Homework 6
BSysE 530
Classification and Motion

Abhijay Ghildyal

EECS Washington State University
abhijay.ghildyal@wsu.edu

10th April 2019

Contents

1 Introduction

In this assignment we shall be learning about two different concepts Classification and Motion.

1.1 Machine Learning an overview

Machine Learning has been found to be useful by many developers for training or automating tasks which otherwise would not work manually. These methods are applicable in a diverse range of industrial and research applications. The diverse array of ML algorithms find an optimal solution by training on several candidates which are judged for optimality on the basis of a performance metric. Many of the ML algorithms learn a function through approximation. These algorithms also require a bunch of parameters which need to be tuned so as to act optimally as per the performance metric.

There are several questions which arise regarding the volume of data to be used, robustness to errors etc. which can be answered by statistical decision theory and computational theory. Being at the crossroads of many different disciplines ML has led to the intermingling of concepts from different fields of study trying to tap potential synergies of diverse experiments that improve with experience.

With large amounts of data, the applications that are being built today can be personalized. A number of these customized (even for an individual) applications because of their availability in large amounts can be used to provide services using ML which poses a huge advantage.

Most widely used methods for solving ML problems are supervised learning, unsupervised learning and reinforcement learning. In the supervised learning paradigm, the predictions are made using a learned mapping function. One area on which supervised learning has made a high impact is Deep Learning. These systems use gradient-based optimization algorithms to tune parameters and achieve optimized results. Another progress in the area of parallel computing architectures where these large networks can perform in lesser time or increased in size. Computer vision and speech recognition systems have majorly improved due to these reasons. Unsupervised learning takes advantage of the underlying patterns and structures in data to give observations, recommendations or results which otherwise could not be found. Another paradigm that is Reinforcement Learning is closely related to control-theory literature where the objective is to maximize expected rewards over time and the learning task is to find an optimal policy or behavior. Semi-supervised learning uses unlabeled data and labels some of the data for supervised learning. Using this, predictions are made on un-predicted data which is then added to the base learner again for re-training. Active learning takes feedback from the trainer for labeling which it uses for its iterative learning. Causal modeling is another paradigm which exploits information for causality and hence influences.

LEARNING = REPRESENTATION + EVALUATION + OPTIMIZATION

There are three components to learning algorithms which are representation, evaluation and optimization.

1. 'Representation' is the set of classifiers which can be written in a formal language for computers to understand

2. 'Evaluation' methods a.k.a objective functions and scoring functions define the performance of the classifier as it learns on the data. They depend on the classifier chosen and can be internal as well as external

3. 'Optimization' strategies applied to classifiers are vital for efficient learning and choice of the technique used is often crucial for obtaining optimum learning results

1.2 Classification

Classification is the technique of predicting the class or label or category of given data points (can be thought of as rows in tabular data). In terms of predictive modeling it is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

In more formal terms let's say that the learning goal is to find hypothesis $h \epsilon H$ that closely approximates a target concept C, where h is the output classifier and the target concept may not be in H. Here classification would be $h(x) \epsilon C$ and regression is $h(x) \epsilon \mathbb{R}$

1.3 Motion History and Energy

Motion history and motion energy is particularly helpful in applications related to analyzing a sequence of events. This can include:

1. Sports analysis
2. Gesture recognition
3. Building intelligent machines for imitating a human demonstrator
4. Tracking objects in general

2 Materials and Methods

2.1 Scikit-learn

Scikit-learn is a free software machine learning library using the Python programming language and has several implementations for the various classification, regression and clustering algorithms like support vector machines, random forests, gradient boosting and k-means. It is designed to interoperate with the Python numerical and scientific libraries like NumPy and SciPy. For my assignment I used the Naive Bayes implementation of scikit-learn library.

2.2 Naive Bayes

The Naive Bayes prediction is given by,

Given $x = [u_1, u_2....., u_d]^T$

$$P(y = 1|x) = \frac{P(y = 1) \prod_{i=1}^{n} P(x_i = u_i|y = 1)}{\sum_{y' \epsilon (0,1)} P(y = y') \prod_{i=1}^{n} P(x_i = u_i|y = y')}$$

The assumption taken here is that given a label y, a feature X_i is conditionally independent of other $X_i s$

It can also be expressed as,

$$P(Y = 1|X) = \frac{P(Y = 1)P(X|Y = 1)}{P(Y = 1)P(X|Y = 1) + P(Y = 0)P(X|Y = 0)}$$

2.3 OpenCV Finding Contours

Contours are the curves joining continuous points (along the boundary) which are having same color or intensity. These contours are most useful tool for shape analysis and object detection and recognition. In this assignment finding contours will be used for finding the shape of various objects which will be used as input for our classifier. In order to get better accuracy, binary images should be used which can be done by applying threshold or canny edge detection. This function returns all the contours, and contour approximation compresses horizontal, vertical, and diagonal segments and leaves only their end points. For example, an up-right rectangular contour is encoded with 4 points. There is an optional output vector, containing information about the image topology. It will have as many elements as the number of contours. Not going into too much detail about how it works, this method was devised by Satoshi Suzuki and Keiichi AbePublished, published in Computer Vision, Graphics, and Image Processing in 1985, was a part of Topological structural analysis of digitized binary images by border following.

2.4 Neural Network and Adam optimizer

A neural network consists of an input layer, a number of hidden layers (in which each node is a state) and an output layer. The edges connecting the nodes have weights associated with it and each node is an activation function which is connected to every other node in the next layer. When the input to node multiplied by the weight is passed to the activation function, the output of function decides whether the information will be passed onto the nodes of the next layer. Once all the values are passed on, then at the output layer the values are compared to the actual output. The error found is back propagated to the hidden layers where the weights are adjusted accordingly to get the output correct in the next pass.

According to 'universal approximation theorem' most problems can be learned using one hidden layer. I used 2 layers with 16, 32 hidden units (or neurons) in the expansion and then 32, 16 next. This design was chosen for the hidden layers as per a rule of thumb as given in the neural network design book by Martin Hagan et al.

A number of optimizers can be used for finding the adjustment or change needed for tweaking the weights in the neural network. Adam optimizer is currently a state of the art optimizer which is widely used for deep learning. I have used the same in my assignment. Adam combines two extensions that were made to the classical gradient descent algorithm. Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) that have a per parameter learning rate (hence adaptive).

2.5 Motion History and Energy

In motion history an absolute difference is taken with the predecessor frame. An update is made to the motion history with

$$motionHistory(x, y, t) = timestamp, \;\; if\; D(x, y, t) == 1$$

where $D(x, y, t)$ is the binary image with regions of motion marked. I have also used a decay parameter to decay away the motion history as seen a couple of timesteps away.
Motion energy is given by the following,

$$E_\tau = \cup_{i=0}^{\tau-1} D(x, y, t-1)$$

2.6 Background Subtraction

Using background subtraction the foregorund objects can be identified. Using the foreground mask I can perform the same operations for detecting change or motion in my image. The GMG technique used for background subtraction uses probabilistic foreground segmentation algorithm and detects foreground pixels using Bayesian inference. Unwanted noise is also removed using morphological filtering operations like closing and opening. I use the foreground mask and apply the same operations for generating motion history and energy.

3 Results and Discussions

3.1 Part 1

As mentioned for feature extraction I used opencv find contours. The sticky note was not getting detected because the color change on edges wasn't as prominent as for other objects in the image. So extracted that features of the sticky note separately. Here are some of the results:

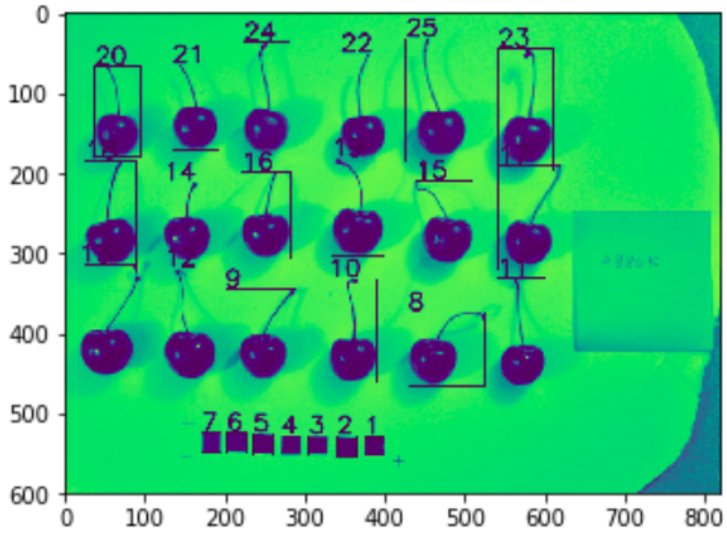

Figure 1: Example of the features extracted

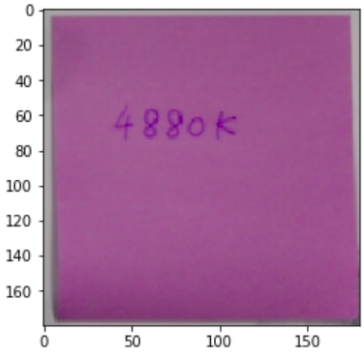

Figure 2: Example of the sticky note extracted separately

```
        print (i, area, label, meanRed[i-1])
```

```
1 585.0 Dark Color Rectangles 36.43512658227848
2 570.0 Dark Color Rectangles 40.30194805194805
3 547.5 Medium Color Rectangles 51.24451939291737
4 595.0 Medium Color Rectangles 59.80529595015577
5 583.5 Medium Color Rectangles 70.58730158730158
6 566.5 Light Color Rectangles 115.38499184339315
7 554.0 Light Color Rectangles 127.88333333333334
8 3097.0 Dark Color Cherries 42.6740467404674
9 3195.5 Dark Color Cherries 50.30199940316324
10 3008.5 Dark Color Cherries 42.39200251889169
11 2931.5 Dark Color Cherries 49.64324846922333
12 3438.5 Dark Color Cherries 44.116768123962366
13 3553.5 Dark Color Cherries 43.938589434164655
14 2826.5 Medium Color Cherries 74.35883547731889
15 3245.5 Medium Color Cherries 50.60405643738977
16 2967.5 Medium Color Cherries 69.11690821256039
17 3336.5 Medium Color Cherries 55.57705479452055
18 3162.0 Medium Color Cherries 67.79319686935581
19 3382.0 Medium Color Cherries 56.05806087936866
20 2667.0 Light Color Cherries 131.18810117563234
21 2706.5 Light Color Cherries 109.6911971830986
22 2750.0 Light Color Cherries 96.20839064649243
23 3947.0 Light Color Cherries 112.85876388218252
24 2927.0 Light Color Cherries 124.87879767291533
25 3697.0 Light Color Cherries 109.26846153846154
```

Figure 3: Example of the values generated from feature contours extracted (does not contain the feature values of sticky note as they were calculated in a separate cell)

Important Note:
The above only shows area, label and mean red values while mean blue, mean green, mean hue, mean saturation, mean intensity and solidity were used for the feature set. Hence, I'm using 8 features per object

3.1.1 Results using a Bayesian Classifier

Experiment 3: Use the feature set to build a Bayesian Classifier

```
In [115]: from sklearn.naive_bayes import GaussianNB

In [116]: trainLen = int(features.shape[0]*0.75)

In [117]: X = features[0:trainLen,:8].astype(np.float)
          Y = features[0:trainLen,9]
          test_X = features[trainLen::,:8].astype(np.float)
          test_Y = features[trainLen::,9]

In [118]: gnb = GaussianNB()
          model = gnb.fit(X, Y)

In [120]: predicted = model.predict(X)
          print ("Training Accuracy: ",(predicted==Y).sum()/len(Y))
          Training Accuracy:  0.8775510204081632

In [121]: predicted = model.predict(test_X)
          print ("Testing Accuracy: ",(predicted == test_Y).sum()/len(test_Y))
          Testing Accuracy:  0.8181818181818182
```

Figure 4: Naive Bayes Result

3.1.2 Results using a Neural Network

Experiment 4: Use the feature set to build a Neural Network Classifier

```
In [94]: from sklearn.neural_network import MLPClassifier

In [152]: clf = MLPClassifier(solver='adam', alpha=1e-5, hidden_layer_sizes= \
                              (16, 32, 32, 16), random_state=1, max_iter=2000)

In [153]: model = clf.fit(X, Y)

In [154]: print("Training Accuracy: ", (model.predict(X)==Y).sum()/len(Y))
          Training Accuracy:  0.9387755102040817

In [156]: print("Testing Accuracy: ",(model.predict(test_X)==test_Y).sum()/len(test_Y))
          Testing Accuracy:  0.8484848484848485
```

Figure 5: Neural Network Result

The neural network does better in both training and test accuracy and also does not show any sign of overfitting

3.2 Part 2

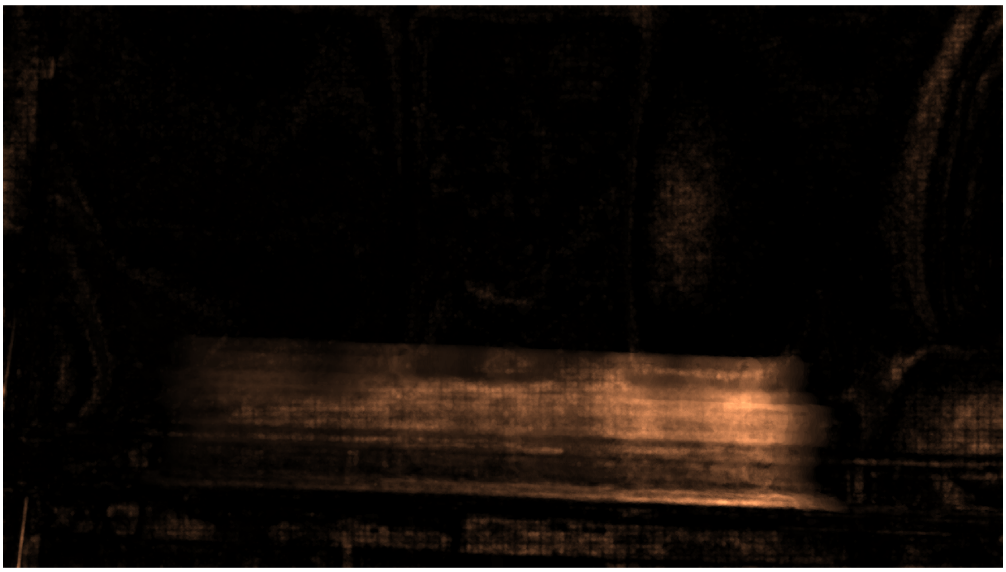

Figure 6: Example of Motion History

I'm decaying the motion history so it'll show history of motion until a certain point.

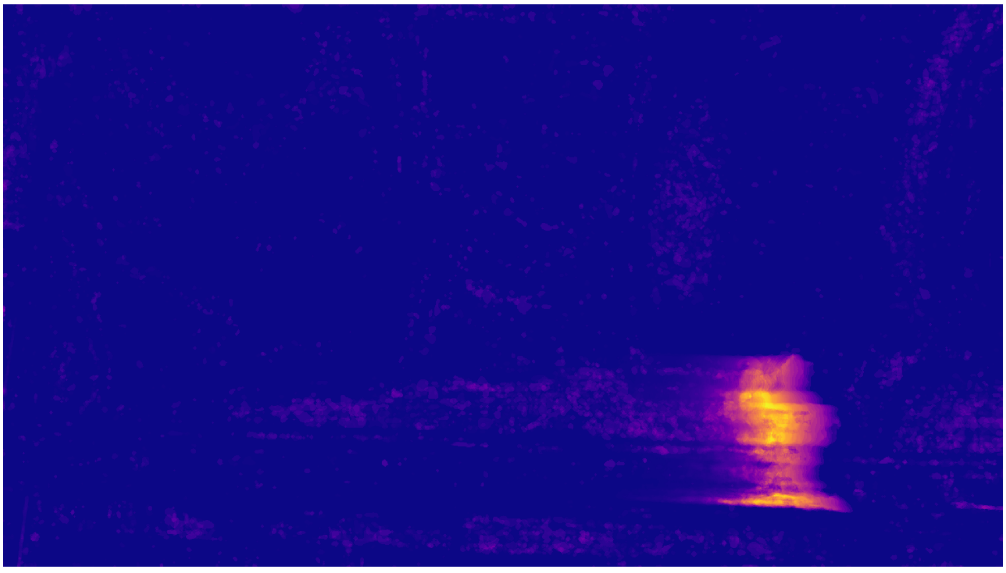

Figure 7: Example of Motion Energy

Important: Please see videos in the zip folder.

4 Conclusion

In part 1 the naive bayes classifier could not perform as well as the neural network classifier. The learning in Generative classifiers is based on creating a model using bayes rules and predict the most likely label of the target variable. While dsicriminative classifiers create a direct map from features to the target variable. The Naive Bayes classifier is the generative classifier. It is a common belief in the Machine Learning community that discriminative classifiers outperform generative classifiers. The aforementioned may not be the case always and it depends on the training set size. Through experiments done (by Andrew Y. Ng, Michael I. Jordan: On Discriminative vs. Generative Classifiers) it has been shown that though discriminative classifier has a lower asymptotic error but a generative classifier also approaches a similar but slightly higher asymptotic error faster.

The motion history and energy results might look a bit different than the matlab implementations as I am using background subtraction but it able to capture the essense in history and energy. **I use the same concept of union of the changed pixels (for energy) and timestamp decaying for motion history.**