Deep learning & applications

Practice#4

Tae Hyun Kim

Tensorflow

- Official tensorflow website
  - https://www.tensorflow.org/
- Both CPU/GPU versions are available
  - https://www.tensorflow.org/install

```
# Current stable release for CPU and GPU
$ pip install tensorflow
```

- Free GPU?
  - Colab
    - **Colab** is a Google research project created to help disseminate machine learni ng education and research. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.
      - For non-commercial (educational) use only.
      - Allow to develop deep learning applications using popular libraries such as **Keras**, **TensorFlow**, **PyTorch,** and **OpenCV**
    - https://colab.research.google.com/notebooks/gpu.ipynb

Tensorflow

- Quick guide
  - https://www.tensorflow.org/tutorials
  - Provides API for Beginners aw well as Experts

For beginners

The best place to start is with the user-friendly Keras sequential API. Build models by plugging together building blocks. After these tutorials, read the Keras guide.

Beginner quickstart

This "Hello, World!" notebook shows the Keras Sequential API and `model.fit`.

Keras basics

This notebook collection demonstrates basic machine learning tasks using Keras.

Load data

These tutorials use `tf.data` to load various data formats and build input pipelines.

For experts

The Keras functional and subclassing APIs provide a define-by-run interface for customization and advanced research. Build your model, then write the forward and backward pass. Create custom layers, activations, and training loops.

Advanced quickstart

This "Hello, World!" notebook uses the Keras subclassing API and a custom training loop.

Customization

This notebook collection shows how to build custom layers and training loops in TensorFlow.

Distributed training

Distribute your model training across multiple GPUs, multiple machines or TPUs.

Tensorflow and Keras (from WiKi)

- **Keras**: open source neural network library written in Python
  - Capable of running on top of Tensorflow, Theano, and so on
  - Enable fast experimentation with DNN
  - Primary author and maintainer is François Chollet (Googler)
    - Author of the XCeption deep neural network model.
  - In 2017, Google's TensorFlow team decided to support Keras in TF
    - Chollet explained that Keras was conceived to be an interface rather than a standalone ML framework
  - Offers a higher-level, more intuitive set of abstractions
    - make it easy to develop deep learning models regardless of the computational backend used

Quick start

- https://www.tensorflow.org/tutorials/quickstart/beginner

Task: *Build and train a net for* binary classification using logistic regression with TF

Input: 2-dim vector, $\boldsymbol{x} = \{x_1, x_2\}$
Output: predicted label of the input, **y** $\in \{0,1\}$

Pseudo code

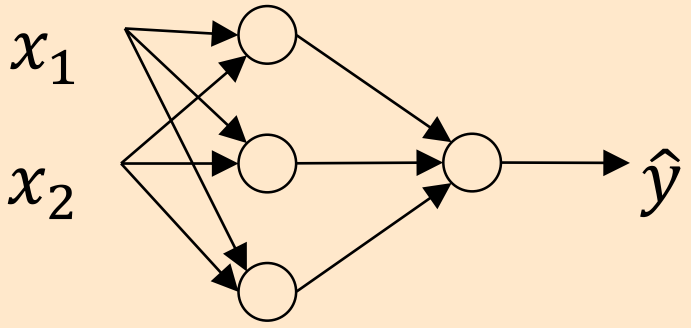
Step 1. Generate 1000(=m) train samples, 100(=n) test samples:

```
x1_train=[], x2_train=[], y_train=[]
for i in range(m):
    x1_train.append(random.unifrom(-2, 2))
    x2_train.append(random.uniform(-2, 2))
    if x1_train[-1]*x1_train[-1] > x2_train[-1]:
        y_train.append(1)
    else:
        y_train.append(0)
x1_test=[], x2_test=[], y_test=[] #generate 'n' test samples!
```

Task: *Build and train a net for* binary classification using logistic regression with TF

Input: 2-dim vector, $\boldsymbol{x} = \{x_1, x_2\}$
Output: predicted label of the input, **y** $\in \{0,1\}$

Pseudo code



Step 2. Build a net with TF and train with 'm' samples for (1000=K) iterations: #K grad updates with ADAM, batch_size = mini batch size = 1000

Step 2-1. calculate the cost with m train samples!
Step 2-2. calculate the cost with n test samples!
Step 2-3. print accuracy with m train samples! (display the number of correctly predicted outputs/m*100)
Step 2-4. print accuracy with n test samples! (display the number of correctly predicted outputs/n*100)

Report

- Submission due: (5/27, 11:59pm)
  - English only
  - Late submission will not be counted
- Submissions: (through blackboard system)
  - 1 source files: allows only .py or .pynb
  - Single page pdf: studentid_name.pdf
    - Include the tables in the next pages

Table 1

- Compare loss functions
  - how to set?
    - https://keras.io/api/losses/
  - Use 'SGD' optimizer

	BinaryCrossentropy	MeanSquaredError
Accuracy (with train set)		
Accuracy (with test set)		

Table 2

- Compare optimizers
  - how to set?
    - https://keras.io/api/optimizers/

	SGD result	RMSProp	Adam
Accuracy (with train set)			
Accuracy (with test set)			
Train time [sec]			
Inference (test) time [sec]			

Table 3

	Your Python Result (in practice#3)	Your best results (CPU version)	Your best results (GPU version)
Accuracy (with train set)			
Accuracy (with test set)			
Train time [sec]			
Inference (test) time [sec]			
Loss type			
Optimizer type			

Table 4 (optional)

- Mini-batch size (gpu ver. only), 1000 epochs!

	Mini-batch = 1	Mini-batch = 32	Mini-batch = 128	Mini-batch = 1000
Accuracy (with train set)				
Accuracy (with test set)				
Train time [sec]				
Inference (test) time [sec]				
Loss type				
Optimizer type				