Biostatistics Lab 5 Report

Archana Neupane Timsina

November 24, 2020

Abstract

Here is abstract

1 Introduction

Advancement of neural network is being better and better day by day. Using deep learning computer can recognize the objects by observing the images. More specially, the computer with neural network programming can differentiate the picture of Zebra and Airplane which means computers with deep learning programming can understand the wrold. The idea behind all this process is first of all the given image is divided into the set of pixels. Then each pixel is changed into the set of number according as a given scale of number such as black means 0 and white means like 200. All other colors are catagorize in between 0 and 200. Then by optimization synthesization with convolution neural network, computer can detect the image. Lab 5 is all about the recognization of flower using neural network.

2 Theory

We use the Multilayer Perceptrons(MLP) in deep learning methods to analysis the data using machine learnings.

$$\begin{cases} h_1 = \phi_1(M_1 x + b_1) \\ h_2 = \phi_2(M_2 h1 + b_2) \\ . \\ . \\ . \\ h_{n-1} = \phi_{n-1}(M_{n-1} h_{n-2} + b_{n-1}) \\ y = \phi_n(M_n h_{n-1} + b_n) \end{cases} \tag{1}$$

In this equation $M_1, M_2, ..., M_n$ are matrix, $b_1, b_2 ... b_n$are vectors and all $h_i's$ are hidden layers. . x is training sets and $\phi(.)$ is known as activation function.There are different types of functions are used as activation function for example rectifed linear unit (ReLU), $\phi(x) = (max0, x)$, the hyperbolic tangent, $tanhx$, and the logistic sigmoid,$\phi(x) = (1/1 + e^{-x})$.

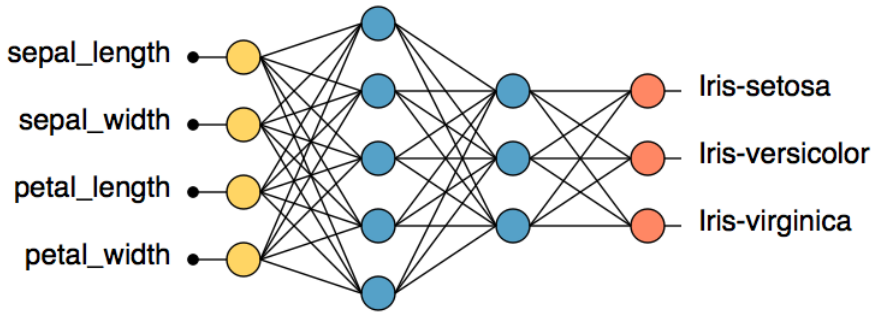

Figure 1: Caption

Once we get data set, This dataset is known as the training set. It should consist of a large number of (x, y) pairs, also known as samples. Each sample specifies an input to the model, and what we want the model's output to be when given that input.Next we need to define a loss function $L(y, \hat{y})$, where y is the actual output from the model and $\hat{y}$ is the target value specified in the training set. This is how we measure whether the model is doing a good job of reproducing the training data. It is then averaged over every sample in the training set:

$$\begin{cases} average \quad loss = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{y}_i) \\ \\ L(y_i, \hat{y}_i) = \sqrt{\sum_i (y_i - \hat{y}_i)}^2 \end{cases} \tag{2}$$

We want to search for the parameter values that minimize the average loss over the training set.There are many ways to do this, but most work in deep learning uses some variant of the gradient descent algorithm. Let θ represent the set of all parameters in the model. Gradient descent involves taking a series of small steps:

$$\theta \leftarrow \theta - \epsilon \frac{\partial L}{\partial \theta}$$

where L is the average loss over the training set. Each step moves a tiny distance in the "downhill" direction. It changes each of the model's parameters by a little bit, with the goal of causing the average loss to decrease.ϵ is called the learning rate, and it determines how much the parameters change on each step. It needs to be chosen very carefully: too small a value will cause learning to be very slow, while too large a value will prevent the algorithm from learning at all. Vallidation is the process where we need to test whether the algorithm works for test set. Overfitting is a major problem for anyone who uses machine learning. One of method to regularize is use as large as number of training set to make the algorithm work perfectly. All of the above discription are theory behind to analysis the given data set.

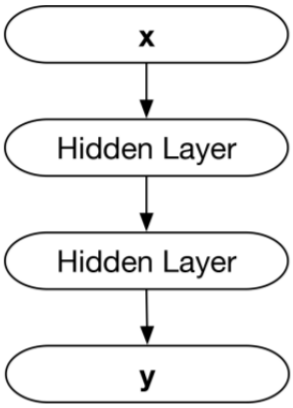

Figure 2: A multilayer perceptron

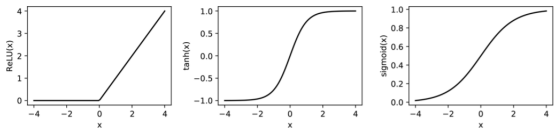

Figure 2-3. Three common activation functions: the rectified linear unit, hyperbolic tangent, and logistic sigmoid.

Figure 3: Activated function

3 Procedures

We use Python colab to analysis the computational part.bring the library of the python from the given URL and We set up the import library and other receipe as follows.

```
!pip install git+https://github.com/williamedwardhahn/mpcr
from mpcr import *

pip install flashtorch
!pip install barbar

from flashtorch.utils import apply_transforms
from flashtorch.saliency import Backprop
import itertools
```

We After setting up all of the data into the my drive and calling that into colab, the following is the selection of training set and test set from data Then we use the following codes to set up the given data into model and get parameter.

```
def train_model(model, num_epochs=25):

    model = model.to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

```
    scheduler = lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)

    for epoch in range(num_epochs):

        print('Epoch: ',epoch+1,'/',num_epochs)

        ###Train
        model.train()
        running_corrects = 0
        for inputs, labels in Bar(dataloaders['train']):
            inputs = inputs.to(device)
            labels = labels.to(device)

            optimizer.zero_grad()

            outputs = model(inputs)

            preds = torch.max(outputs, 1)[1]
            running_corrects += torch.sum(preds == labels.data)

            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

        print("Train ", 'Acc: {:.2f}'.format(running_corrects.double()/dataset_sizes
    ['train']))

        scheduler.step()

        ###Val
        model.eval()
        running_corrects = 0
        for inputs, labels in Bar(dataloaders['valid']):
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            preds = torch.max(outputs, 1)[1]
            running_corrects += torch.sum(preds == labels.data)

        print("Valid ", 'Acc: {:.2f}'.format(running_corrects.double()/dataset_sizes
    ['valid']))
        print("########################")
    return model

    model = models.resnet18(pretrained=True)
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, 102)

def visualize_model(model, num_images=16):
    model.eval()
    index = 0
    for i, (inputs, labels) in enumerate(dataloaders['valid']):
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)

```

```
        preds = torch.max(outputs, 1)[1]

        for j in range(inputs.size()[0]):
            index += 1
            title1 = 'predicted: ' + dataset_labels[int(class_names[preds[j]])-1] +
    '      class: ' + dataset_labels[int(class_names[labels[j]])-1]
            imshow(inputs.cpu().data[j],title1)

            if index == num_images:
                return
```

All other formula are in [2]

4 Analysis

We take an example from images-amazon.com and observe the following result.

```
def train_model(model, num_epochs=25):
image = io.imread('https://images-na.ssl-images-amazon.com/images/I/51dZp-%2B4W9L.
    _AC_.jpg')
plt.imshow(image);
```

Figure 4: Flower recognize by computer

5 Conclusions

We conclude that using neural network deep learning we can recognize flower.

References

[1] Why life science?

[2] Biostatistics Lab 2. https://github.com/aneupanetims2016/Fall-2020/blob/master/Copy$_of_Biostatistics_Lab_5_F$