CS423: Data mining
Assignment 1: Feature extraction

September 4, 2017

Instruction

In this assignment, you will be using a subset of CIFAR-10 image dataset to train a neural network classifier so that you can use the network to predict an unseen image belonging to one of ten possible classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

Unfortunately, the images you have are still images in RGB format and we have learnt from the class that there are several ways to convert an image into its useful vector representation. The simplest way is simply reshaping the image matrix into vector. A more sophisticated method of course exists. It is your task to implement image feature extraction methods introduced in the class and use them for extracting meaningful features. You can also use others image features that were not discussed in the class. The goal is to learn a neural network classifier with two hidden layers each having 10 nodes for predicting images found in the 'test' folder (not included) as accurate as possible.

- The architecture of the network is fixed to `net = init_network([n,10,10])` where `n` is the dimension of your feature vector.
- Maximum learning epoch for each image is fixed to 1000
- The success of feature extractions is measured by classification accuracy on test dataset (not included). Meanwhile you will be testing your algorithm on hold-out training dataset.

Useful Julia packages

BackpropNeuralNet, Image, ImageView, ImageFeature: `http://juliaimages.github.io/latest/`

What to hand in ?

A zipfile containing

1. A report explaining your feature extraction algorithm in details. (HW1_5xxxxxxxx.pdf)
2. A source code of your feature extraction algorithm. (HW1_5xxxxxxxx.jl)

Email the zipfile to `jakramate.b@cmu.ac.th` using '[CS423]' prefix in your email title.

Hints

Below is the template you might want to start with.

```
# Image classification script
# Homework 1 of CS423
# getting all filenames in current directory, only the files with 'png' extension
images = filter(x->contains(x,".png"),readdir(pwd()))

x = zeros(5000,1024)
y = zeros(length(images),10)

for i=1:1
    # load image
    imm = load(images[i])

    # label extraction
    if contains(images[i],"airplane")
        y[i,1] = 1
    elseif contains(images[i], "automobile")
        y[i,2] = 1
    elseif contains(images[i], "bird")
        y[i,3] = 1
    elseif contains(images[i], "cat")
        y[i,4] = 1
    elseif contains(images[i], "deer")
        y[i,5] = 1
    elseif contains(images[i], "dog")
        y[i,6] = 1
    elseif contains(images[i], "frog")
        y[i,7] = 1
    elseif contains(images[i], "horse")
        y[i,8] = 1
    elseif contains(images[i], "ship")
        y[i,9] = 1
    elseif contains(images[i], "truck")
        y[i,10] = 1
    end
    # feature extraction
    x[i,:] = extractFeature(imm)
end

function extractFeature(image)  # you will write this
    # now simply return vectorised greyscale image
    rimg = map(x->convert(Int64,x),rawview(channelview(Gray.(image))))
    return reshape(rimg, 1, 32^2)
end
```

```
# classification step
using BackpropNeuralNet   # we will use simple neural network

# initialise the network with 1024 input nodes,
# 2 hidden layers with 10 nodes each
# and output layer with 10 nodes to match our 10-class problem
net = init_network([1024,10,10,10])

# ------------------------------
# Training phase
# ------------------------------
epoch = 1000     # maximum learning epoch for each image

for i=1:1            # learning from all of the training images
    for j=1:epoch
        train(net, x[i,:],y[i,:])
    end
end


# ------------------------------
# Testing phase
# ------------------------------

error = 0    # our initial error count is zero
for i=1:1    # loop over the unseen test images
    # the error is simple hamming distance between predicted output vector
    # and the true output vector
    error = error + sum(abs((round(net_eval(net, x[i,:]))-y[i,:])))
end
```