EE 58*J* Data Mining Project 2

Oğuzhan Sevim

April 25, 2020

I. INTRODUCTION

In this project, we will perform image recognition on the Vispera-SKU101 – 2019 dataset. In order to extract features, 38 different filters will applied on the given images. For the classification purpose, we will be using variations of minimum distance classifier and naive Bayes classifier. Python programming language is preferred for this assignment.

II. DATA

Vispera-SKU101 – 2019 data is composed of approximately 10.000 arbitrary–sized images in jpeg format. Each image belongs to one of 101 classes, and is kept in the folder named with SKU (stock keeping unit) it belongs to. Each of these class folders contains approximately 100 instances/images. An example image is given in Figure 1 in which the image belongs to class of ”33*cl* Fanta in can” with the class SKU of 8736.

Fig. 1. An example image with 75×224 pixels. The image belongs to ”33*cl* Fanta in can” class with SKU of 8736

The data is first downloaded to the local computer, and then retrieved one by one for further processes. In order to navigate easily between the local folders, ”os” library is used.

III. IMAGE DESCRIPTORS

After resizing all the images to 128×128 pixels, images are filtered by RFS filter bank kernels. The RFS filter bank consists of 2 anisotropic filters (an edge and a bar filter, at 6 orientations and 3 scales), and 2 rotationally symmetric ones (a Gaussian and a Laplacian of Gaussian) which results in 38 filters in total. Python script for RFS filter bank can be obtained from https://gist.github.com/amueller/3129692. Filters are applied by using OpenCV library's $filter2D()$ function which convolves a given filter on every channel of image.

After filtering, we each image ends up with $38 \times 128 \times 128 \times 3 = 1,867,776$ features which is quite high to play with. Therefore, a max pooling with 8×8 window is applied on each image. As a result we ended up with $38 \times 16 \times 16 \times 3 = 29,184$ features for each image. Max pooling is done by using scikit-image library's $measure.block\_reduce()$ function.

The feature extraction process described above uses RGB images as input. On top of that, images are turned into HSV and their features are also tested for experimental purposes.

IV. MINIMUM DISTANCE CLASSIFIERS

After obtaining the descriptors from images, minimum distance method is used for classification of the test images. This algorithm is implemented in 2 different ways. First, each test sample is classified by the label of class that has the closest mean vector to the given test instance. This rule can be described as

$$y^* = \underset{c}{\operatorname{argmin}} (\mu_c - x)^T (\mu_c - x) \quad (1)$$

where μ_c is the mean vector of instances that belong to class c. By taking variance of features into consideration, another classification rule can be expressed as follows:

$$y^* = \underset{c}{\operatorname{argmin}} (\mu_c - x)^T \sigma^{-1} (\mu_c - x) \quad (2)$$

where σ is the diagonal variance matrix. Each element of σ corresponds the variance of a single feature in training data.

When we test these classification rules with different max pooling sizes and different image formats, the results shown in Tables I and II are obtained. In these tables, each column corresponds to max pooling window sizes.

TABLE I
MINIMUM DISTANCE CLASSIFIER WHEN VARIANCE IS IGNORED

	8x8	**16x16**
RGB	40.44	43.1
HSV	52.54	54.96

TABLE II
MINIMUM DISTANCE CLASSIFIER WITH VARIANCE

	8x8	**16x16**
RGB	49.88	48.67
HSV	59.08	55.93

By interpreting Tables I and II, we may state that HSV coded images are better classified with minimum distance classifiers. Also, we get the maximum accuracy 59.08% when max pooling with 8×8 window size is used with the classification rule given in (2).

V. NAIVE BAYES CLASSIFIERS

In this section, we will experiment with different Naive Bayes (NB) classifiers. For a given test sample x, Naive Bayes classification rule can be defined as follows:

$$y^* = \operatorname*{argmax}_y p(y) \prod_{k=1}^{d} p(x_k|y) \tag{3}$$

where prior $p(y)$ can assumed to be equal for all classes for simplicity.

A. NB Classifier with Gaussian Parametric Estimation

In the first NB classifier, we will make a simple parametric estimation that each feature likelihood $p(x_k|y)$ will be approximated by the normal distribution $\mathcal{N}(\mu_k^y, \sigma_k^{y2})$. Here, μ_k^y is the mean of feature k in class y, and σ_k^{y2} is the variance. Then, by taking natural logarithm of both sides of (3) our first NB classification rule can be derived as follows:

$$\begin{aligned} y^* &= \operatorname*{argmax}_y \frac{1}{\prod_{k=1}^{d} \sigma_k^y \sqrt{2\pi}} e^{-\frac{1}{2}\sum_{k=1}^{d}\left(\frac{x_k-\mu_k^y}{\sigma_k^y}\right)^2} \\ &= \operatorname*{argmax}_y d \ln \frac{1}{\sqrt{2\pi}} - \sum_{k=1}^{d} \ln \sigma_k^y - \frac{1}{2}\sum_{k=1}^{d}\left(\frac{x_k-\mu_k^y}{\sigma_k^y}\right)^2 \\ &= \operatorname*{argmin}_y \sum_{k=1}^{d} \ln \sigma_k^y + \frac{1}{2}\sum_{k=1}^{d}\left(\frac{x_k-\mu_k^y}{\sigma_k^y}\right)^2 \\ &= \operatorname*{argmin}_y \sum_{k=1}^{d}\left(\ln \sigma_k^{y2} + \frac{(x_k-\mu_k^y)^2}{\sigma_k^{y2}}\right). \end{aligned} \tag{4}$$

When the rule given in (4) is used, accuracies are obtained as shown in Table III. It seems like our first Naive Bayes classifier outperforms the minimum distance classifiers of the previous section.

TABLE III
NAIVE BAYES CLASSIFIER WITH GAUSSIAN PARAMETRIC ESTIMATION

	8x8	16x16
RGB	50.12	53.75
HSV	61.50	61.74

B. NB Classifier with Parzen-Window Estimation

Next, we will implement the second NB classifier by estimating the likelihoods $p(x_k|y)$ using histogram density estimation which is a kind of Parzen-window method. Density of feature k in class y, $p(x_k|y)$, will be approximated by the following equation:

$$p(x_k|y) = \sum_{i=1}^{n_y} \phi(x_k - x_i) \tag{5}$$

where $\phi(x)$ is the window function that will slide over $1D$ feature space. In order to have a valid density function (i.e., its integral over $1D$ feature space is 1), we may define the window function as follows:

$$\phi(x) = \begin{cases} \frac{1}{n_y h}, -h/2 < x < h/2, \\ 0, \text{otherwise}. \end{cases} \tag{6}$$

Here, h is the window width that needs to be tuned experimentally, and n_y is the number of samples in class y. When we implement the classification rule with different h parameters Table IV is obtained. Please note that it is possible that some $p(x_k|y)$ values may end up being zero. This makes whole product in (3) zero as well. As a solution, zero $p(x_k|y)$ values are simply replaced with a small positive number which is determined by tuning.

TABLE IV
NAIVE BAYES CLASSIFIER WITH PARZEN WINDOW ESTIMATION

h=2	h=5	h=8	h=10
67.07	67.80	67.07	65.86

VI. CATEGORY–WISE ACCURACIES

From the experimental results given in previous sections, the best accuracy is obtained when the images are turned into HSV format. Also, the best classifier is the Naive Bayes classifier with Parzen window estimation where the window width is taken as $h = 5$. When we apply this combination as the classifier, classification accuracies of each category and the confusion matrix are obtained as shown in Figures 2 and 3, respectively.

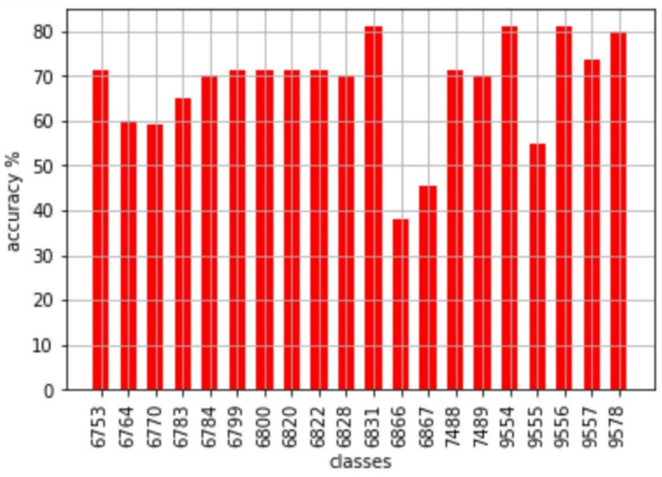
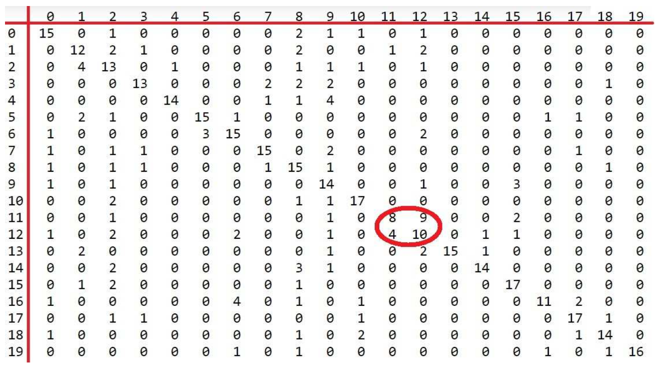


Fig. 2. Accuracies of each confectionary class



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	15	0	1	0	0	0	0	0	2	1	1	0	1	0	0	0	0	0	0	0
1	0	12	2	1	0	0	0	0	2	0	0	1	2	0	0	0	0	0	0	0
2	0	4	13	0	1	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	2	2	2	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	14	0	0	1	1	4	0	0	0	0	0	0	0	0	0	0
5	0	2	1	0	0	15	1	0	0	0	0	0	0	0	0	0	1	1	0	0
6	1	0	0	0	0	3	15	0	0	0	0	0	2	0	0	0	0	0	0	0
7	1	0	1	1	0	0	0	15	0	2	0	0	0	0	0	0	0	1	0	0
8	1	0	1	1	0	0	0	1	15	1	0	0	0	0	0	0	0	0	1	0
9	1	0	1	0	0	0	0	0	0	14	0	0	1	0	0	3	0	0	0	0
10	0	0	2	0	0	0	0	0	1	1	17	0	0	0	0	0	0	0	0	0
11	0	0	1	0	0	0	0	0	0	1	0	8	9	0	0	2	0	0	0	0
12	1	0	2	0	0	0	2	0	0	1	0	4	10	0	1	1	0	0	0	0
13	0	2	0	0	0	0	0	0	0	1	0	0	2	15	1	0	0	0	0	0
14	0	0	2	0	0	0	0	0	3	1	0	0	0	0	14	0	0	0	0	0
15	0	1	2	0	0	0	0	0	1	0	0	0	0	0	0	17	0	0	0	0
16	1	0	0	0	0	0	4	0	1	0	1	0	0	0	0	0	11	2	0	0
17	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	17	1	0
18	1	0	0	0	0	0	0	0	1	0	2	0	0	0	0	0	0	1	14	0
19	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	16

Fig. 3. Confusion matrix for confectionary class

From Figure 2, we see that classes with SKU 6866 and 6867 have the lowest accuracies. Also, by inspecting the confusion matrix, we can see that most confused classes are again these two classes. When we look the products in these folders, we see that these classes have the same product: Altınbaşak Girissini. That is why they are mostly confused.