Mathematical Building Blocks of Neural Network

2021.6.5

Neural network example

```
# load library
from keras.datasets import mnist
from keras import models
from keras import layers
from keras.utils import to_categorical
import numpy as np

# load MNIST dataset in Keras
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# prepare the image data
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255

# network architecture
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))

# compilation step
network.compile(optimizer='rmsprop',loss='categorical_crossentropy',
                metrics=['accuracy'])

# prepare the labels
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

# fit the model
network.fit(train_images, train_labels, epochs=5, batch_size=128)
```

The core building of neural network is the *layer*, data-processing module can be thought as a *filter* for data. Layers extract *representation* out of the data fed into them. Most of **dl** consist of chaining together simple layers that will implement a form of progressive *data distillation*.

Here, out neural network consist of a sequence of two **Dense** layers, which are *densely connected* (fully connected) neural layers. The second layer is 10-way *softmax* layer, which means it will return an array of 10 probability scores.

Compilation step: loss function, optimizer, metrics to monitor during training and testing.

2.2 Data representation for neural networks

2.2.1 Scalars (0D tensors) A tensor that contains only one number is called a *scalar*. Its dimension is 0.

2.2.2 Vectors (1D tensors) An array of number is called a *vector*. A 1D tensor is said to have exactly one axis. Its dimension is 1.

2.2.3 Matrices (2D tensors) An array of vector is called a *matrix*. A 2D tensor is said to have exactly two axes. Can be interpreted as a rectangular grid of numbers.

2.2.5 Key attributes A tensor is defined by three key attributes: *Number of axes*, *shape*, *data type*.

```
# we want to display 4th image in training dataset
n = 4
digit = train_model[n]

import matplotlib.pyplot as plt
plt.imshow(digit, cmap = plt.cm.binary)
plt.show()
```

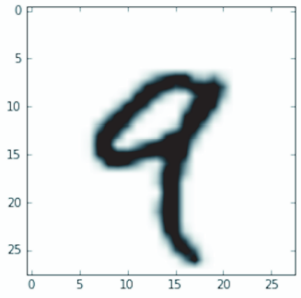

Figure 2.2 The fourth sample in our dataset

2.2.7 The notion of data batches In general, the first axis in all data tensors you will come cross in **dl** will be the *sample axis* (sometimes called *samples dimension*). In addition, **dl** models don't process an entire dataset at once; rather, they break the data into small batches. When considering such a batch tensor, the first axis is called *batch axis* or *batch dimension*.

2.3 Gears of neural network: tensor operations

2.3.6 Geometric interpolation of deep learning In 3D, the following mental image may prove useful. Imagine two sheets of colored paper: one red and one blue. Put one on top of the other. Now crumple them together into a small ball. That crumpled paper ball is your input data, and each sheet of paper is a class of data in a classification problem. What a neural network (or any other machine-learning model) is meant to do is figure out a transformation of the paper ball that would uncrumple it, so as to make the two classes cleanly separable again. With deep learning, this would be implemented as a series of simple transformations of the 3D space, such as those you could apply on the paper ball with your fingers, one movement at a time.

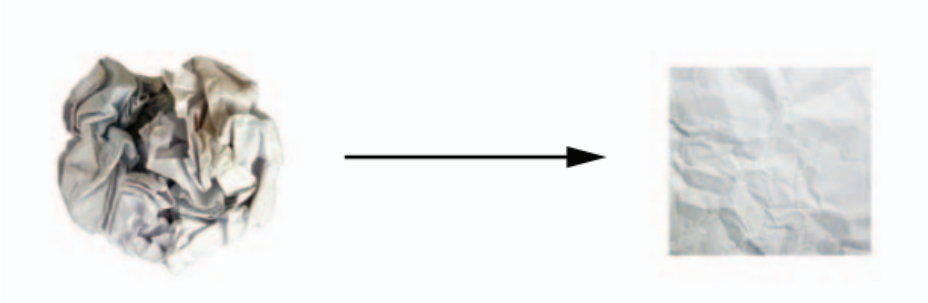

Figure 2.9 Uncrumpling a complicated manifold of data

2.4 Engine of neural network: gradient-based optimization

training loop:

1. Draw a batch of training samples x and corresponding targets y.
2. Run the network on x (a step called the forward pass) to obtain predictions y_pred.

3. Compute the loss of the network on the batch, a measure of the mismatch between y_pred and y.
4. Update all weights of the network in a way that slightly reduces the loss on this batch.

2.4.3 Stochastic gradient desent

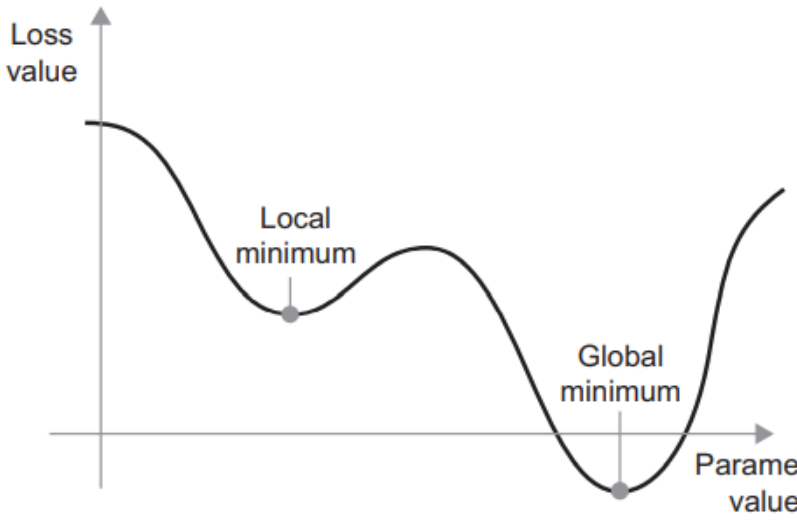

Figure 2.13 A local minimum and a global minimum

There is a local minimum: around that point, moving left would result in the loss increasing, but so would moving right. If the parameter under consideration were being optimized via SGD with a small learning rate, then the optimization process would get stuck at the local minimum instead of making its way to the global minimum.
You can avoid such issues by using momentum. Momentum is implemented by moving the ball at each step based not only on the current slope value but also on the current velocity. In practice, this means updating the parameter w based not only on the current gradient value, but also on the previous parameter update.

2.4.4 Chaining derivatives: Backpropagation algorithm

Summary

⋆ Learning happens by drawing random batches of data samples and their targets, and computing the gradient of the network parameters with respect to the loss on the batch. The network parameters are then moved a bit (the magnitude of the move is defined by the learning rate) in the opposite direction from the gradient.

⋆ The optimizer specifies the exact way in which the gradient of the loss will be used to update parameters: for instance, it could be the RMSProp optimizer, SGD with momentum, and so on.