Report R Text Classification Problem

Team Number:

SHUANG WANG sxw171930

YING ZHOU yxz179830

In this homework, we use deep learning algorithms to perform text classification with keras, and tm packages. The dataset (<https://archive.ics.uci.edu/ml/datasets/Roman+Urdu+Data+Set>) of “Roman Urdu Data Set Data Set”, which a data corpus comprising of more than 20000 records was collected.

As we observe the dataset, this dataset owns two columns. Each record comprises of two string datatype values. The first one for Comment/Review and the second column for sentiment. Although the description of dataset builds the info that the tagged for sentiment only has “Positive”, “Negative”, and “Neutral” labels. We check for the correctness of the second column and find that there are 4 levels with “Neative”, “Negative”, “Neutral”, and “Positive”. Apparently, there must be something wrong with “Neative”. So we change the “Neative” into “Negative”. We store the labels into the “label\_trail” and change it into numeric as “Negative” =2, “Neutral” =1, “Positive” =0 within label.

Next, we would like to introduce some natural language processing (NLP) to build language model for implementing linguistic and terminological approaches for text mining. For instance, word tokenization, word stemming and so on.

We will use the package tm (for text processing) in this homework.

Before modeling, data preprocessing and cleaning are necessary to reduce noises and useless information. We usually perform the following steps for lexical normalization:

(1). convert to lower cases

(2). remove punctuation

(3). remove numbers

(4). remove white space

(5). remove stopwords (very common and non-specific words such as a, the, is, …)

(6). perform word stemming (e.g. computers, computer, computing, compute -> comput)

(7). tokenize the corpus into words

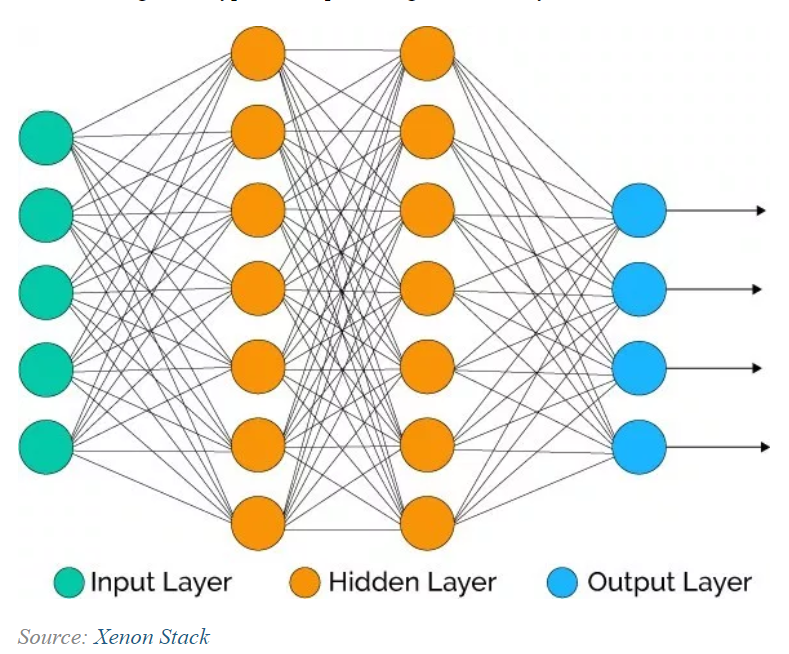
What we need to pay attention to is that the stopwords are not in English but in Roman. Luckily, the tm package owns the stopwords with Roman. After lexical normalization, the next step will be generating a document-term matrix for further analyses and deep learning tasks. Thankfully, tm package also provides a function DocumentTermMatrix (or TermDocumentMatrix, if you need the reverse one) to generate the document-term matrix using sparse matrix.

Building A deep learning Model

Hidden layers of the neural network can learn an interesting representation of the data. We can use the hidden layer representation for many things, for example, dimensionality reduction and feature learning. In this part we’d like to stack multiple neural network layers (multilayer autoencoder) to learn the hidden representation of data.

Next, we need to prepare the input data for deep neural network.

Free text needs to be converted to index sequence for Keras input. text\_tokenizer and fit\_text\_tokenizer functions help us tokenize our raw data, and texts\_to\_sequences function converts our textual data to a list of index sequences. Since lengths of comments are all different, we use pad\_sequences to pad the sequences (pad the sequences with 0 to the left), and let all notes (now index sequences) have all the same length. We can utilize head(data\_idx) to see how the data looks like after transformation.



We’ll build a four-layer MLP with Keras. And the steps are in the following:

(1). **Initialize a sequential model**: The first step is to initialize a sequential model with keras\_model\_sequential(), which is the beginning of our Keras model. The sequential model is composed of a linear stack of layers.

(2). **Apply layers to the sequential model:** Layers consist of the input layer, hidden layers and an output layer. The input layer is the data and provided it’s formatted correctly there’s nothing more to discuss. The hidden layers and output layers are what controls the ANN inner workings.

(3**). Compile the model**: The last step is to compile the model with compile (). We’ll use optimizer = "adam", which is one of the most popular optimization algorithms. We select loss = "sparse\_categorical\_crossentropysince this is a binary classification problem. We’ll select metrics = c("accuracy") to be evaluated during training and testing.

And then we try to build a deep training model with one input layer, four hidden layer and one output layer. The specific parameters are shown in R code. The result can be displayed with the following history graph.

The partial result of this model is:

Epoch 1/50

14160/14160 [==============================] - 3s 181us/step - loss: 1.1122 - acc: 0.3316

Epoch 2/50

14160/14160 [==============================] - 2s 153us/step - loss: 1.5598 - acc: 0.3456

Epoch 3/50

14160/14160 [==============================] - 2s 157us/step - loss: 1.7267 - acc: 0.3437

Epoch 4/50

14160/14160 [==============================] - 2s 156us/step - loss: 1.0681 - acc: 0.3349

Epoch 5/50

14160/14160 [==============================] - 2s 156us/step - loss: 1.0671 - acc: 0.3377

Epoch 6/50

14160/14160 [==============================] - 2s 155us/step - loss: 1.0719 - acc: 0.3344

Epoch 7/50

14160/14160 [==============================] - 2s 158us/step - loss: 1.0643 - acc: 0.3385

Epoch 8/50

14160/14160 [==============================] - 2s 157us/step - loss: 1.0571 - acc: 0.3597

Epoch 9/50

14160/14160 [==============================] - 2s 156us/step - loss: 1.0460 - acc: 0.3852

Epoch 10/50

14160/14160 [==============================] - 2s 157us/step - loss: 1.0345 - acc: 0.4226

…

…

…

Epoch 40/50

14160/14160 [==============================] - 2s 156us/step - loss: 1.0130 - acc: 0.4812

Epoch 41/50

14160/14160 [==============================] - 2s 158us/step - loss: 1.0199 - acc: 0.4551

Epoch 42/50

14160/14160 [==============================] - 2s 157us/step - loss: 1.0160 - acc: 0.4639

Epoch 43/50

14160/14160 [==============================] - 2s 158us/step - loss: 1.0106 - acc: 0.4870

Epoch 44/50

14160/14160 [==============================] - 2s 158us/step - loss: 1.0135 - acc: 0.4936

Epoch 45/50

14160/14160 [==============================] - 2s 158us/step - loss: 1.0138 - acc: 0.4826

Epoch 46/50

14160/14160 [==============================] - 2s 157us/step - loss: 1.0169 - acc: 0.4672

Epoch 47/50

14160/14160 [==============================] - 2s 158us/step - loss: 1.0126 - acc: 0.4758

Epoch 48/50

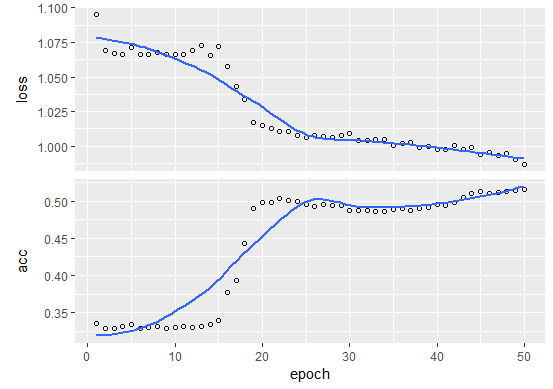
14160/14160 [==============================] - 2s 159us/step - loss: 1.0122 - acc: 0.4859

Epoch 49/50

14160/14160 [==============================] - 2s 159us/step - loss: 1.0123 - acc: 0.4940

Epoch 50/50

14160/14160 [==============================] - 2s 158us/step - loss: 1.0119 - acc: 0.4922



Furthermore, we will apply the model on the test part and see how well it does. Obviously, the accuracy of the test result **is 0.391**, but the accuracy of the training result is **0.4922**, which is a big difference.

> model %>% evaluate(x\_test, y\_test)

6069/6069 [==============================] - 0s 13us/step

$loss

[1] 1.130388

$acc

[1] 0.3908387

Next, we will improve the performance with varying the parameters and run our code and report the parameters used and accuracy obtained.

Vary Parameters 1:

Epoch 1/50

14160/14160 [==============================] - 9s 621us/step - loss: 1.1092 - acc: 0.3474

Epoch 2/50

14160/14160 [==============================] - 9s 629us/step - loss: 1.0924 - acc: 0.3403

Epoch 3/50

14160/14160 [==============================] - 9s 633us/step - loss: 1.0761 - acc: 0.3349

Epoch 4/50

14160/14160 [==============================] - 16s 1ms/step - loss: 1.0696 - acc: 0.3339

Epoch 5/50

14160/14160 [==============================] - 9s 644us/step - loss: 1.1521 - acc: 0.3431

Epoch 6/50

14160/14160 [==============================] - 9s 629us/step - loss: 1.0734 - acc: 0.3314

Epoch 7/50

14160/14160 [==============================] - 9s 632us/step - loss: 1.0706 - acc: 0.3337

Epoch 8/50

14160/14160 [==============================] - 9s 621us/step - loss: 1.3405 - acc: 0.3631

Epoch 9/50

14160/14160 [==============================] - 12s 853us/step - loss: 1.0825 - acc: 0.3355

Epoch 10/50

14160/14160 [==============================] - 9s 665us/step - loss: 1.1154 - acc: 0.3359

Epoch 40/50

…

…

…

14160/14160 [==============================] - 9s 618us/step - loss: 1.4066 - acc: 0.3362

Epoch 41/50

14160/14160 [==============================] - 9s 623us/step - loss: 1.3412 - acc: 0.3297

Epoch 42/50

14160/14160 [==============================] - 9s 627us/step - loss: 1.2837 - acc: 0.3314

Epoch 43/50

14160/14160 [==============================] - 9s 631us/step - loss: 1.3330 - acc: 0.3547

Epoch 44/50

14160/14160 [==============================] - 9s 635us/step - loss: 1.0691 - acc: 0.3326

Epoch 45/50

14160/14160 [==============================] - 9s 631us/step - loss: 1.0963 - acc: 0.3417

Epoch 46/50

14160/14160 [==============================] - 9s 616us/step - loss: 1.0729 - acc: 0.3333

Epoch 47/50

14160/14160 [==============================] - 9s 622us/step - loss: 1.1075 - acc: 0.3351

Epoch 48/50

14160/14160 [==============================] - 9s 627us/step - loss: 1.0662 - acc: 0.3311

Epoch 49/50

14160/14160 [==============================] - 9s 626us/step - loss: 1.0688 - acc: 0.3326

Epoch 50/50

14160/14160 [==============================] - 9s 630us/step - loss: 1.0667 - acc: 0.3321

And the evaluation of this model with graph:

> model\_1 %>% evaluate(x\_test, y\_test)

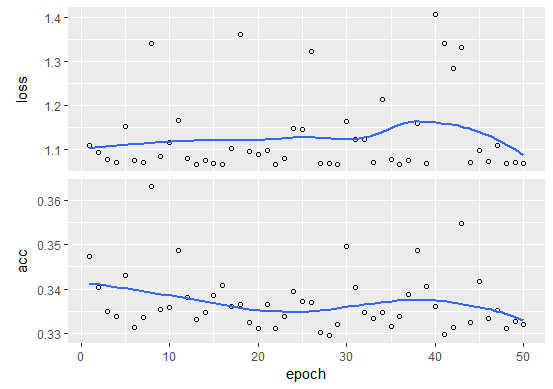
6069/6069 [==============================] - 0s 16us/step

$loss

[1] 1.105464

$acc

[1] 0.3257538



Vary Parameters 2:

Epoch 1/50

14160/14160 [==============================] - 2s 146us/step - loss: 1.1600 - acc: 0.3428

Epoch 2/50

14160/14160 [==============================] - 2s 153us/step - loss: 1.0658 - acc: 0.3369

Epoch 3/50

14160/14160 [==============================] - 2s 132us/step - loss: 1.0659 - acc: 0.3403

Epoch 4/50

14160/14160 [==============================] - 2s 133us/step - loss: 1.0655 - acc: 0.3438

Epoch 5/50

14160/14160 [==============================] - 2s 134us/step - loss: 1.0656 - acc: 0.3368

Epoch 6/50

14160/14160 [==============================] - 2s 130us/step - loss: 1.0653 - acc: 0.3389

Epoch 7/50

14160/14160 [==============================] - 2s 134us/step - loss: 1.0896 - acc: 0.3326

Epoch 8/50

14160/14160 [==============================] - 2s 137us/step - loss: 1.0725 - acc: 0.3391

Epoch 9/50

14160/14160 [==============================] - 2s 145us/step - loss: 1.0719 - acc: 0.3342

Epoch 10/50

14160/14160 [==============================] - 2s 159us/step - loss: 1.0660 - acc: 0.3364

…

…

…

Epoch 40/50

14160/14160 [==============================] - 2s 131us/step - loss: 1.0207 - acc: 0.4607

Epoch 41/50

14160/14160 [==============================] - 2s 132us/step - loss: 1.0198 - acc: 0.4595

Epoch 42/50

14160/14160 [==============================] - 2s 133us/step - loss: 1.0192 - acc: 0.4627

Epoch 43/50

14160/14160 [==============================] - 2s 131us/step - loss: 1.0174 - acc: 0.4624

Epoch 44/50

14160/14160 [==============================] - 2s 133us/step - loss: 1.0173 - acc: 0.4624

Epoch 45/50

14160/14160 [==============================] - 2s 133us/step - loss: 1.0183 - acc: 0.4614

Epoch 46/50

14160/14160 [==============================] - 2s 132us/step - loss: 1.0175 - acc: 0.4612

Epoch 47/50

14160/14160 [==============================] - 2s 142us/step - loss: 1.0188 - acc: 0.4583

Epoch 48/50

14160/14160 [==============================] - 2s 136us/step - loss: 1.0188 - acc: 0.4629

Epoch 49/50

14160/14160 [==============================] - 2s 133us/step - loss: 1.0173 - acc: 0.4633

Epoch 50/50

14160/14160 [==============================] - 2s 139us/step - loss: 1.0189 - acc: 0.4621

And the evaluation of this model with graph:

> model\_2 %>% evaluate(x\_test, y\_test)

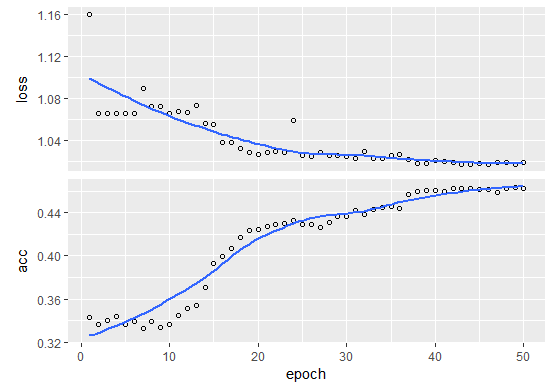
6069/6069 [==============================] - 0s 53us/step

$loss

[1] 1.117797

$acc

[1] 0.3870489



Vary Parameters 3:

Epoch 1/50

14160/14160 [==============================] - 3s 185us/step - loss: 1.1191 - acc: 0.3362

Epoch 2/50

14160/14160 [==============================] - 3s 177us/step - loss: 1.0664 - acc: 0.3324

Epoch 3/50

14160/14160 [==============================] - 3s 179us/step - loss: 1.0701 - acc: 0.3357

Epoch 4/50

14160/14160 [==============================] - 2s 173us/step - loss: 1.0671 - acc: 0.3314

Epoch 5/50

14160/14160 [==============================] - 2s 173us/step - loss: 1.0664 - acc: 0.3284

Epoch 6/50

14160/14160 [==============================] - 2s 172us/step - loss: 1.0666 - acc: 0.3285

Epoch 7/50

14160/14160 [==============================] - 2s 172us/step - loss: 1.0667 - acc: 0.3289

Epoch 8/50

14160/14160 [==============================] - 2s 173us/step - loss: 1.0666 - acc: 0.3290

Epoch 9/50

14160/14160 [==============================] - 2s 176us/step - loss: 1.0678 - acc: 0.3311

Epoch 10/50

14160/14160 [==============================] - 2s 173us/step - loss: 1.0665 - acc: 0.3285

…

…

…

Epoch 40/50

14160/14160 [==============================] - 2s 173us/step - loss: 5.2601 - acc: 0.4772

Epoch 41/50

14160/14160 [==============================] - 2s 170us/step - loss: 5.2603 - acc: 0.4772

Epoch 42/50

14160/14160 [==============================] - 2s 171us/step - loss: 5.2603 - acc: 0.4772

Epoch 43/50

14160/14160 [==============================] - 2s 173us/step - loss: 5.2603 - acc: 0.4772

Epoch 44/50

14160/14160 [==============================] - 2s 175us/step - loss: 5.2602 - acc: 0.4772

Epoch 45/50

14160/14160 [==============================] - 2s 175us/step - loss: 5.2603 - acc: 0.4772

Epoch 46/50

14160/14160 [==============================] - 2s 174us/step - loss: 5.2602 - acc: 0.4772

Epoch 47/50

14160/14160 [==============================] - 2s 174us/step - loss: 5.2603 - acc: 0.4772

Epoch 48/50

14160/14160 [==============================] - 2s 174us/step - loss: 5.2602 - acc: 0.4772

Epoch 49/50

14160/14160 [==============================] - 2s 173us/step - loss: 5.2604 - acc: 0.4772

Epoch 50/50

14160/14160 [==============================] - 2s 173us/step - loss: 5.2603 - acc: 0.4772

And the evaluation of this model with graph:

> model\_3 %>% evaluate(x\_test, y\_test)

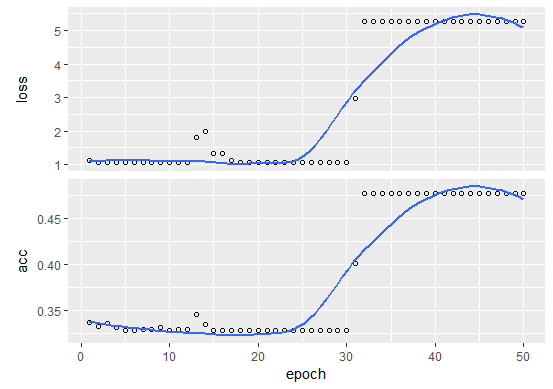
6069/6069 [==============================] - 0s 24us/step

$loss

[1] 5.879735

$acc

[1] 0.3578843



Vary Parameters 4:

Epoch 1/50

14160/14160 [==============================] - 4s 314us/step - loss: 1.0974 - acc: 0.3339

Epoch 2/50

14160/14160 [==============================] - 4s 307us/step - loss: 1.0699 - acc: 0.3285

Epoch 3/50

14160/14160 [==============================] - 4s 301us/step - loss: 1.0667 - acc: 0.3282

Epoch 4/50

14160/14160 [==============================] - 4s 293us/step - loss: 1.0669 - acc: 0.3282

Epoch 5/50

14160/14160 [==============================] - 4s 302us/step - loss: 1.0669 - acc: 0.3285

Epoch 6/50

14160/14160 [==============================] - 4s 301us/step - loss: 1.0673 - acc: 0.3282

Epoch 7/50

14160/14160 [==============================] - 4s 298us/step - loss: 1.0664 - acc: 0.3284

Epoch 8/50

14160/14160 [==============================] - 5s 324us/step - loss: 1.0679 - acc: 0.3280

Epoch 9/50

14160/14160 [==============================] - 4s 317us/step - loss: 1.0667 - acc: 0.3277

Epoch 10/50

14160/14160 [==============================] - 7s 475us/step - loss: 1.0666 - acc: 0.3277

…

…

…

Epoch 41/50

14160/14160 [==============================] - 8s 551us/step - loss: 1.0663 - acc: 0.3277

Epoch 42/50

14160/14160 [==============================] - 8s 553us/step - loss: 1.0663 - acc: 0.3277

Epoch 43/50

14160/14160 [==============================] - 8s 553us/step - loss: 1.0663 - acc: 0.3277

Epoch 44/50

14160/14160 [==============================] - 8s 561us/step - loss: 1.0664 - acc: 0.3277

Epoch 45/50

14160/14160 [==============================] - 8s 552us/step - loss: 1.0663 - acc: 0.3277

Epoch 46/50

14160/14160 [==============================] - 7s 475us/step - loss: 1.0663 - acc: 0.3277

Epoch 47/50

14160/14160 [==============================] - 4s 306us/step - loss: 1.0663 - acc: 0.3277

Epoch 48/50

14160/14160 [==============================] - 4s 301us/step - loss: 1.0663 - acc: 0.3277

Epoch 49/50

14160/14160 [==============================] - 4s 299us/step - loss: 1.0663 - acc: 0.3277

Epoch 50/50

14160/14160 [==============================] - 4s 300us/step - loss: 1.0663 - acc: 0.3277

And the evaluation of this model with graph:

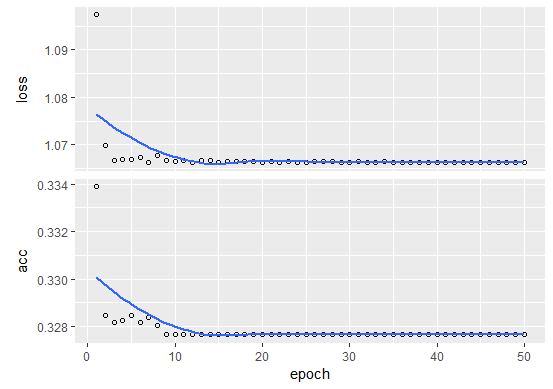
6069/6069 [==============================] - 0s 70us/step

$loss

[1] 1.102459

$acc

[1] 0.3237766



Vary Parameters 5:

Epoch 1/50

14160/14160 [==============================] - 3s 192us/step - loss: 1.1265 - acc: 0.3376

Epoch 2/50

14160/14160 [==============================] - 2s 173us/step - loss: 1.0783 - acc: 0.3364

Epoch 3/50

14160/14160 [==============================] - 2s 169us/step - loss: 1.0665 - acc: 0.3285

Epoch 4/50

14160/14160 [==============================] - 2s 173us/step - loss: 2.3558 - acc: 0.3622

Epoch 5/50

14160/14160 [==============================] - 2s 171us/step - loss: 1.1047 - acc: 0.3577

Epoch 6/50

14160/14160 [==============================] - 3s 184us/step - loss: 1.0598 - acc: 0.3835

Epoch 7/50

14160/14160 [==============================] - 3s 206us/step - loss: 1.0431 - acc: 0.4097

Epoch 8/50

14160/14160 [==============================] - 3s 211us/step - loss: 1.0326 - acc: 0.4191

Epoch 9/50

14160/14160 [==============================] - 2s 173us/step - loss: 1.0279 - acc: 0.4403

Epoch 10/50

14160/14160 [==============================] - 6s 394us/step - loss: 1.0171 - acc: 0.4621

…

…

…

Epoch 41/50

14160/14160 [==============================] - 5s 371us/step - loss: 1.0043 - acc: 0.5037

Epoch 42/50

14160/14160 [==============================] - 5s 339us/step - loss: 1.0036 - acc: 0.5044

Epoch 43/50

14160/14160 [==============================] - 2s 169us/step - loss: 1.0033 - acc: 0.5056

Epoch 44/50

14160/14160 [==============================] - 2s 172us/step - loss: 1.0033 - acc: 0.5047

Epoch 45/50

14160/14160 [==============================] - 2s 169us/step - loss: 1.0218 - acc: 0.5034

Epoch 46/50

14160/14160 [==============================] - 2s 169us/step - loss: 1.0036 - acc: 0.5043

Epoch 47/50

14160/14160 [==============================] - 2s 175us/step - loss: 1.0034 - acc: 0.4992

Epoch 48/50

14160/14160 [==============================] - 2s 174us/step - loss: 1.0024 - acc: 0.5006

Epoch 49/50

14160/14160 [==============================] - 2s 171us/step - loss: 1.0013 - acc: 0.4994

Epoch 50/50

14160/14160 [==============================] - 2s 175us/step - loss: 1.0096 - acc: 0.4996

And the evaluation of this model with graph:

> model\_5 %>% evaluate(x\_test, y\_test)

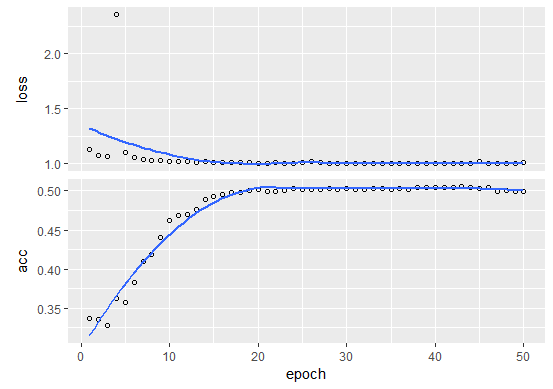
6069/6069 [==============================] - 0s 62us/step

$loss

[1] 1.147017

$acc

[1] 0.3926512



Vary Parameters 6:

Epoch 1/50

14160/14160 [==============================] - 3s 237us/step - loss: 1.1022 - acc: 0.3479

Epoch 2/50

14160/14160 [==============================] - 3s 217us/step - loss: 1.0680 - acc: 0.3295

Epoch 3/50

14160/14160 [==============================] - 3s 215us/step - loss: 1.0681 - acc: 0.3346

Epoch 4/50

14160/14160 [==============================] - 3s 216us/step - loss: 1.0665 - acc: 0.3309

Epoch 5/50

14160/14160 [==============================] - 3s 214us/step - loss: 1.0678 - acc: 0.3306

Epoch 6/50

14160/14160 [==============================] - 3s 216us/step - loss: 1.0668 - acc: 0.3280

Epoch 7/50

14160/14160 [==============================] - 3s 215us/step - loss: 1.0666 - acc: 0.3284

Epoch 8/50

14160/14160 [==============================] - 3s 245us/step - loss: 1.0663 - acc: 0.3283

Epoch 9/50

14160/14160 [==============================] - 3s 218us/step - loss: 1.0664 - acc: 0.3283

Epoch 10/50

14160/14160 [==============================] - 3s 218us/step - loss: 1.8343 - acc: 0.3422

…

…

…

Epoch 40/50

14160/14160 [==============================] - 3s 213us/step - loss: 0.9967 - acc: 0.5066

Epoch 41/50

14160/14160 [==============================] - 3s 225us/step - loss: 0.9943 - acc: 0.5075

Epoch 42/50

14160/14160 [==============================] - 3s 219us/step - loss: 0.9945 - acc: 0.5087

Epoch 43/50

14160/14160 [==============================] - 4s 266us/step - loss: 0.9981 - acc: 0.5012

Epoch 44/50

14160/14160 [==============================] - 3s 238us/step - loss: 0.9957 - acc: 0.4937

Epoch 45/50

14160/14160 [==============================] - 5s 335us/step - loss: 0.9979 - acc: 0.4935

Epoch 46/50

14160/14160 [==============================] - 7s 481us/step - loss: 0.9925 - acc: 0.4968

Epoch 47/50

14160/14160 [==============================] - 5s 340us/step - loss: 0.9908 - acc: 0.4942

Epoch 48/50

14160/14160 [==============================] - 3s 214us/step - loss: 0.9882 - acc: 0.4970

Epoch 49/50

14160/14160 [==============================] - 3s 211us/step - loss: 0.9867 - acc: 0.4970

Epoch 50/50

14160/14160 [==============================] - 3s 214us/step - loss: 0.9908 - acc: 0.4999

And the evaluation of this model with graph:

> model\_6 %>% evaluate(x\_test, y\_test)

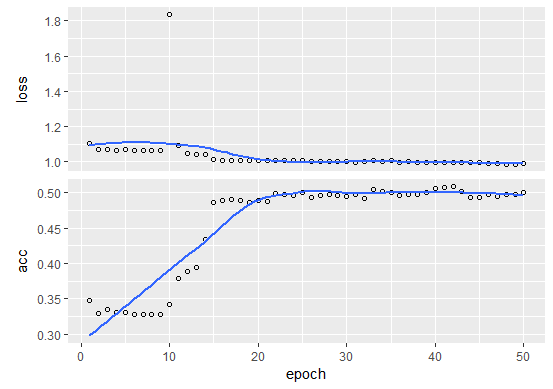
6069/6069 [==============================] - 0s 42us/step

$loss

[1] 1.141961

$acc

[1] 0.3722195



Vary Parameters 7:

Epoch 1/50

14160/14160 [==============================] - 3s 223us/step - loss: 1.1116 - acc: 0.3636

Epoch 2/50

14160/14160 [==============================] - 3s 185us/step - loss: 1.0504 - acc: 0.4516

Epoch 3/50

14160/14160 [==============================] - 3s 193us/step - loss: 1.0312 - acc: 0.4812

Epoch 4/50

14160/14160 [==============================] - 3s 202us/step - loss: 1.0277 - acc: 0.4816

Epoch 5/50

14160/14160 [==============================] - 3s 205us/step - loss: 1.0220 - acc: 0.4839

Epoch 6/50

14160/14160 [==============================] - 5s 329us/step - loss: 1.0236 - acc: 0.4928

Epoch 7/50

14160/14160 [==============================] - 6s 415us/step - loss: 1.0217 - acc: 0.4905

Epoch 8/50

14160/14160 [==============================] - 5s 343us/step - loss: 1.0171 - acc: 0.4939

Epoch 9/50

14160/14160 [==============================] - 3s 182us/step - loss: 1.0170 - acc: 0.4889

Epoch 10/50

14160/14160 [==============================] - 3s 187us/step - loss: 1.0212 - acc: 0.4911

…

…

…

Epoch 40/50

14160/14160 [==============================] - 3s 196us/step - loss: 1.0015 - acc: 0.5081

Epoch 41/50

14160/14160 [==============================] - 3s 184us/step - loss: 1.0014 - acc: 0.5084

Epoch 42/50

14160/14160 [==============================] - 3s 189us/step - loss: 1.0024 - acc: 0.5088

Epoch 43/50

14160/14160 [==============================] - 3s 193us/step - loss: 1.0021 - acc: 0.5085

Epoch 44/50

14160/14160 [==============================] - 3s 191us/step - loss: 1.0017 - acc: 0.5087

Epoch 45/50

14160/14160 [==============================] - 3s 189us/step - loss: 1.0034 - acc: 0.5061

Epoch 46/50

14160/14160 [==============================] - 3s 182us/step - loss: 1.0031 - acc: 0.5069

Epoch 47/50

14160/14160 [==============================] - 3s 192us/step - loss: 1.0022 - acc: 0.5068

Epoch 48/50

14160/14160 [==============================] - 3s 195us/step - loss: 1.0024 - acc: 0.5035

Epoch 49/50

14160/14160 [==============================] - 3s 185us/step - loss: 1.0012 - acc: 0.5070

Epoch 50/50

14160/14160 [==============================] - 3s 192us/step - loss: 1.0015 - acc: 0.5076

And the evaluation of this model with graph:

> model\_7 %>% evaluate(x\_test, y\_test)

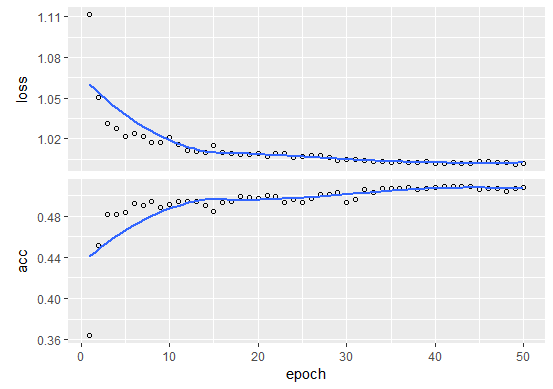
6069/6069 [==============================] - 1s 94us/step

$loss

[1] 1.12255

$acc

[1] 0.3870489



Vary Parameters 8:

Epoch 1/50

14160/14160 [==============================] - 3s 225us/step - loss: 1.0743 - acc: 0.4118

Epoch 2/50

14160/14160 [==============================] - 3s 181us/step - loss: 1.0270 - acc: 0.4843

Epoch 3/50

14160/14160 [==============================] - 3s 178us/step - loss: 1.0200 - acc: 0.4912

Epoch 4/50

14160/14160 [==============================] - 3s 179us/step - loss: 1.0141 - acc: 0.4963

Epoch 5/50

14160/14160 [==============================] - 3s 177us/step - loss: 1.0106 - acc: 0.4923

Epoch 6/50

14160/14160 [==============================] - 3s 181us/step - loss: 1.0108 - acc: 0.4929

Epoch 7/50

14160/14160 [==============================] - 3s 178us/step - loss: 1.0090 - acc: 0.4874

Epoch 8/50

14160/14160 [==============================] - 3s 178us/step - loss: 1.0101 - acc: 0.4883

Epoch 9/50

14160/14160 [==============================] - 3s 178us/step - loss: 1.0080 - acc: 0.4886

Epoch 10/50

14160/14160 [==============================] - 3s 181us/step - loss: 1.0086 –

…

…

…

Epoch 41/50

14160/14160 [==============================] - 2s 173us/step - loss: 1.0032 - acc: 0.5105

Epoch 42/50

14160/14160 [==============================] - 2s 175us/step - loss: 1.0003 - acc: 0.5109

Epoch 43/50

14160/14160 [==============================] - 3s 179us/step - loss: 1.0018 - acc: 0.5085

Epoch 44/50

14160/14160 [==============================] - 3s 178us/step - loss: 1.0011 - acc: 0.5086

Epoch 45/50

14160/14160 [==============================] - 3s 177us/step - loss: 1.0007 - acc: 0.5084

Epoch 46/50

14160/14160 [==============================] - 3s 178us/step - loss: 0.9996 - acc: 0.5094

Epoch 47/50

14160/14160 [==============================] - 3s 179us/step - loss: 0.9989 - acc: 0.5109

Epoch 48/50

14160/14160 [==============================] - 3s 177us/step - loss: 0.9985 - acc: 0.5107

Epoch 49/50

14160/14160 [==============================] - 2s 176us/step - loss: 0.9976 - acc: 0.5094

Epoch 50/50

14160/14160 [==============================] - 3s 179us/step - loss: 0.9988 - acc: 0.5091

And the evaluation of this model with graph:

> model\_8 %>% evaluate(x\_test, y\_test)

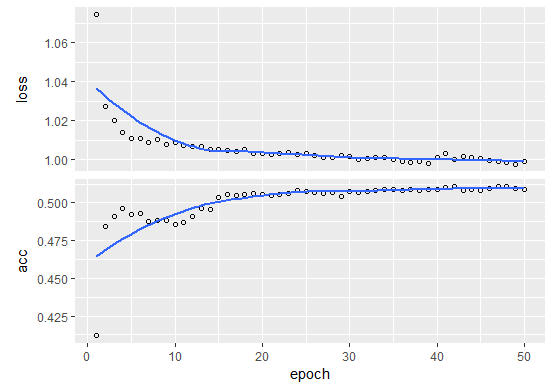
6069/6069 [==============================] - 1s 103us/step

$loss

[1] 1.133988

$acc

[1] 0.3718899



Vary Parameters 9:

Epoch 1/50

14160/14160 [==============================] - 3s 224us/step - loss: 1.3559 - acc: 0.4368

Epoch 2/50

14160/14160 [==============================] - 3s 194us/step - loss: 1.0591 - acc: 0.4708

Epoch 3/50

14160/14160 [==============================] - 3s 204us/step - loss: 1.0512 - acc: 0.4766

Epoch 4/50

14160/14160 [==============================] - 3s 244us/step - loss: 1.0478 - acc: 0.4780

Epoch 5/50

14160/14160 [==============================] - 5s 336us/step - loss: 1.0430 - acc: 0.4800

Epoch 6/50

14160/14160 [==============================] - 3s 221us/step - loss: 1.0397 - acc: 0.4796

Epoch 7/50

14160/14160 [==============================] - 2s 174us/step - loss: 1.0387 - acc: 0.4778

Epoch 8/50

14160/14160 [==============================] - 2s 174us/step - loss: 1.0318 - acc: 0.4831

Epoch 9/50

14160/14160 [==============================] - 2s 172us/step - loss: 1.0314 - acc: 0.4828

Epoch 10/50

14160/14160 [==============================] - 2s 171us/step - loss: 1.0259 - acc: 0.4869

…

…

…

Epoch 40/50

14160/14160 [==============================] - 5s 365us/step - loss: 1.0108 - acc: 0.5056

Epoch 41/50

14160/14160 [==============================] - 5s 368us/step - loss: 1.0102 - acc: 0.5046

Epoch 42/50

14160/14160 [==============================] - 2s 172us/step - loss: 1.0109 - acc: 0.5045

Epoch 43/50

14160/14160 [==============================] - 2s 173us/step - loss: 1.0170 - acc: 0.5010

Epoch 44/50

14160/14160 [==============================] - 2s 168us/step - loss: 1.0100 - acc: 0.5018

Epoch 45/50

14160/14160 [==============================] - 2s 169us/step - loss: 1.0111 - acc: 0.5024

Epoch 46/50

14160/14160 [==============================] - 2s 168us/step - loss: 1.0116 - acc: 0.4992

Epoch 47/50

14160/14160 [==============================] - 2s 170us/step - loss: 1.0120 - acc: 0.5054

Epoch 48/50

14160/14160 [==============================] - 2s 172us/step - loss: 1.0105 - acc: 0.5053

Epoch 49/50

14160/14160 [==============================] - 2s 173us/step - loss: 1.0114 - acc: 0.5072

Epoch 50/50

14160/14160 [==============================] - 2s 170us/step - loss: 1.0082 - acc: 0.5088

And the evaluation of this model with graph:

> model\_9 %>% evaluate(x\_test, y\_test)

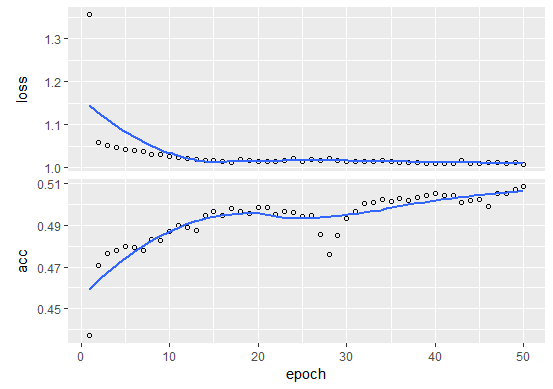
6069/6069 [==============================] - 1s 99us/step

$loss

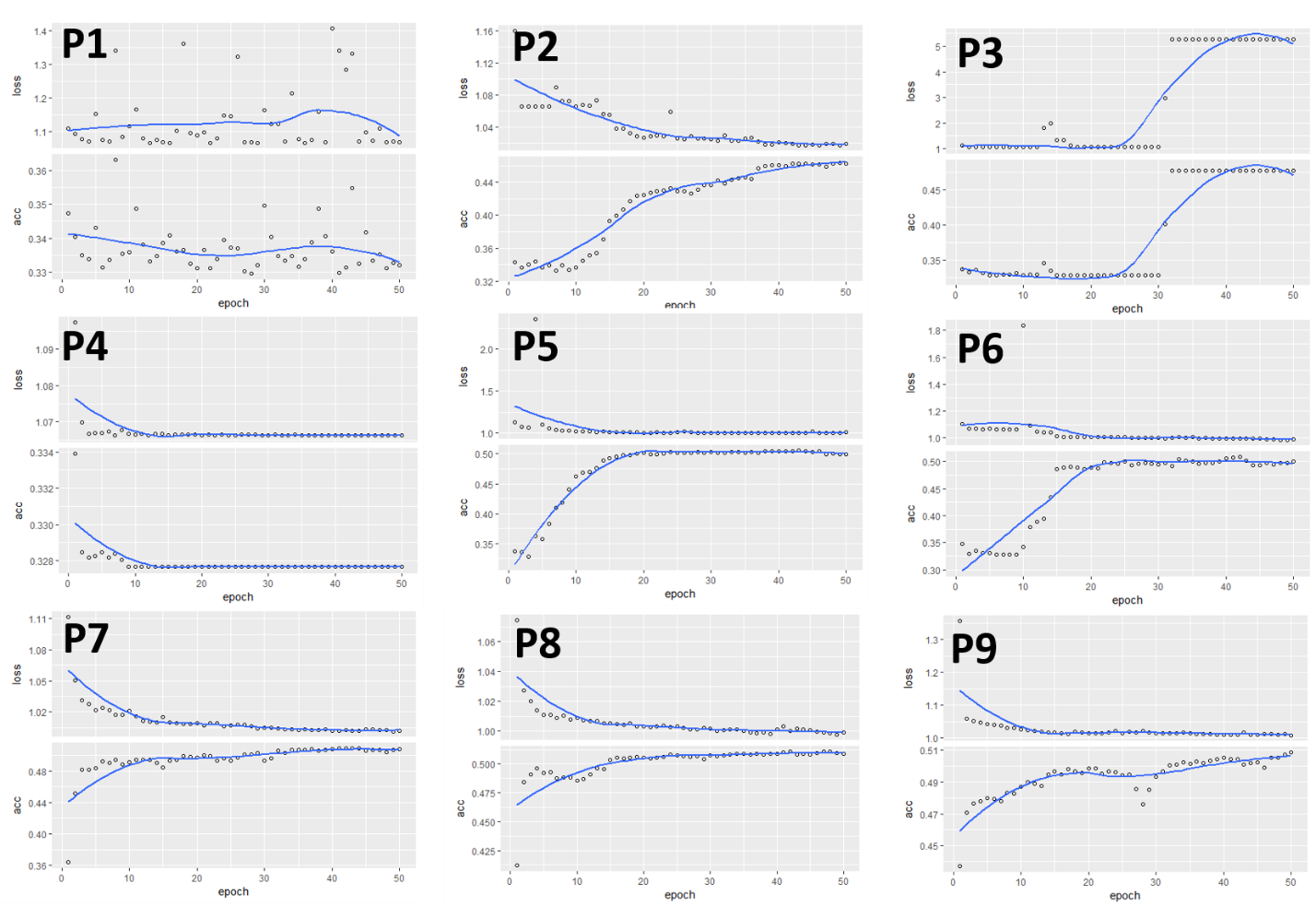
[1] 1.131139

$acc

[1] 0.3826001



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO. | Parameters | Model | Accuracy Training | Accuracy Test |
| 0 | maxlen <- 200  encoding\_dim <- 32  batch\_size <- 10  epochs <- 50 | model %>%  layer\_dense(name = 'e1', units = encoding\_dim, activation = 'relu',input\_shape = 200 ) %>%  layer\_dense(name = 'e2', units = 64, activation = 'relu') %>%  layer\_dense(name = 'e3', units = 128, activation = 'relu') %>%  layer\_dense(name = 'd1', units = 64, activation = 'relu') %>%  layer\_dense(name = 'd2', units = 128, activation = 'relu') %>%  layer\_dense(name = 'd3', units = 3, activation = 'sigmoid') | 0.4922 | 0.391 |
| 1 | maxlen\_1 <- 200  encoding\_dim\_1 <- 32  batch\_size\_1 <- 2  epochs\_1 <- 50 | layer\_dense(name = 'e1', units = encoding\_dim\_1, activation = 'relu',input\_shape = 200 ) %>%  layer\_dense(name = 'e2', units = 64, activation = 'relu') %>%  layer\_dense(name = 'd3', units = 3, activation = 'sigmoid') | 0.3321 | 0.3257538 |
| 2 | maxlen\_2 <- 200  encoding\_dim\_2 <- 64  batch\_size\_2 <- 10  epochs\_2 <- 50 | layer\_dense(name = 'e1', units = encoding\_dim\_1, activation = 'relu',input\_shape = 200 ) %>%  layer\_dense(name = 'e2', units = 128, activation = 'relu') %>%  layer\_dense(name = 'd3', units = 3, activation = 'sigmoid') |  |  |
| 3 | maxlen\_3 <- 200  encoding\_dim\_3 <- 128  batch\_size\_3 <- 10  epochs\_3 <- 50 | layer\_dense(name = 'e1', units = encoding\_dim\_3, activation = 'relu',input\_shape = 200 ) %>%  layer\_dense(name = 'e2', units = 256, activation = 'relu') %>%  layer\_dense(name = 'd3', units = 3, activation = 'sigmoid') | 0.4772 | 0.3578843 |
| 4 | maxlen\_4 <- 200  encoding\_dim\_4 <- 32  batch\_size\_4 <- 10  epochs\_4 <- 50 | layer\_dense(name = 'e1', units = encoding\_dim\_4, activation = 'relu',input\_shape = 200 ) %>%  layer\_dense(name = 'e2', units = 256, activation = 'relu') %>%  layer\_dense(name = 'e3', units = 256, activation = 'relu') %>%  layer\_dense(name = 'e4', units = 256, activation = 'relu') %>%  layer\_dense(name = 'e5', units = 256, activation = 'relu') %>%  layer\_dense(name = 'd3', units = 3, activation = 'sigmoid') | 0.3277 | 0.3237766 |
| 5 | maxlen\_5 <- 200  encoding\_dim\_5 <- 32  batch\_size\_5 <- 10  epochs\_5 <- 50 | layer\_dense(name = 'e1', units = encoding\_dim\_5, activation = 'relu',input\_shape = 200 ) %>%  layer\_dense(name = 'e2', units = 64, activation = 'relu') %>%  layer\_dense(name = 'e3', units = 128, activation = 'relu') %>%  layer\_dense(name = 'e4', units = 128, activation = 'relu') %>%  layer\_dense(name = 'e5', units = 64, activation = 'relu') %>%  layer\_dense(name = 'd3', units = 3, activation = 'sigmoid') | 0.4996 | 0.3926512 |
| 6 | maxlen\_6 <- 200  encoding\_dim\_6 <- 32  batch\_size\_6 <- 10  epochs\_6 <- 50 | layer\_dense(name = 'e1', units = encoding\_dim\_4, activation = 'relu',input\_shape = 200 ) %>%  layer\_dense(name = 'e2', units = 64, activation = 'relu') %>%  layer\_dense(name = 'e3', units = 128, activation = 'relu') %>%  layer\_dense(name = 'e4', units = 256, activation = 'relu') %>%  layer\_dense(name = 'e5', units = 128, activation = 'relu') %>%  layer\_dense(name = 'e6', units = 64, activation = 'relu') %>%  layer\_dense(name = 'd3', units = 3, activation = 'sigmoid') | 0.4999 | 0.3722195 |
| 7 | maxlen\_7 <- 300  encoding\_dim\_7 <- 4  batch\_size\_7 <- 10  epochs\_7 <- 50 | layer\_dense(name = 'e1', units = encoding\_dim\_7, activation = 'relu',input\_shape = 200 ) %>%  layer\_dense(name = 'e2', units = 16, activation = 'relu') %>%  layer\_dense(name = 'e3', units = 32, activation = 'relu') %>%  layer\_dense(name = 'e5', units = 16, activation = 'relu') %>%  layer\_dense(name = 'd3', units = 3, activation = 'sigmoid') | 0.508 | 0.387 |
| 8 | maxlen\_8 <- 300  encoding\_dim\_8 <- 4  batch\_size\_8 <- 10  epochs\_8 <- 50 | layer\_dense(name = 'e1', units = encoding\_dim\_8, activation = 'relu',input\_shape = 200 ) %>%  layer\_dense(name = 'e2', units = 8, activation = 'relu') %>%  layer\_dense(name = 'e3', units = 16, activation = 'relu') %>%  layer\_dense(name = 'e4', units = 8, activation = 'relu') %>%  layer\_dense(name = 'd3', units = 3, activation = 'sigmoid') | 0.509 | 0.372 |
| 9 | maxlen\_9 <- 300  encoding\_dim\_9 <- 4  batch\_size\_9 <- 10  epochs\_9 <- 50 | layer\_dense(name = 'e1', units = encoding\_dim\_9, activation = 'relu',input\_shape = 200 ) %>%  layer\_dense(name = 'e2', units = 8, activation = 'relu') %>%  layer\_dense(name = 'd3', units = 3, activation = 'sigmoid') | 0.509 | 0.383 |

And the best training model is model\_5 with training accuracy 49.99% and test accuracy 39.27% .

Conclusion

In this tutorial, you have learned how to:

(1). Construct neural networks with Keras

(2). Scale data appropriately

(3). Calculate training and test losses

(4). Make predictions using the neural network model