SOP : CNN-lytical

Soham Joshi
210051004

April 14, 2022

Contents

1 Motivation for the Project **3**

2 Previous Experience in Coding **3**

3 My Understanding of the Project **3**

4 Summary of Lec-2 **5**
- 4.1 Nearest Neighbour 5
- 4.2 kNN classifier 5
- 4.3 Linear Classifier 5

5 Summary of Lec-3 **6**
- 5.1 Loss function 6
  - 5.1.1 Multiclass SVM Loss 6
  - 5.1.2 Softmax Loss 6
  - 5.1.3 Weight Regularization 6
- 5.2 Optimization 7
  - 5.2.1 Random Search 7
  - 5.2.2 Gradient Descent 7
- 5.3 Features 8

1 Motivation for the Project

Mathematics is something I have always been fascinated by and one of the things I don't see in programming as often as I would like. So, developing fields like Machine Learning, AI and Data Science have always been appealing to me since they are about exploration of things that are not just different permutations of the well known "syntax". This Project will be a very good experience for me to dive into these niche fields that use mathematics and ideas in all their glory. Moreover, these fields sound so mystical that the curious mind always wonders what actually happens in these areas of computer science !
I have done a machine learning course partially, but none of what I did seemed coherent, eventhough I was acing all the questions. The matrix equations I wrote always seemed a bit "forced" and not natural, through this project I would actually like to apply what I will learn and have the ability to write a program containing neural networks just as fluently as I write a program of C++ or python.

2 Previous Experience in Coding

1. Basic Python (class 11,12 as part of IP Curriculum)
2. Stock Market Predictor Project (class 12)
3. Advanced Python (Udemy : Python Zero to Hero by Jose Portilla)
4. GameDev Hackathon by DevCom (Air Hockey Game) - HTML/CSS/JavaScript
5. CodeWars-v1 (3rd place)
6. CS101 - C++ (Sem-1)
7. CS101 Project (Sem-1)

3 My Understanding of the Project

As I understand it the Project is going to be more of a learning experience dealing with the fields of ML, Quantum Computing that are going to use a

lot of numpy in Python and is going to contain a lot of theory as well as applications

4 Summary of Lec-2

This section approaches neural networks through the standpoint of computer vision, and different problems like the semantic gap (the computer just views a grid of numbers) and the variation in illumination, deformation, viewpoint, occlusion, background clutter and interclass variation is discussed.
We will mainly deal with problems where the labels are already known and we evaluate the best category for an image based on a "score".
We then see a data-driven approach where we collect datasets of images and labels and train them to evaluate the classifier on a withheld set of test images
We see the following classifiers :

4.1 Nearest Neighbour

Takes the distance as manhatten metric and evaluates how "different" the image is from the training set images and gives the label same as that of the "nearest image" . This has speed of evaluation inversely proportional to the size of data

4.2 kNN classifier

This takes the k-nearest images using a distance metric and labels the image as the majority of the nearest labels.
This generally doesn't work very well because

1. Terrible Performance at test time
2. Distance Metrics can be very unintuitive

4.3 Linear Classifier

This adopts a parametric approach where we take image column vector and a matrix of weights W, with a constant column vector b and returns a 10 element column vector as scores of the respective categories.
The linear classifier has the functional form :

$$f(x, W, b) = Wx + b \tag{1}$$

where W is a mxn matrix, x is a nx1 column vector and the result is a column vector containing scores of each of the m categories.

Linear classifier can be interpreted as the level set of column vectors which map to zero(scalar) from the image space to the "score" space of a particular object and all the objects belonging to the classifier lie on one side of an n-dimensional plane.
The matrix is nothing but a set of linear classifiers of m objects arranged as row vectors. Generally a set of diverse images is more difficult for a linear classifier to work on, as the "plane" is more and more difficult to find. For this reason, grayscale images would be terrible as there would be variation in very fine details which would be difficult for the classifier to work on.

5 Summary of Lec-3

5.1 Loss function

Lec-3 starts with the discussion of loss function, in order to judge how good or bad a particular result of scores is given a data set and the corresponding labels. The main discussion is related to SVM and Softmax Losses in this module.

5.1.1 Multiclass SVM Loss

Given an example (x_i, y_i) where x_i is the image and y_i is the (integer) label and using shorthand for the scores vector $s_i = f(x_i, W)$

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1) \tag{2}$$

5.1.2 Softmax Loss

Given an example (x_i, y_i) where x_i is the image and y_i is the (integer) label and using shorthand for the scores vector $s_i = f(x_i, W)$

$$L_i = -log(\frac{e^{s_k}}{\sum_j e^{s_j}}) \tag{3}$$

5.1.3 Weight Regularization

The motivation behind weight regularization is that a rather large subset of matrices may give similar losses but not all of them should be treated

equally as they may not take into account all of the properties of the object we are classifying (for instance if a data set has many red cars matrices only checking if an object is red give the correct answers in the training set but may fail test cases)
Hence, we need another additional function to ensure that the function we are implementing as a linear map actually works for general cases as well.
Hence, the complete loss function over all elements of the training set is given by :

$$L = \frac{\sum_i L_i}{n} + \lambda R(W) \tag{4}$$

The regularization is most commonly :

$$R(W) = \sum_k \sum_l W_{k,l}^2 \tag{5}$$

This is known as the L2 regularization and prefers "distributed" values of elements. Note that this does harm the loss function, but there will always be a trade off between precision and generality.

5.2 Optimization

Now our task is to find the matrix W such that the loss function is minimized.

5.2.1 Random Search

Ok, this is a rather bad technique but the principle is that we can generate random values and keep track of which values of W work the best.
This gives approximately 15.5% accuracy, while state of the art methods give around 95% accuracy. So clearly we have a long way to go!

5.2.2 Gradient Descent

This is a rather clever technique. Basically we know the gradient of the loss function at a particular point (i.e for a particular value of W for an image whose label is known) as it will just be a function of W, so we can find the gradient by two ways:

1. Numerically : This is a slow and tedious process, moreover we cannot cover every direction

2. Analytically : This works using calculus, but is error prone, has a hyperparameter of learning rate

Thus, Analytical gradient with the occasional gradient check is the way to go in this case. So our task boils down to calculating gradient of loss function $L = f(W)$, updating W accordingly until the loss function is minimized.
Sidenote : this may run into the pitfall of multiple maxima/minima, there might be a way to handle that in subsequent lectures
Gradient descent works even better in smaller batch sizes as it is more feasible to cover large steps on small data than small steps on a large data. Small batch sizes give a good idea of the optimal W matrix

5.3 Features

This was used in older times where Optimization and loss were applied to a feature vector extracted from the image containing some property like :

1. Color hues (histogram)
2. Figure Edges
3. Bag of Words
4. Pixel intensities

In the modern setting, feature vector is not used, these techniques are applied directly on the image itself