


Lecture 27

Final Exam and Course Review

1. Fisher's Discriminant Analysis
2. Logistic Regression
3. Back Propagation

<http://www.ece.uah.edu/~dwp/pan/course/ee610/slides/Spatial%20Filtering.pdf>
Convolutional Neural Network (deep learning)

Lots of Parameters to Learn!



The diagram shows a vertical flow of layers: imageinp, conv_1, relu_1, maxpool_1, conv_2, relu_2, maxpool_2, fc, softmax, and classoutput. Each layer is connected to the next by a downward arrow.

ANALYSIS RESULT				
	Name	Type	Activations	Learnables
1	imageinput 28x28x1 images with 'zerocenter' normalization	Image Input	28x28x1	-
2	conv_1 32 3x3x1 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	26x26x32	Weights 3x3x1x32 Bias 1x1x32
3	relu_1 ReLU	ReLU	26x26x32	-
4	maxpool_1 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	13x13x32	-
5	conv_2 64 3x3x32 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	11x11x64	Weights 3x3x32x64 Bias 1x1x64
6	relu_2 ReLU	ReLU	11x11x64	-
7	maxpool_2 2x2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	5x5x64	-
8	fc 10 fully connected layer	Fully Connected	1x1x10	Weights 10x1600 Bias 10x1
9	softmax softmax	Softmax	1x1x10	-
10	classoutput crossentropyx with '0' and 9 other classes	Classification Output	-	-

```
>> net = trainNetwork (XTrain, YTrain, layers, options);  
>> analyzeNetwork (net)
```

'cnn_demo.py'

Convolutional Neural Network implementation on Keras.

TensorFlow is an open-sourced end-to-end platform, a library for multiple machine learning tasks, while Keras is a high-level neural network library that runs on top of TensorFlow.

David Pan, UAH

"""

Setup

import numpy as np

from tensorflow import keras

from tensorflow.keras import layers

Prepare the data

Model / data parameters

num_classes = 10

input_shape = (28, 28, 1)

MNIST a dataset of 60,000 28x28 grayscale images of the 10 digits,

along with a test set of 10,000 images.

Load the data and split it between train and test sets

(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

Make sure images have shape (28, 28, 1)

x_train = np.expand_dims(x_train, -1)

x_test = np.expand_dims(x_test, -1)

print("x_train shape:", x_train.shape)

print(x_train.shape[0], "train samples")

print(x_test.shape[0], "test samples")

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)

Build the model

model = keras.Sequential(

[

keras.Input(shape=input_shape),

layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),

layers.MaxPooling2D(pool_size=(2, 2)),

layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),

layers.MaxPooling2D(pool_size=(2, 2)),

layers.Flatten(),

layers.Dense(num_classes, activation="softmax"),

]

)

model.summary()

18496 parameters = 64*(32*3*3+1)

Train the model

batch_size = 128

epochs = 3

model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0.1)

Evaluate the trained model

score = model.evaluate(x_test, y_test, verbose=0)

```
print("Test loss:", score[0])  
print("Test accuracy:", score[1])
```

From <http://www.ece.uah.edu/~dwp/pan/course/ee610/code/Introduction/cnn_demo.py>