HW0

Yuanxing Cheng, A20453410, CS577-f22

September 20, 2022

Questions

1

Explain the difference between training, validation, and testing data sets. Explain the need for each datasets.

- all three data sets are different
- training data sets are fed to the model first
- then use validation data sets to "train" again and adjust the hyperparameters
- through overfit the validation data sets, we can avoid overfit the test data sets
- test data sets can test the model generalization

2

Explain the 4 steps used in writing a network program using Keras (data, model, learning process, fitting)

---

1. in the data step, we load and vectorize the data, including their labels, then split the data. We also need to obtain the mean and std of the training data sets to normalize all the data, and build a dictionary of the vectorized data to their actual meaning.
2. in the model step, we build a neural network with several layers of units, for now we use densely-connected, but could be convolution or others. For each layer, it comes with predefined number of hidden units, an activation function, and for the first layer, the shape of input.
3. in the learning step, we compile model in the last step with optimizer algorithm, loss function, and the metric to record
4. in the fitting step, we fit the model with training and validation data sets to tune the hyper-parameters

3

Explain the basic parameters used to define a dense layer (number of units, activation)

- number of units is some positive integer and represents the number of affine function used in that layer for approximation
- activation function is another function applied to output value before next layer

4

Explain the network configuration aspects when compiling the model (optimizer, loss, metrics). Explain the difference between loss and metric.

- optimizer stands for the optimizization algorithm used when doing gradient descent
- loss function defines the quantity to minimize through optimizization
- metrics is a measurement of how good your model is, or how "far" your prediction from the true label.
- difference between loss and metric: loss used in train, metric used in test and validation.

5

Explain the 5 basic arguments provided to fit

- input: training data
- output: training labels
- batch size: training data size for each training
- epoch: 1 epoch represents that all training data are used via many batchs
- validation data: used to evaluate the loss and metrics at the end of each epoch

6

Explain the steps used to convert a variable length text string into a binary feature vector

1. get a word list index and convert the words into their indexes
2. encode as list indicate if a word is present or not, e.g., [3,5] to [0,0,1,0,1,0,...,0]

7

Explain possible conclusions when observing training and validation loss graphs over epochs (underfitting and overfitting)

- underfitting means that the training loss is not stable yet after the last epoch
- overfitting means while training loss being stable however validation loss have a "V" shape. Overfit happens when that curve moves up

8

Explain possible hyper-parameters that can be tuned (layers, units per layer, activation functions, loss)

- use more layers usually when the output is more complex. And if cannot improve the result, use the simplest one
- more units will increase accuracy generally.
- what activation function to use depends on the output type. Relu is used often in the hidden layers; for the output layer, we have

  - binary output: use sigmoid
  - multi-class output: use softmax
  - regression problem: N/A
- what loss function to use also depends on the output type.
  - binary output: use binary cross entropy
  - multi-class output: use categorical cross entropy
  - regression problem: mse or mae

9

Explain how a vector of predictions from a binary classifier with a logistic function in the output layer can be converted to class decisions

---

for binary classifier we usually label as class 2 if the probability is less than .5; and as class 1 otherwise.

10

Explain how one-hot-encoding is used to encode class labels of a multi-class classification problem

---

1. suppose total number of label is M, then all the labels are transform first, into a M length zero vector
2. then for class i, its vector element at is changed to 1.

11

Explain the meaning of the output layer when using softmax as an activation function in it

---

softmax will transform output values into a probability distribution and have the property that all values after activation summed up to 1.

12

Explain the difference between sparse-categorical-crossentropy and categorical-crossentropy.

- sparse-categorical-crossentropy is used when the labels for different classes are integers
- categorical-cross is used when using one-hot-encode as in problem 10.
- the result should be the same

13

Assume a dataset with 5 classes where each class is represented equally, what will be the accuracy of a random classifier?

---

for every sample, the random clasifier has a probability of being correct 1/5. The accuracy is the percentage of samples being correctly classified. So the accuracy is binormial distributed random variable. Suppose totally N data, X_i, $i = 1, \ldots, N$ then

$$acc = \sum_i X_i/N, P(acc = k/N) = P(\sum_i X_i = k) = \binom{N}{k} 0.2^k 0.8^{N-k}$$

14

Explain how to normalize feature vectors to have equal mean and standard deviation. Explain the purpose of such normalization

---

- the easiest way is to make all features have zero mean and unit variance, and to do so, we do the following two steps
  1. get column means for training data sets and substruct them from all data column-wisely
  2. get column standard deviation for training data then all data are divided by them column-wisely
- purpose is to make sure the scale of different features are alike and thus the learning spead for each feature are also alike

15

Explain the difference between MSE and MAE metrics. Which is easier to intepret?

- MSE stands for mean square error and MAE stands for mean absolute error; MSE emphasis more on large errors; MAE is unit-preserving.
- MAE is easier to interpret.

16

Explain how to perform k-fold cross validation. When is k-fold cross-validation needed?

---

1. first divide train dataset into $k-1$ part so that with validation dataset we have k block of data in training process
2. train on $k-1$ blocks ot data and validate on the rest 1 block of data
3. repeat k times for diferent blocks of data.

When total number of available data is small, we can do this.

17

Explain when performing k-fold cross validation, how to report the validation error and how to train the final model

---

- the validation error is the average of k times of training on different blocks
- the final model is trained on the whole dataset containing all k blocks of data

Programming

1

data types

- A: a ring of label 1, and label 2 inside
- B: first and third quadrant of label 1, and label 2 otherwise
- C: 2 non-overlapping circle
- D: 2 spiral curve

neural network architecture are represented in a vector: with length of the vector equal to the number of hidden layer and elements being the hidden units in that corresponding layer. For example, 1 hidden layer with 1 neuron is represented as $(1,)$; 2 hidden layers with 1 neuron in the first layer and 2 neurons in the second layer is represented as $(1, 2,)$. The results are summarized in the following table.
Also, all experiments are run under the following condition

- ratio of training to test data: 0.7
- noise: 0
- batch size: 30

Data	features	learning rate	activation	regularization	reg-rate
A	x_1^2, x_2^2	0.3	Tanh	None	N/A
B	$x_1 x_2$	1	Tanh	None	N/A
C	$x_1, x_2, x_1 x_2$	1	Tanh	None	N/A
D	$x_1, x_2, x_1^2, x_2^2, \sin(x_1), \sin(x_2)$	0.03	Tanh	L2	0.001

Data	epochs	network architecture	trainingtraining loss	test loss
A	72	(1,)	0.000	0.000
B	46	(1,)	0.000	0.000
C	22	(1,)	0.000	0.000
D	800	(8,7)	0.014	0.017

2

I will use colab to do all the work. The dataset are load from tensorflow.datasets so no extra link. The classes I pick are the first three.
The training image is vectorized and converted to value between 0 and 1, and their labels are already in integers so no further process on it. The tuning process is summarized in the following tables

- train data size: 15000
- validation data size: 1500
- test data size: 3000
- test acc: 0.7643

NN architecture	activation	optimizer	epoch	batch size
(64,64,3)	(relu,relu,softmax)	rmsprop, lr=0.001	20	512
(64,64,3)	(relu,relu,softmax)	rmsprop, lr=0.001	100	1024
(64,64,3)	(relu,relu,softmax)	rmsprop, lr=0.001	200	1024
(64,64,3)	(relu,relu,softmax)	rmsprop, lr=0.001	75	1024
(64,64,3)	(Tanh,Tanh,softmax)	rmsprop, lr=0.001	75	1024
(64,64,64,64,3)	(relu,...,relu,softmax)	rmsprop, lr=0.001	75	1024
(99,99,99,99,3)	(relu,...,relu,softmax)	rmsprop, lr=0.001	75	1024
($64_{\times 7}$,3)	(relu,...,relu,softmax)	rmsprop, lr=0.001	75	1024

loss	acc	val loss	val acc	comment on next tuning
0.7066	0.7066	0.6712	0.7153	haven't cvg yet and need bigger batch size
0.5245	0.7851	0.5729	0.7767	haven't cvg yet, but batch size enough
0.3873	0.8447	0.5927	0.7787	actually overfit after 75 epoch
0.5873	0.7577	0.5944	0.7547	tune activation
0.5590	0.7662	0.6739	0.7167	not good, tune NN
0.5394	0.7780	0.5991	0.7620	better, try more units then
0.5830	0.7582	0.6156	0.7413	not good, try more layers then
0.5597	0.7648	0.6177	0.7467	no big improvement, so save the 6th model