EE 573 Pattern Recognition Project 5

Oğuzhan Sevim

I. INTRODUCTION

In this project, we design 3 different classifiers. The first task is to implement a Perceptron model which is a linear binary classifier. In the second task, we build a linear classifier with Ho–Kashyap method.

A. Dataset

The given dataset consists of $n = 1315$ samples where each sample has $d = 600$ features. The given dataset is classified into $C = 8$ different classes. For each class c, where $c = 1, \ldots, 8$, we portion the randomly selected 75% of the class c data as D_c and the remaining 25% as T_c. Also, each given data point has been partitioned into 6 different feature types where each type consist of 100 features. A voting scheme within the feature types will be used for the classification. Since distribution of D_c sets are used for future predictions, they can be considered as training data. Therefore, in the remaining of this report, D_c will be referred as training sets, where we will call T_c as test sets.

The dataset is partitioned into $t_f = 6$ feature types. So, for each task, there will be 6 different classification rule where at the end a voting scheme will be implemented for final decision.

Since some features have considerable higher values than the others, a zero mean unit variance standardization is applied on the dataset for each task.

II. TASK 1: PERCEPTRON MODEL

In this section, we build a multiclass classifier by using fixed–increment single–sample perceptron models. For each $t_f = 6$ feature types we will be implementing $c = 8$ different one–vs–all perceptron models.

A fixed–increment single–sample perceptron is used to create linear decision surfaces and updates its weights by using the misclassified samples on the update rule. The algorithm is designed to be stopped when there is no misclassified samples left in the dataset. However, our dataset is not linearly separable. Therefore, for each model, we need to set a limit for the maximum number of epochs such as 500. The results of the algorithm are shown in Table I.

The corresponding code for this section can be found in Perceptron_main.m file.

TABLE I
METRICS FOR PERCEPTRON ALGORITHM

Class	**c1**	**c2**	**c3**	**c4**	**c5**	**c6**	**c7**	**c8**
Tra. Prec.	0.95	0.85	0.98	0.96	0.96	0.95	1.00	0.93
Tra. Recall	0.98	0.97	0.99	0.79	0.83	1.00	1.00	1.00
Tst. Prec.	0.91	0.80	0.97	0.94	0.92	0.94	0.97	0.88
Tst. Recall	1.00	0.95	0.95	0.75	0.77	1.00	0.93	0.97

III. TASK 2: HO–KASHYAP METHOD

In this section, we train a classifier by using Ho–Kashyap method. Like the previous methods, we applied standardization in the preprocessing process. For each of the feature types, $c = 8$ different models are trained for each class. The classification is carried out by *one–vs–all* method. For the test, the decision is made by looking at $c = 8$ different outcomes and choosing the class with highest. Then, a voting scheme is applied within different feature types for final decision. The results are shown in Table II.

The corresponding code for this section can be found in HoKash_main.m file.

TABLE II
METRICS FOR HO–KASHYAP METHOD

Class	**c1**	**c2**	**c3**	**c4**	**c5**	**c6**	**c7**	**c8**
Tra. Prec.	0.98	0.98	0.99	0.93	0.99	1.00	1.00	1.00
Tra. Recall	1.00	1.00	1.00	0.97	0.84	0.99	0.99	1.00
Tst. Prec.	0.94	0.97	0.98	0.94	0.97	1.00	1.00	1.00
Tst. Recall	1.00	0.97	1.00	0.97	0.93	1.00	0.90	0.97

IV. COMPARISON OF APPROACHES

By looking at the Tables I and II, one can see that Ho–Kashyap algorithm outperforms the Perceptron algorithm both in test set and the training set.