Deep Learning 4

3007/7059 Artificial Intelligence

School of Computer Science
The University of Adelaide

Cross-Entropy Loss

- Mean squared error (MSE)

$$J = \frac{1}{N}\sum_{i=1}^{N}(yi - h_\theta(x_i))^2$$

- Cross-entropy loss function.

Binary classification: for each class, we get the predicted probability p and $1-p$.

$$L = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -[y_i \cdot log(p_i) + (1-y_i)\cdot log(1-p_i)]$$

y_i : 1 or 0

p_i : the predicted probability of label as 1

Multiclass classification：

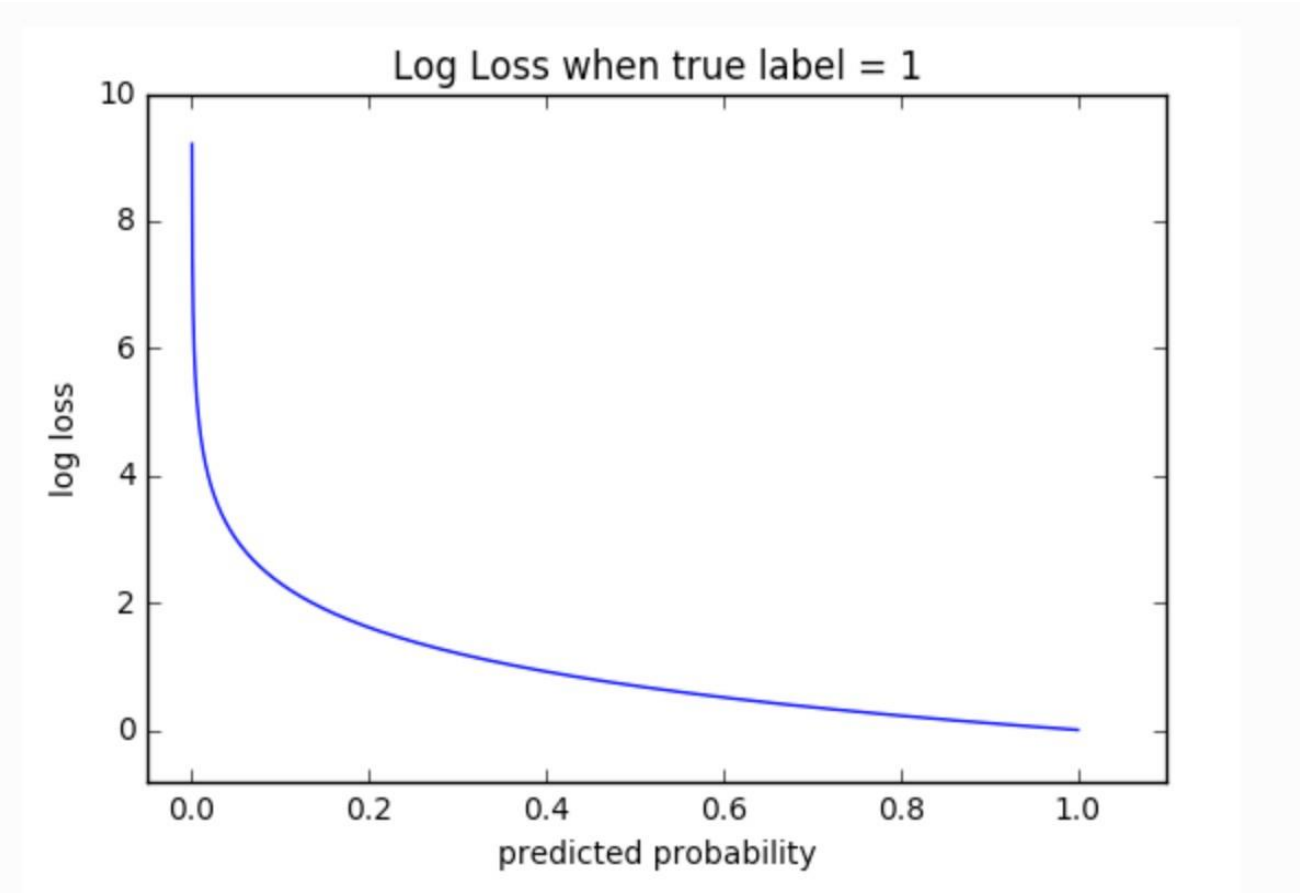
$$L = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -\sum_{c=1}^{M} y_{ic}\log(p_{ic})$$

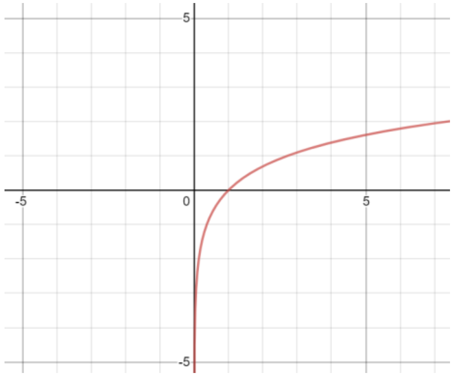
M : the number of classes

y_{ic} : binary indicator (0 or 1) if class label c is the correct classification for observation i

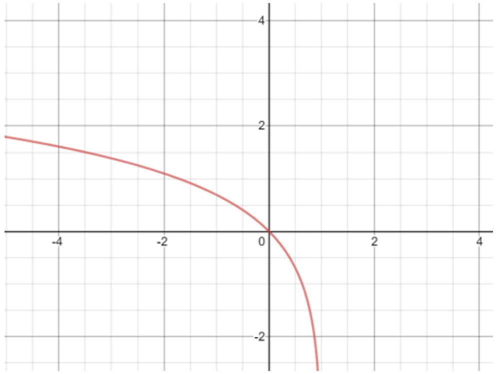
p_{ic}: predicted probability for observation i is of class c

Cross entropy loss when true label is 1





$$y = \log(x)$$

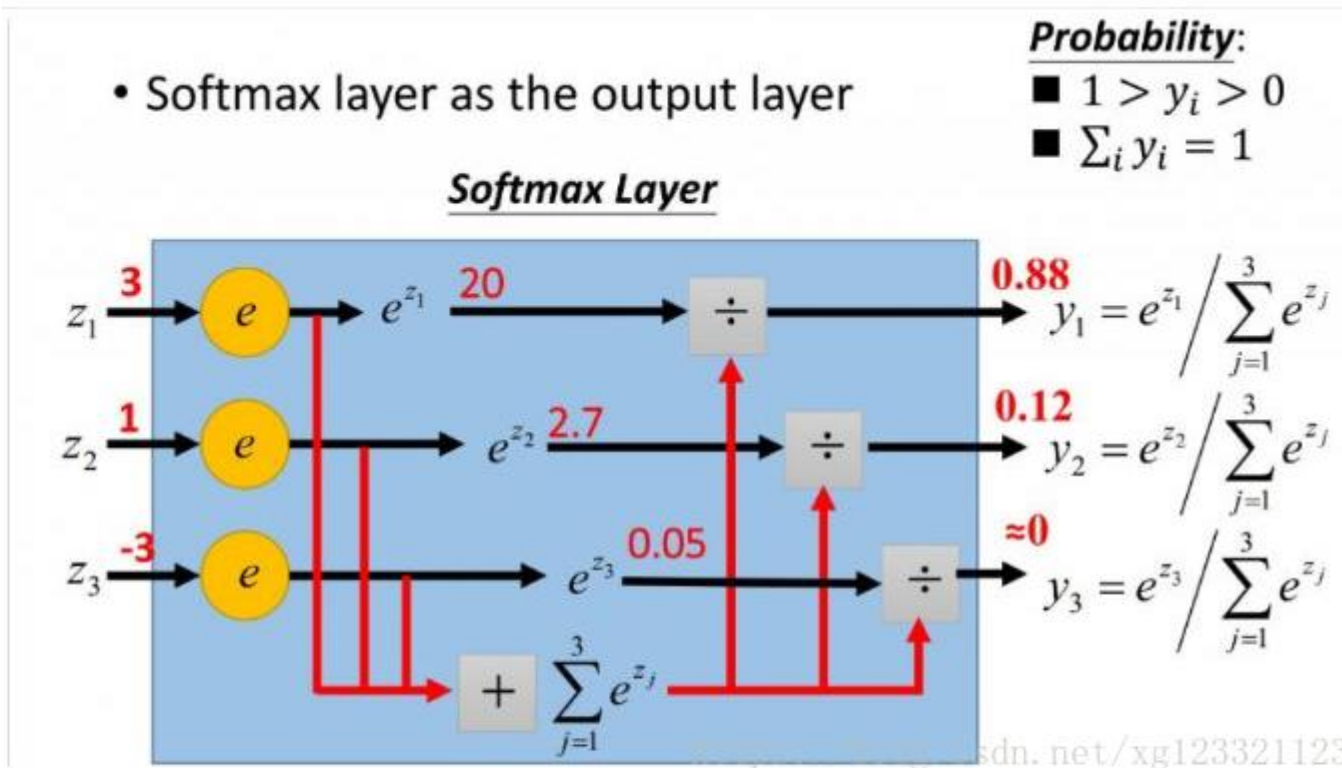


$$L = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -[y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)]$$

$$y = \log(1 - x)$$

Softmax

$$\sigma_i(z) = \frac{\exp(z_i)}{\sum_{j=1}^{m} \exp(z_j)}, \quad i = 1, \ldots, m$$



MNIST LeNet Classifier

- Required packages:

  - Python version 3.5 or later
  - numpy version 1.10 or later: http://www.numpy.org/
  - scipy version 0.16 or later: http://www.scipy.org/
  - matplotlib version 1.4 or later: http://matplotlib.org/
  - pandas version 0.16 or later: http://pandas.pydata.org
  - scikit-learn version 0.15 or later: http://scikit-learn.org
  - keras version 2.0 or later: http://keras.io
  - tensorflow version 1.0 or later: https://www.tensorflow.org
  - ipython/jupyter version 4.0 or later, with notebook support

- Optional packages:
  - pyyaml
  - hdf5 and h5py (required if you use model saving/loading functions in keras)
  - NVIDIA cuDNN if you have NVIDIA GPUs on your machines. https://developer.nvidia.com/rdp/cudnn download

- Anaconda has most of the packages above.

MNIST LeNet Classifier

- ./jupyter notebook
- Test your packages

In [1]:
```python
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import matplotlib
import IPython
import sklearn
import keras
```

```
Using TensorFlow backend.
```

In [2]:
```python
print('numpy:', np.__version__)
print('scipy:', sp.__version__)
print('matplotlib:', matplotlib.__version__)
print('iPython:', IPython.__version__)
print('scikit-learn:', sklearn.__version__)
print('keras: ', keras.__version__)
import tensorflow as tf
print('Tensorflow: ', tf.__version__)
```

```
numpy: 1.13.3
scipy: 1.0.0
matplotlib: 2.1.0
iPython: 6.2.1
scikit-learn: 0.19.1
keras:  2.1.2
Tensorflow:  1.4.0
```

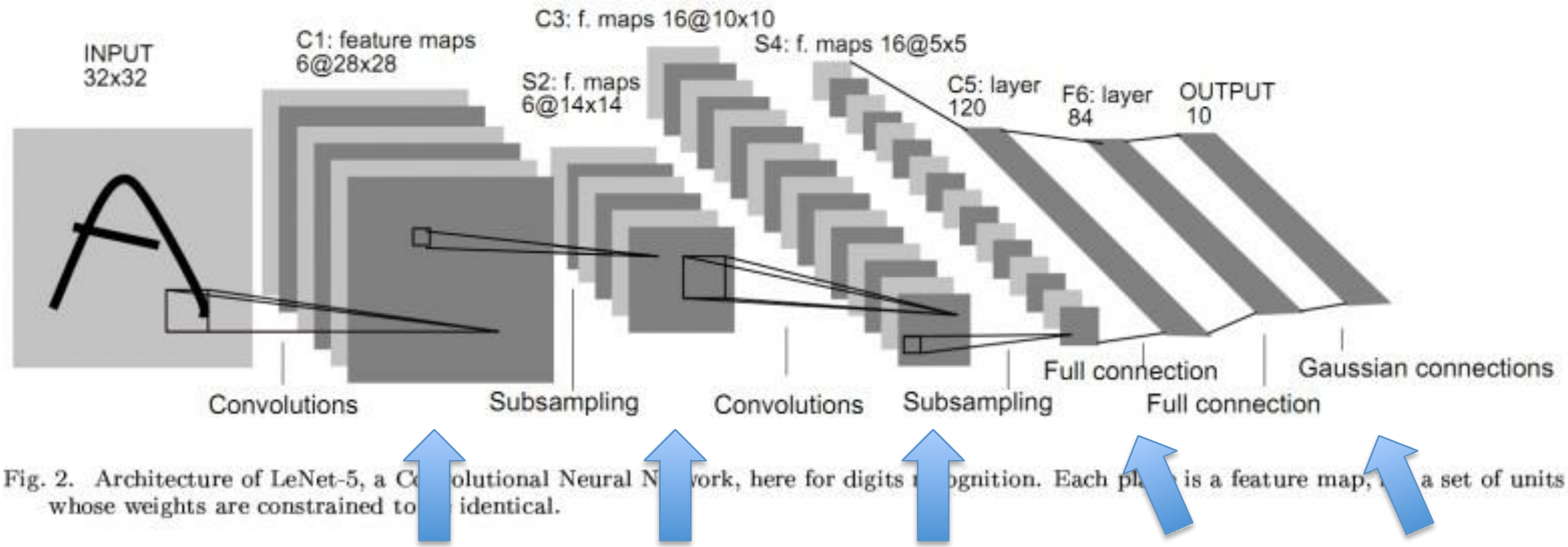
MNIST LeNet Classifier

- Load Keras packages for the CNN layers

```
In [3]: from keras.datasets import mnist
        from keras.models import Sequential
        from keras.layers import Dense, Flatten
        from keras.layers import Conv2D, MaxPooling2D
        from keras import backend as K
```

- mnist has the MNIST dataset
- Sequential model is a linear stack of layers
- Dense, Flatten, Conv2D and MaxPooling2D are the layer types we will use

MNIST LeNet Classifier



Conv2D (6 filters of size 5x5)

MaxPooling2D

Conv2D (16 filters of size 5x5)

MaxPooling2D + Flatten

3 Dense layers

MNIST LeNet Classifier

- Training parameters
  - batch_size: number of images at each step of gradient descent
  - num_classes: number of MNIST classes (10)
  - epochs: number of times the whole training set is used for training
  - img_rows, img_cols: image size

In [4]:
```
# batch size for gradient descent
batch_size = 128
# number of MNIST classes
num_classes = 10
# number of epochs (1 epoch = amount of iterations that covers the whole training set)
epochs = 3 # try a larger number of epochs here (for example 10 or larger)
# input image dimensions
img_rows, img_cols = 28, 28
```

MNIST LeNet Classifier

- Loading the data, and adjusting image dimensions

```
In [5]: # the data, split between train and test sets
        (x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
In [6]: # adjust training image format
        if K.image_data_format() == 'channels_first':
            x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
            x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
            input_shape = (1, img_rows, img_cols)
        else:
            x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
            x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
            input_shape = (img_rows, img_cols, 1)
```

Some versions of keras use data format (samples, channel, rows, columns)
Some versions of keras use data format (samples, rows, columns, channel)

MNIST LeNet Classifier

- Type casting input to be float32
- Normalizing gray values to be in [0,1]
- Verifying training and testing sets

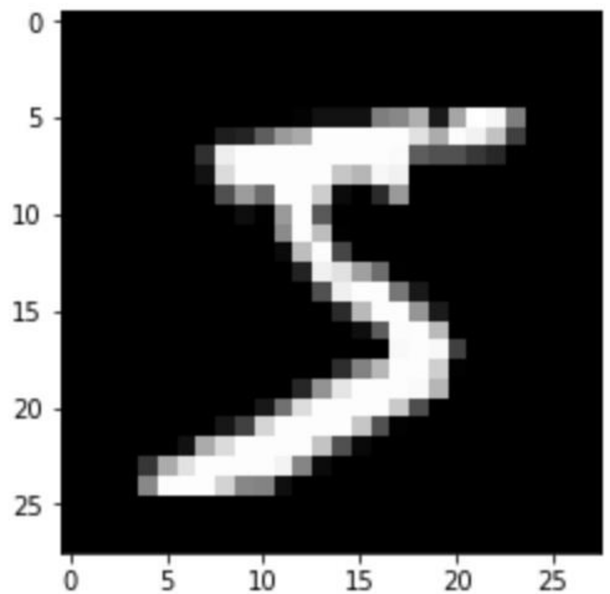
In [7]:
```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

MNIST LeNet Classifier

- Visualizing the dataset

In [8]:
```
for i in range(10):
    first_image = x_train[i,:,:,0]
    first_image = np.array(first_image, dtype='float')
    pixels = first_image.reshape((28, 28))
    plt.imshow(pixels, cmap='gray')
    plt.show()
```



MNIST LeNet Classifier

- Convert labels to one-hot vectors

In [9]:
```
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

- Example
  - 4 -> [0 0 0 0 1 0 0 0 0 0]
  - 9 -> [0 0 0 0 0 0 0 0 0 1]

- Cross-entropy loss function.

MNIST LeNet Classifier

- Create Model

In [10]:
```
model = Sequential()
model.add(Conv2D(6, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

Note: softmax activation

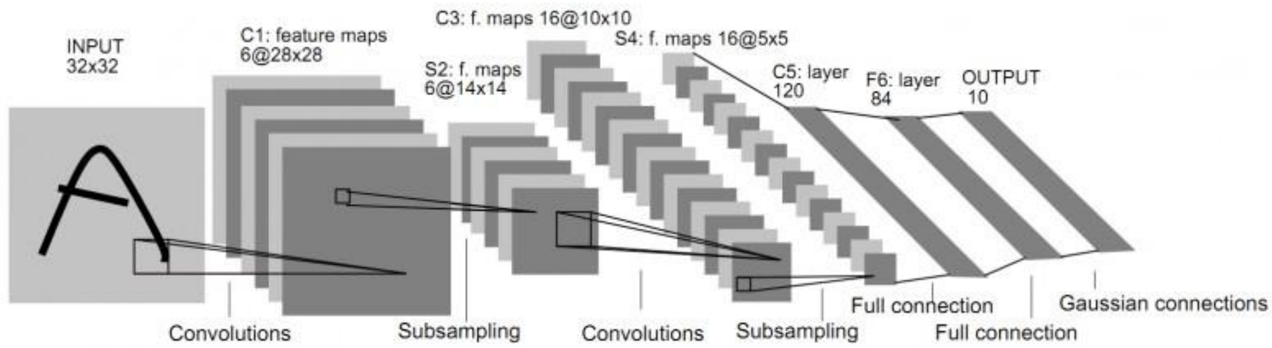


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

MNIST LeNet Classifier

- Configuring the learning process:
  - An optimizer. This could be the string identifier of an existing optimizer (such as rmsprop or adagrad), or an instance of the Optimizer class. See: optimizers.
  - A loss function. This is the objective that the model will try to minimize. It can be the string (loss=‘categorical_crossentropy’) or identifier of an existing loss function. See: losses.
  - A list of metrics. For any classification problem you will want to set this to metrics=['accuracy']. A metric could be the string identifier of an existing metric or a custom metric function.

```
In [11]: model.compile(loss=keras.losses.categorical_crossentropy,
                       optimizer=keras.optimizers.Adadelta(),
                       metrics=['accuracy'])
```

MNIST LeNet Classifier

- Training… finally!

In [12]:
```
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/3
60000/60000 [==============================] - 16s 267us/step - loss: 0.3937 - acc: 0.8734 - val_loss: 0.1172 - val_a
cc: 0.9630
Epoch 2/3
60000/60000 [==============================] - 16s 261us/step - loss: 0.1030 - acc: 0.9677 - val_loss: 0.0770 - val_a
cc: 0.9757
Epoch 3/3
60000/60000 [==============================] - 16s 260us/step - loss: 0.0776 - acc: 0.9759 - val_loss: 0.0737 - val_a
cc: 0.9751
```

MNIST LeNet Classifier

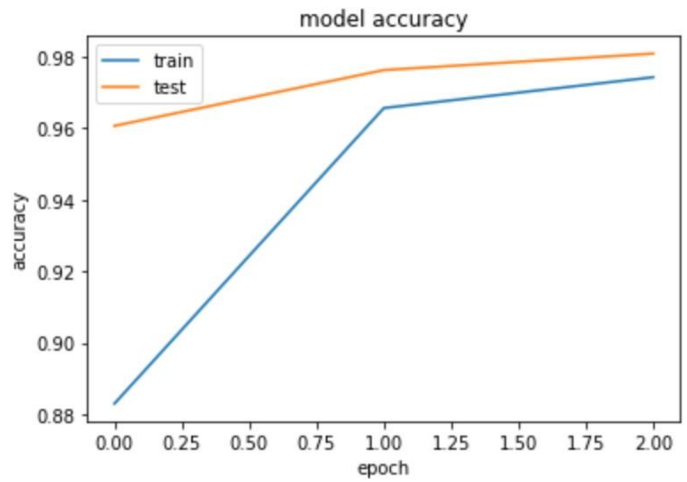
- Running the classifier

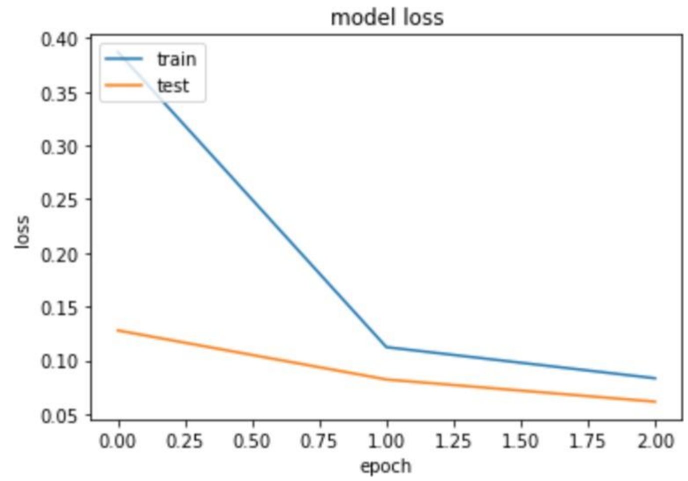
```
In [13]: score = model.evaluate(x_test, y_test, verbose=0)
         print('Test loss:', score[0])
         print('Test accuracy:', score[1])

         Test loss: 0.0736543145942
         Test accuracy: 0.9751
```

MNIST LeNet Classifier

- Plot graphs

```
In [15]: # summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





Google Colab for Deep Learning

- A free cloud service
- Can save notebooks to Google Drive
- Jupyter Notebooks: Tensorflow, Keras, PyTorch
- Free GPU
  - Nvidia T4: 16GB of GPU memory for free
  - "The best available hardware is prioritized for users who use Colaboratory interactively rather than for long-running computations."
- More details here: https://colab.research.google.com/notebooks/welcome.ipynb

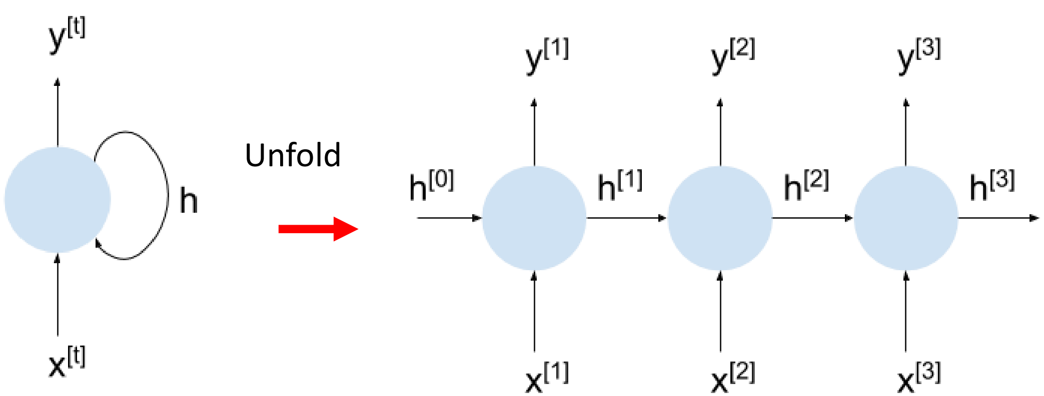
LSTMs

- Let’s look at some LSTM examples – just for fun, this is not going to be in your exam…

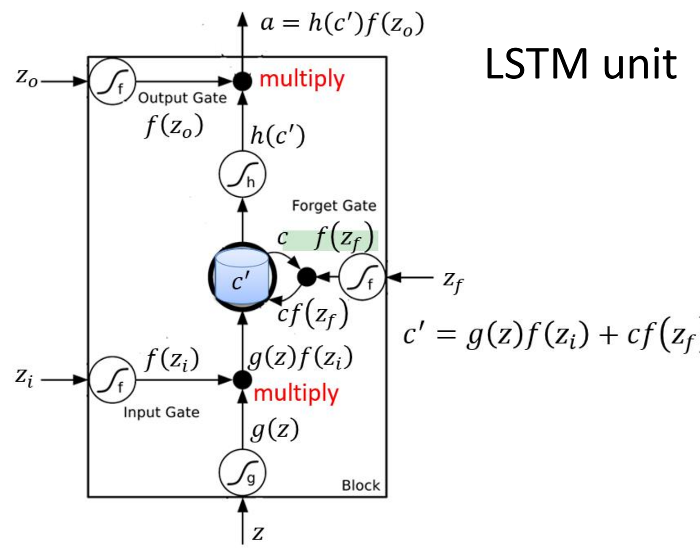
Long short-term memory (LSTM)

- Based on recurrent neural network
- Short-term Memory
- Gating
  - Choose whether update or not
  - Choose whether input or not
  - Reset the state variable
- Modelling sequential data, iterative process

RNN



x is the input vector (at time step *t*), *y* is the output vector and *h* is the *state vector* kept inside the model.



LSTM unit

Classification

- 15. LSTM Classification.ipynb in Google Colab.

Sentimental Analysis using LSTM

- Sentimental Analysis or Opinion Mining

  "Sentiment analysis is a type of data mining that measures the inclination of people's opinions through natural language processing (NLP), computational linguistics and text analysis, which are used to extract and analyze subjective information from the Web - mostly social media and similar sources."

- Can be modelled as a classification problem

https://www.techopedia.com/definition/29695/sentiment-analysis

Sentimental Analysis using LSTM

- Dataset
- Combined Dataset of Tweets, Movie/Book Reviews
- https://www.kaggle.com/arshjat/question2/

Sentimental Analysis using LSTM

- Train.csv

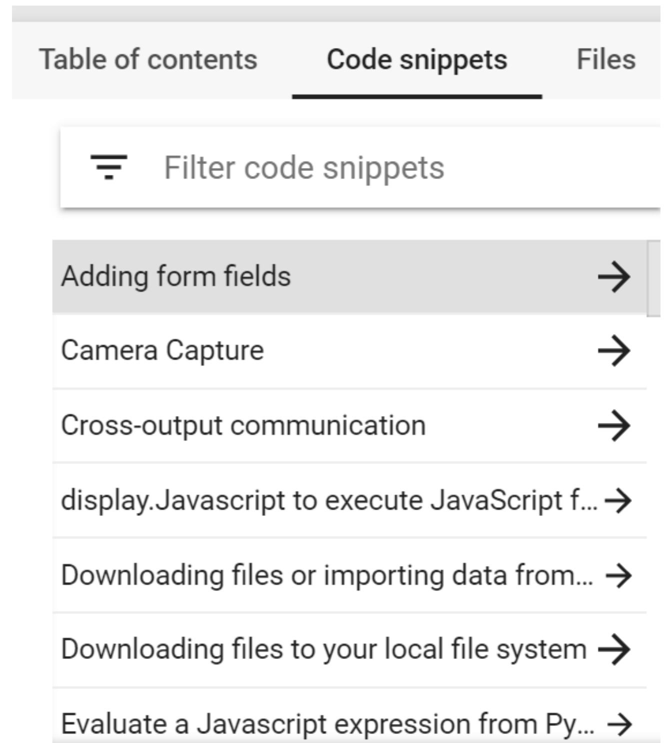
tweet;sentiment
@FrankieTheSats best interview with mcfly ever! dougie was adorable;1
@shaundiviney well rnt u smart shaun. u must of had lots of fun tho;1
i finished my packet of chocolate snakes;0
Last night was a fun adventure. The weather is amazing today! I'm mentally preppin' myself to work out. My legs are sore from dancing;1
MADDIE I LOVE YOUR OLD FASHIONED WAYS;1
http://twitpic.com/3ky85 - Me and James hanging out. .. I love him;1
@SmellTheRainbow awwwwwww what time 2 do have 2 leave??? i don't want u 2 leave it's a poppy time;0
"@gerdaduring LOL;1
I dnt get to go play lasertag w/ my besties!   *old me's dead and gone*;0

- Test.csv

tweet;					
Thanks for	John Carpe	but this lo	and they s	you've alre	and dor
@vintagevandal but! This does give you more time to get your r					
hectic crazy monday-ness. at least its already after 1!;					

Import Dataset to Colab

- Upload to Google Drive, then import to notebook
- Load directly from local machine
- Commit to Github, then clone



Sentimental Analysis using LSTM

- 15. LSTM_twitter_sa.ipynb in Google Colab.