PHASE 1 REPORT

EEE 485 TERM PROJECT

Ahmad Zafar Khan
21403048

Hassan Ahmed
21500439

Problem and Dataset Description

The problem we are trying to address is trying to identify and classify fruits from a set of images. This problem, while specifically related to fruits for our particular case can be remodeled to classify any object given the right set of images. This makes it have a wide range of applications and can be a more versatile option to use for object detection and recognition in images compared to traditional computer vision methods that are more limited if variety is introduced in the data. This makes it a powerful replacement especially considering the decreasing cost of computational resources making it more feasible to run time consuming ML code

The dataset we plan to use is a collection of images of fruits taken from different rotated angles. The total size of the dataset is 65429 images. It is divided into a training and test set of 48905 images and 16421 images respectively. There are 95 different fruits which means the classification algorithm we implement will have 95 classes to choose from. There is also a smaller dataset of 103 images with multiple fruits in the same image. This can be useful to test our algorithm on more complex training sets to see how they would perform in situations more similar to the real world. The dataset also does not combine different varieties of fruit so apples are separated into five different classes corresponding to five varieties.

The images are sized as 100 x 100 pixels. While it is possible to use individual pixel values converted from RGB to HSV, this will cause the problem to be extremely complex due to the high number of parameters and hence makes it infeasible to use without some preprocessing. We take advantage of computer vision libraries and existing methods to extract features from the image and then use those as inputs for our ML algorithms. Some options available to us are the Histogram of Oriented Gradients (HOG) or a more localized feature detection algorithm such as SURF or SIFT. SURF has support for it built into MATLAB so it is easier to use but the processing time for these algorithms also need to be taken into account. This makes HOG the best option available for us to use.

Review of Methods

1. Neural Network

We plan to train a neural network on a dataset. A neural network is a collection of perceptrons that work together similar to neurons firing in our brain. While, they can be physical too, they are mostly implemented as algorithms in Machine Learning examples. Neural Networks and especially their subclass of convolutional neural networks (CNN) are commonly used in image analysis and object detection.

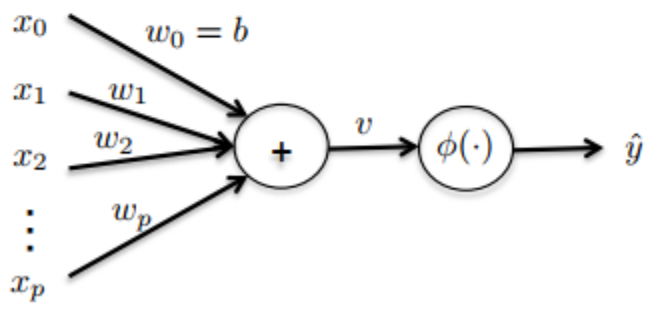

Fig. 1 Structure of a perceptron

The perceptron works by adding a series of weighted sum of features in addition to a commonly added bias term. If the sum satisfies a condition, it activates a specific output. Perceptrons may be connected in a wide configuration of series or parallel specifications depending on the requirements of the model.

In mathematical terms, this translates to:

$$v = \sum_{j=1}^{p} w_j x_j + b$$

Where: $x_0 = 1$

There is an activation function that is used to decide whether the inputs will cause a positive or negative output. This will depend on our problem. We will try to use simple activation functions such as the ReLu to simplify the complexity of the problem. This is necessary to make it computationally feasible to solve depending on the machine capabilities but there are a few challenges that we will need to account for with this approach. Due to the unbounded nature of this activation function, it can blow the activation which might force us to use a sigmoid function such as $tanh(x)$ instead.

The ReLu function can be described by the statement:

$$A(x) = \max(0, x)$$

Stacking perceptrons leads to a neural network. A typical topography of a neural network is shown below.

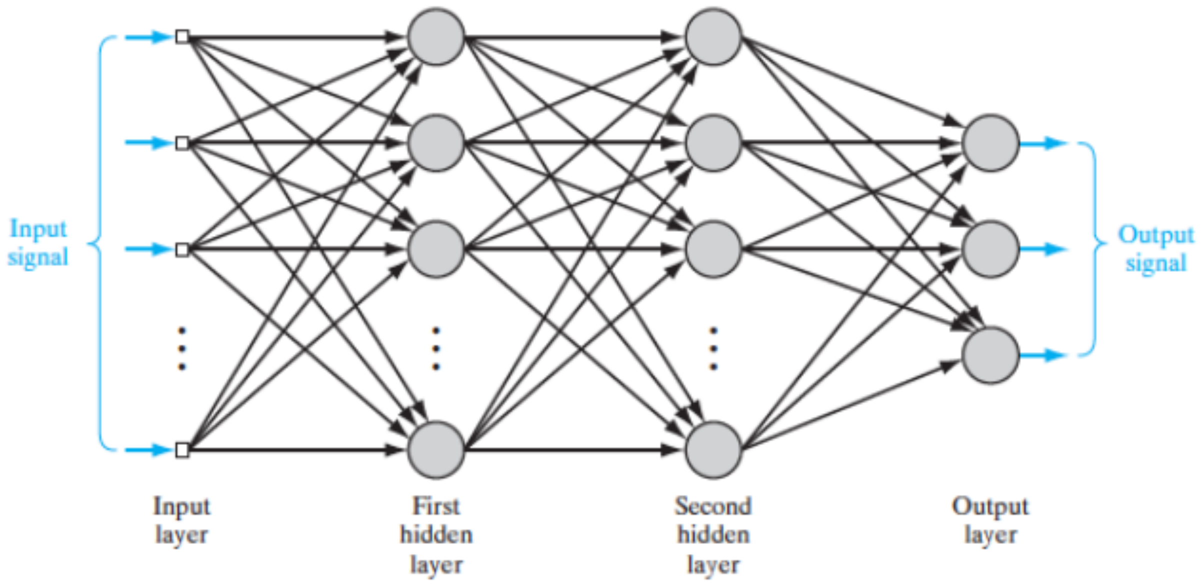

Fig. 2 Typical neural network structure

There are a couple of challenges that are come with the territory of designing a neural network. These are choosing the number of layers and the type of layer such as a convolution or a pooling layer. This will be particularly necessary to reduce the complexity of the problem. Max pooling layers decrease the complexity and hence the computational resource necessary by decreasing the amount of parameters. This has the added advantage of tackling another challenge inherent to neural networks which is of overfitting. When implementing a Neural Networks have a tendency of overfitting data which give really high training accuracy but a bad test accuracy. By reducing the parameters, we can decrease this tendency to overfit and hence give more accurate results.

To measure the accuracy of this model. We will build a confusion matrix and use that to find the precision and recall. We will also calculate the accuracy through the cross-entropy function given below:

$$cross-entropy: -\sum_{k=1}^{d^L} y_{ik}\log(x_k^L)$$

Where $k = 95$ for the number of classes in the dataset and L is the output layer of our Neural Network.

2. Naïve Bayes Classifier

The Naïve Bayes classifier is a simple yet powerful classifier that is commonly used. It is a fast classifier with a computational complexity of order O(nf). It is based on Bayesian Probability theory. Specifically in this case, the algorithm for Naive Bayes is based on Bayes' rule as given below:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

The object of the classifier is to learn the conditional probability of "y" given an observed "X". This is done by the finding the MAP estimate of "y" given the the observation, weighted by the prior probability of the fruits in the dataset. This prior probability labelling a relatively large dataset from the training set [1].

This classifier can be multivariate, i.e. we can classify, beyond the binary case, many classes which in our project are the 95 different fruits. The operating principle for the classifier is that all the features that make up the vector "X" are independent. Hence, the reason why this classifier is called "Naïve." As such, it is a generalization of the Maximum a Posteriori estimation.

Depending on the type of data, the Naïve Bayes Classifier is commonly chosen to be either multinomial, bernoulli or gaussian. Given that our predictors will be made of image data taking on a large range values we will use the gaussian classifier. The log-likelihood function of this distribution will serve as our loss function. The distribution function is given below:

$$P(Y = y_k|X_1, X_2, \ldots, X_n) = \frac{1}{\sqrt{2\pi\sigma_x^2}}exp(\frac{-(y_k - \mu_x)^2}{2\sigma_x^2})$$

To measure the accuracy of this classifier we will construct a confusion matrix and display the values for precision, recall and accuracy as our performance measures.

Some of the challenges we expect to face with this algorithm are inherent to the Naive Bayes algorithm such as the assumption that the features are independent. This is rarely the case in real life.

3. Logistic Regression

Logistic Regression, despite its name, is a technique for classification as the outcome variable "Y" is discrete. As the output variable can take on one of 95 different values, we will use Multinomial logistic regression. In Naive Bayes, we were classifying based on the conditional probability of Y given X, whereas in logistic regression we directly learn this probability. We assume that this probability takes on a functional form that is a sigmoid function.

$$g(z) = \frac{1}{1+e^{-z}}$$

Given that there are two sets of 100 pixels our sigmoid function will exist in two dimensions as follows.

$$P(Y|X) = \frac{1}{1+exp(w_o + \sum_{i=1}^{n} w_i Xi)}$$

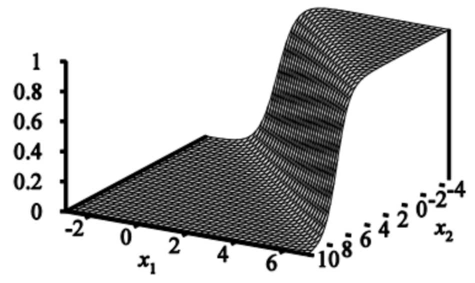

Fig. 3 Two dimensional sigmoid function

In order to do to multinomial logistic regression however, we have to make the important decision of choosing the one-vs-all classifying technique. With this technique we can isolate each class of fruits and learn its probability against all others as a binary classifier. Then repeating the process for each of the 95 classes. So we will have 95 distinct binary classifiers that will be implemented inside a for loop.

In the training stage, we will need 95 sets of model parameters with the features we get from our MATLAB computer vision library. To train the model better, we will need to minimize the cost function below:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}log(h_\theta(x^{(i)})) - (1-y^{(i)})log(1-h_\theta(x^{(i)}))]$$

This function comes from the cross entropy function. In order to optimize our algorithm, we will have to maximize this function. As this function is convex, no closed form expression exists. Therefore, we will have to use mini-batch gradient to update our weights.

$$w(n+1) \leftarrow w(n) - \frac{\alpha}{r} \cdot \frac{\partial \mathrm{y}}{\partial \mathrm{w}}$$

$$b(n+1) \leftarrow b(n) - \frac{\alpha}{r} \cdot \frac{\partial \mathrm{y}}{\partial \mathrm{b}}$$

Where α is our learning rate, which we will fine tune by trial and error, and b is what ever the bias is, found by gradient descent.

LR is quick to run and easily extended to the multinomial case however, this may make the algorithm more computationally exhaustive.

To check the performance of the algorithm, we will construct the confusion matrix of this algorithm as with the NB algorithm.

References

[1] D. Goswami, S. Kalkan and N. Kru¨ger, *Kovan.ceng.metu.edu.tr*, 2019. [Online]. Available: http://www.kovan.ceng.metu.edu.tr/~sinan/publications/dibyenduSCIA.pdf. [Accessed: 20- Mar- 2019].