A
Report
on
Image Analysis and Computer Vision
Spring 2021
(CS 898BA)

Assignment Number: 3

Submitted To
Dr. Ajita Rattani
(Course Instructor)

Submitted By
Suresh Pokharel
WSUID: X254W356
Department of Electrical Engineering and Computer Science
Wichita State University

16th April, 2021

Problem:1

- ❏ Using PASCAL VOC 2012 dataset classify 50 randomly chosen images using VGG-19 model [Follow Dogs versus Atoms example code]. Please report the steps performed and the classification accuracy.

Solution:

The random 50 images of PASCAL VOC 2012 dataset were tested on the pretrained VGG-19 model. The model was loaded with weights pre-trained on an ImageNet dataset of 1000 classes. The default input size for this model is 224x224. So, every test image was preprocessed to resize into (224x224).

Steps:

A. Load the pretrained VGG model and Labels of the image-net dataset.

```
# define the model
model = VGG19(weights='imagenet', include_top=True)

# load the labels from image_net
labels = np.load('labels/imagenet_labels.npy', allow_pickle=True)[()]
```

B. Read all the images with .jpg extension and choose a random sample of size 50.

```
# list all the jpg images from given path
images_all = glob.glob("data/*.jpg")

# select 50 images randomly
images = random.sample(population=images_all, k=50)
```

C. Define a function that takes an image as argument and returns the predicted label from vgg-19 pretrained model.

```
def predict_vgg_19(img):
    # make image ready into array, preprocessing
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)

    # predict from the vgg-19 model
    preds = model.predict(x)

    # find the top 1 class
    idx = (-preds).argsort()[:,:1]

    # split comma and pick first one
    pred_label = str(labels[idx[0][0]].split(',')[0])

    return pred_label
```

Firstly, the image is converted to the array form. Then it is preprocessed to make it compatible with the model. After predicting, only the top-1 class is considered. If there are multiple labels separated by comma(,), it is splitted and only the first value is considered.

D. Define the figure to plot the images. The rows and columns are defined as 10 and 5 respectively.

```
# define a figure of size 12,12
fig = plt.figure(figsize = (12, 24))

# define rows and columns
rows = 10
cols = 5
```

E. Plot all the images in subplots and print the predicted labels.

```
for i in range(1,rows*cols+1):
    # load image from the given path and resize to (224,224)
    img = image.load_img(images[i-1], target_size = (224,224))

    # pass the loaded image and get predicted label
    label = predict_vgg_19(img)
#     print(label)

    # add image to subplot
    ax = fig.add_subplot(rows,cols,i)
    ax.set_title(label)
    plt.imshow(img)

    # disable x,y ticks
    plt.xticks([])
    plt.yticks([])
plt.show()
```

The results for 20 test images with the predicted class are printed as follows:

F. The vgg-19 was trained in 1000 classes. Here, we are just testing the different images on pretrained vgg-19. The label of the voc dataset is inconsistent with the labels of vgg-19. So, we are just counting manually how many labels are predicted correctly in 50 given images.
Total Images: 50
Correctly Classified: 38
Incorrectly Classified: 12
Accuracy = 38/50 = 76%

Problem:2

Explain Gradient Descent Method for weight optimization using equations and figures. How is it different from Stochastic Gradient Descent?

Solution:

Gradient Descent method for weight optimization

- Gradient descent (GD) is an optimization algorithm that's used to optimize the weights while training the machine learning model. It is based on a convex function that tweaks the parameters iteratively to minimize a given function to its local minimum.

Loss function or cost function measures the difference between the actual output and predicted output from the model.

While training a neural network, our goal is to train a model in such a way which makes the weights optimized to get the better prediction.

Let us consider a cost function sum of squared errors (SSE) as,

$$J(w) = \frac{1}{2} \Sigma (\text{target}^{(i)} - \text{output}^{(i)})^2 \quad \text{--- (1)}$$

The magnitude of the weight to be updated in next run is computed by,

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$$

Where, η is the learning rate. The weights are then updated after each epoch as:

$$w_{i+1} = w_i + \Delta w \quad \text{—— (11)}$$

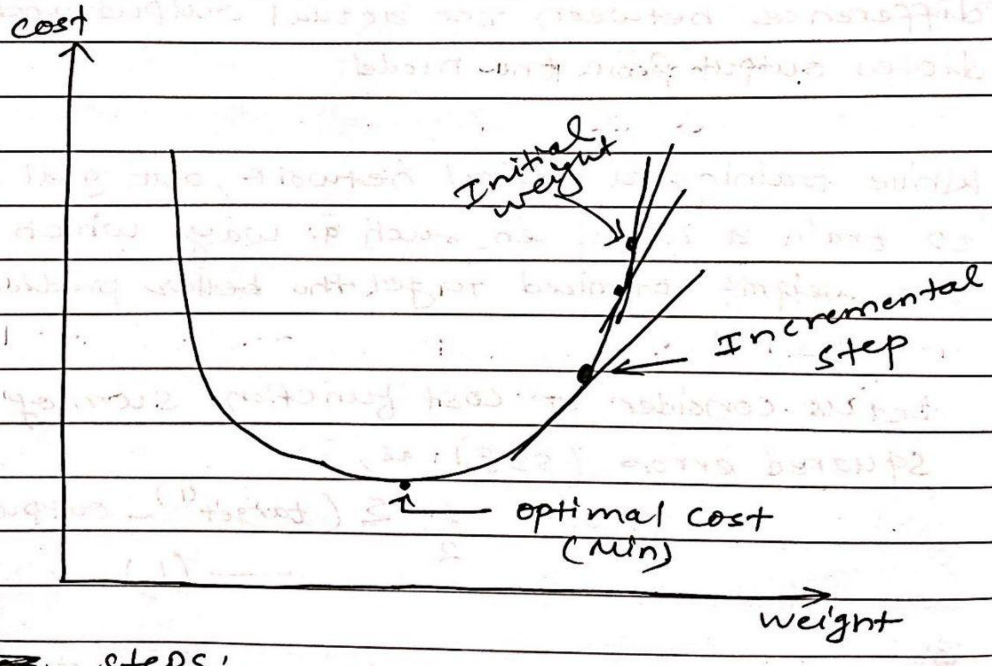
Where, Δw is the change in weight,

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$$

$$= -\eta \sum (\text{target}^{(i)} - \text{output}^{(i)}) (-x_j^{(i)})$$

(from eq. I)

$$= \eta \sum (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)}$$



~~For~~ Steps:

1) Compute the slope that is first order derivative of the function at the current point.

2) Move in opposite direction of the slope increase from the current point by the computed amount.

Stochastic Gradient Descent (SGD)

The gradient descent optimization, the cost gradient is computed based on the complete training test. While working with a large dataset, it takes too long to update the weights that leads to wait longer to converge to global cost minimum.

In stochastic Gradient Descent (SGD), the weights are updated after each training samples as follows.

for each epochs :
    for sample i
        for each weight k

$$w_k = w + \Delta w_k$$

where, $\Delta w_k = \eta \left(target^{(i)} - output^{(i)} \right) x_j$

- The gradient based on a single training sample is a stochastic approximation of 'true' cost gradient.
- SGD is faster than Batch GD.
- SGD can escape shallow local minima more easily.

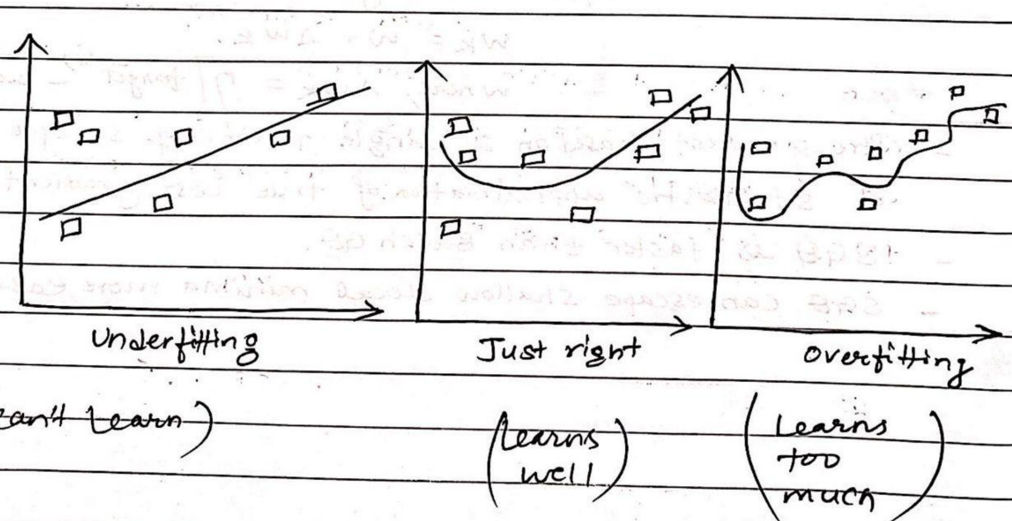
Problem:3

Why is regularization performed in deep neural networks? What are the methods to perform regularization?

Solution:

Regularization

One of the biggest problems that data scientists or machine learning projects face the complexity involved in creating algorithms that perform well on training data as well as new inputs. Regularization is changes made to a learning algorithm to minimize its generalization error without focusing too much on reducing its training error. There are several regularization techniques available, with each working on a different aspect of a learning algorithm or neural network, and each leading to a different outcome.



If a model performs perfectly in the training dataset it cannot be gueranteed to perform as well in new datasets.

Methods to perform the regularization:

a) L2 and L1 Regularization
b) Dropout
c) Data Augmentation
d) Early stopping

L1 and L2 are the most common types of regularization. These update the general cost function by adding another term known as the regularization term.

cost function = Loss (say, binary cross entropy)
+
Regularization term

The values of weight matrices decrease because it assumes that a neural network with smaller weight matrices leads to simpler models.

This regularization term differs in L1 & L2.

In L2, $\text{cost func.} = \text{Loss} + \frac{\lambda}{2m} * \Sigma \|w\|^2$

Here, λ is the regularization parameter. It is the hyperparameter whose value is optimized for better results.

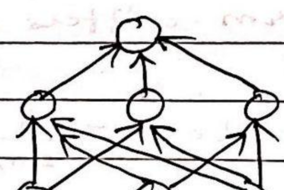
In L1, we have:

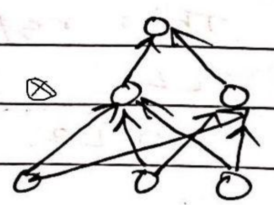
$$\text{Cost function} = \text{Loss} + \frac{\lambda}{2m} * \Sigma \|W\|$$

We penalize the absolute value of the weights. Unlike L2, the weights may be reduced to zero here.
It is very useful when we are trying to compress our model. Otherwise, we usually prefer L2 over it.

Dropout

Dropout is a regularization technique for reducing over fitting in neural networks by preventing complex co-adaptation on training data. It is a very efficient way of performing model averaging with neural networks. 'Dropout' refers to dropping out units in a neural network. Following two figures show the standard neural network and N/N after applying dropout.





Data Augmentation

The simplest way to reduce over-fitting is to increase the size of the training data.

In machine learning, we were not able to increase the size of training data as the labeled data was too costly. Rotation, scaling, shifting, flipping and other transformation are used to perform data augmentation.

Early Stopping

Early stopping is a type of cross validation strategy where we keep one part of the training set as the validation set.

When we see that the performance on the validation set is getting worse, we immediately stop ~~the~~ the training on the model. This process is called early stopping.