Training a deep learning model using TensorFlow

Department of Computer Science, City University of Hong Kong

Table of contents

Introduction

Introduction

Usually, we do not retrain the model all the time. Instead, we train and save the model, and load it (in browser) when we need to use it again.

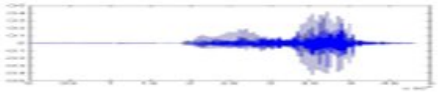
Here, we will introduce how to train the model in local area using TensorFlow (mainly in Python) and use it in browser (website).

Introduction

Machine learning is like training a program to select the best function.

Machine Learning ≈ Looking for a Function

- Speech Recognition

$f($  $)=$ "How are you"

- Image Recognition

$f($  $)=$ "Cat"

- Playing Go

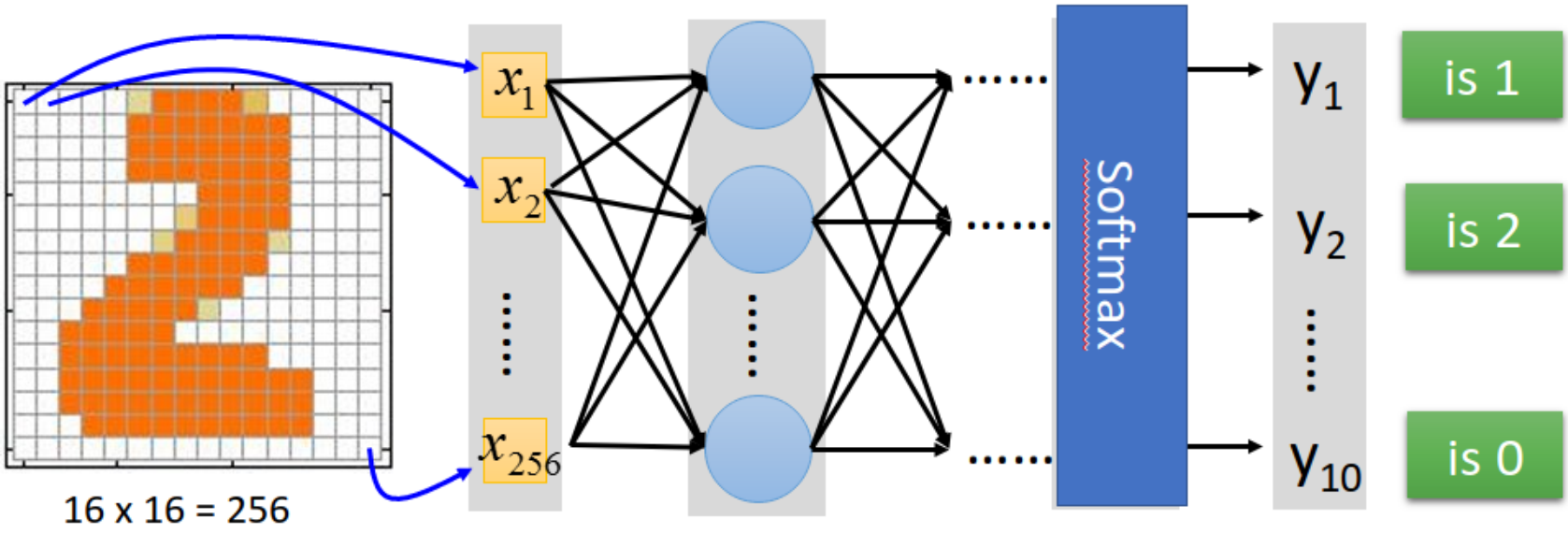
$f($  $)=$ "5-5" (next move)

- Dialogue System

$f($ "Hi" $)=$ "Hello"

(what the user said) (system response)

Learning Target



Ink → 1

No ink → 0

The learning target is ……

Input:  → y_1 has the maximum value

Input:  → y_2 has the maximum value

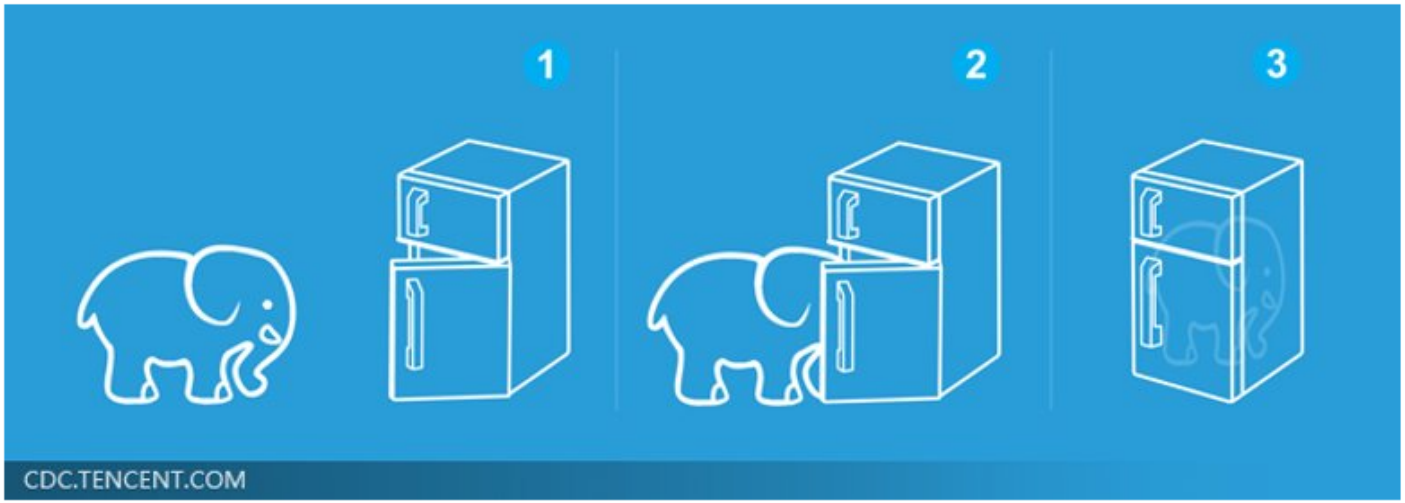
Three Steps for Deep Learning

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ……



Introduction

We will train a Convolutional Neural Network (CNN) model using TensorFlow for hand-written digit classification.

We use the open source dataset provided by MNIST.

Setup

Setup

Environment requirements:

- Python 3 (Anaconda3 is recommended)
- TensorFlow (TensorFlow-GPU if GPU is available)
- TensorFlowjs
- keras

Code

Code

Import necessary packages:

```
import numpy
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.optimizers import Adam
from keras.utils import np_utils
```

Code

Load the dataset and data preprocessing:

```
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Reshaping to format which CNN expects (batch, height, width, channels)
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2], 1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1).astype('float32')
```

Code

Data preprocessing:

```
# normalize inputs from 0-255 to 0-1
X_train/=255
X_test/=255

# one hot encode
number_of_classes = 10
y_train = np_utils.to_categorical(y_train, number_of_classes)
y_test = np_utils.to_categorical(y_test, number_of_classes)
```

Code

Build the model:

```
# create model
model = Sequential()
model.add(Conv2D(32, (5, 5), input_shape=(X_train.shape[1], X_train.shape[2], 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(number_of_classes, activation='softmax'))
```

Code

Train and save the model, evaluate the model using test set:

```
# Compile model
model.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])

# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=200)

# Save the model
model.save('models/mnistCNN.h5')

# Final evaluation of the model
metrics = model.evaluate(X_test, y_test, verbose=0)
print("Metrics(Test loss & Test Accuracy): ")
print(metrics)
```

Code

Convert the model for future use:

```
import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model,r'D:\Jupyter Program\Machine Learning\Mnist')
```

Several supporting files will be created and named as model.json and group1-shard1of1. These files can help us to load our trained CNN model in the website when we need to use it again in the future.

Reference

Reference

https://medium.com/coinmonks/
handwritten-digit-prediction-using-convolutional-neural-networks
https:
//github.com/tankala/ai-examples/blob/master/mnistCNN.py
https://medium.com/@ashok.tankala/
build-your-first-deep-learning-neural-network-model-using-keras

Q&A