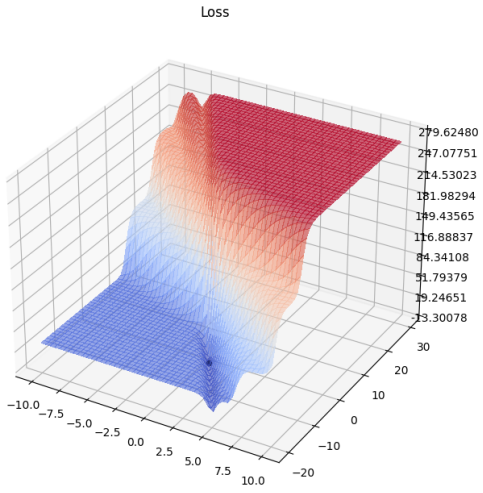
$$=\frac{1}{1+e^{-a}}\cdot\frac{e^{-a}+1-1}{1+e^{-a}}=\frac{1}{1+e^{-a}}\cdot(\frac{1+e^{-a}}{1+e^{-a}}-\frac{1}{1+e^{-a}})=\frac{1}{1+e^{-a}}\cdot(1-\frac{1}{1+e^{-a}})$$

$$\frac{\partial}{\partial\theta_0}\frac{1}{1+e^{-a}}=\frac{\partial}{\partial b}\frac{1}{1+e^{-a}}=\frac{\partial}{\partial a}\cdot\frac{\partial}{\partial b}=\frac{e^{-a}}{(1+e^{-a})^2}\cdot 1=\frac{1}{1+e^{-a}}\cdot(1-\frac{1}{1+e^{-a}})\cdot 1$$

$$\frac{\partial}{\partial\theta_1}\frac{1}{1+e^{-a}}=\frac{\partial}{\partial W}\frac{1}{1+e^{-a}}=\frac{\partial}{\partial a}\cdot\frac{\partial}{\partial W}=\frac{e^{-a}}{(1+e^{-a})^2}\cdot x=\frac{1}{1+e^{-a}}\cdot(1-\frac{1}{1+e^{-a}})\cdot x$$

w=1.1346469402126882 b=-4.6482628586296135 loss=0.014018127818400896 learning_rate=0.0001490000000000007

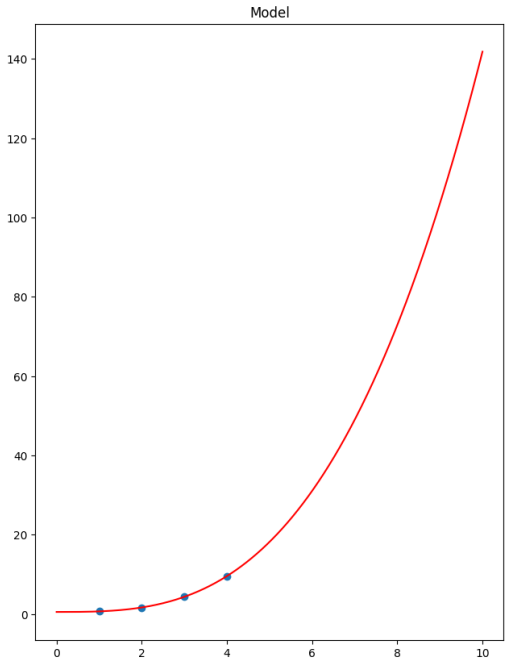


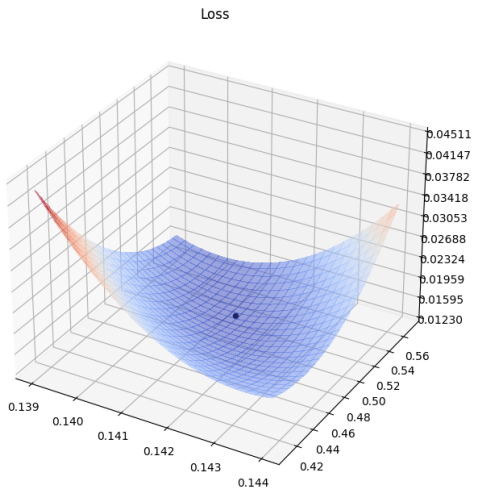


It looks cool but I don't find sigmoid logical here because there is no chance that 10 floor building may cost same as 100 floor. What about cubic $b+W\cdot x^3$? Where b would shift the line vertically and W would regulate its width. That was my intuition on how to fit the line. The nice part here is that loss function derivative dW, db is the same as for linear. And dx is:

$$dx\,cubic: \frac{\partial}{\partial x}[W\cdot x^3+b]=W\cdot 3\cdot x^2$$

w=0.141346153846152 b=0.5163461538462171 loss=0.013832747781064567 learning_rate=0.0014859999999999995





RMSprop

Metrikas

novērtēt cik labs rezultāts

Accuracy

F1

1.2. Attēlu klasifikācija

Konvolūcijas tikls (ConvNet)

ResNet / DenseNet

InceptionNet

1.3. PyTorch vide

PyTorch ir optimizēta mašīnmācīšanas bibliotēka, kura atvieglo darbu gan ar procesoru (CPU), gan ar videokārti (GPU). Programēšanas valoda ir Python.

Modeļu implementāciju <> viendājumi

Datu ielādes process

Datu ielādes process sastāv no datu kopas (Dataset) un datu ielādes (DataLoader) klasēm. Dataset klase ir atbildīga par datu kopas izmēra noteikšanu caur **__len__()** metodi un pēc indeksa izvēlēta ieraksta atgriešanu caur **__getitem__(index)** metodi. Pašus datus var lejupielādēt no tīmekļa, vai ielādēt no diska inicializācijas solī.

```
class CustomDataset(torch.utils.data.Dataset):
    def __init__(self):
        super().__init__()
        self.x = [1, 2, 3, 4, 5, 6]
        self.y = [0, 1, 0, 1, 0, 1]

    def __getitem__(self, index):
        return self.x[index], self.y[index]

    def __len__(self):
        return len(self.x)
```

Pēc datu kopas definīcijas

2. Metodoloģija

2.1. Datu ielādes metodes

2.1.1. Failu sistēma

2.1.2. NumPy mmap

2.1.3. CuPy mmap

2.1.4. HDF5

2.1.5. Zarr

2.2. Datu kopas

2.2.1. CIFAR10

2.2.2. Tiny ImageNet

.. vel kādas

^ Atrast HiRes datu kopas priekš klasifikācijas

^ Atrast dažādu izmēru attēlu datu kopas

^ 3 dažādu izmēru / kofigurāciju datu kopas

2.3. Apmācības protokols

* Jāizdomā metrikas kā noteikt ietekmi datu ielādes metodēm

* Vidējais epocha ātrums sekundēs

3. Rezultāti

Datu ielādes metodes PRET datu kopām

4. Tālākie pētījumi

5. Secinājumi