Report: Exercise 3: Naïve Bayes Classifier for Spam Filtering-COMP 24111

Name: Kai Chen **Student ID:** 10411044 **Date:** Nov. 18th, 2018

1. Here is my train function and it's used for both **discrete and continuous** occasions:

   [p_condition, p_class] = NBTrain(AttributeSet, LabelSet, config, use_padding, use_cv)

   I add **additional input parameters**. **'Config'** summaries training set, including is_continuous, num_class, num_attr and D. **Use_padding** is bool telling whether to deal with **zero** probability. **Use_cv** is bool telling whether to **use validation** to choose m.

   Training has **two output**: **p_class & p_condition**. **P_class** is a **(num_class, 1) vector** and **P_class(i)** means probability of data point belonging to **i-th class** in training set. It's **same** between discrete and continuous occasions.

   P_condition is **different** between two. For **discrete**, p_condition represents **P(xj = ajk | ci)**, conditional probability that j-dimension of a training data equals ajk given i-th class. Here ajk is k-th possible value for xj. However for **continuous**, it stands for **mean(xj | ci) & var(xj | ci)**, because we use a normal distribution to fit every feature in every class space and every normal distribution needs average and variance.

2. Part 1 are discrete datasets, while Part 2 are continuous. As I said in Question 1, we can save **p_class in a (num_class, 1) vector** easily for both Part 1 and 2.

   For Part 1, my **p_condition has 3 dimensions, including classes, dimensions of data points and possible values for attributes**. It is a 3-D **tensor** and here I use a **cell array**. Length of p-condition is **num_attr** and every cell of p_condition is a **(num_class, D) matrix**. Every element of the matrix means the conditional probability **P(xj = ajk | ci).**

   For Part 2, **p_condition is a cell array length of 2** and every cell is a **(num_class, D)** matrix. **P_condition{1}** records the **average** of every dimension in every class space, while **p_condition{2}** records the **variance**.

3. 3 discrete datasets in Part 1. I randomly choose **10% training data for validation,** run all 3 datasets 10 times and get an average accuracy. Each dataset I have two accuracy **(accuracy, accuracy_origin)**. Accuracy is result without zero probability, while accuracy_origin is opposite. I calculate **related error** (**Euclidean distance**) to see influence of zero probability. Here is experiment result: **'av2_c2.mat'**: (0.8912, 0.8909), **'av3_c2.mat'**: (0.8940, 0.8935) and **'av7_c3.mat'** (0.8653, 0.8626). **Related error is 0.0016**. We can see there is a **stable accuracy increase (about 0.2%)** after validation. I get **high class-0 sensitivity, precision and F1-score (all over 90%)** in first two datasets. One given continuous dataset in Part 2: **'avc_c2.mat'**. **Test accuracy is 78.4643%**. It has a **93.3% class-0 precision** but its **sensitivity** is only **71.0%. F1-score** is **0.806**.

4. Here I talk about **'spambase.data'**. I run the 10-fold cross validation and get an **average accuracy 81.8696%, and standard deviation is 0.019449**. All folder's **class-0 sensitivity is over 90%** but **precision is around 70% and F1-score around 0.8**, similar to 'avc_c2.mat'.

   The **motivation** is to **fully use dataset when it's small**. I divided dataset equally into 10 different folders and every time I choose one of them as test set and rest as training set.

As a result, **every data point is used as training data for 9 times and as testing data once, without over-fit problem** at the same time. As for cross validation **setting**, I **randomly shuffler dataset first** to introduce a random factor, because the original dataset firstly lists all class-0 data points and then all class-1, so if we run 10-folder on the original dataset directly, it will cause an **unbalanced dataset** and raise risk of under fitting and over fitting.

5. Non-trivial implication

① **Naïve Bayes performs better in discrete datasets than in continuous ones.**

② **Effect of zero probability**

We see in Question 3 adding m gains a **stable but unobvious** increase in all 3 discrete datasets. Zero probability **will cause a cascade effect**, but it only matters when test set has some value which doesn't exist in training set in some class space. In our case, there is **data similarity between training and test set**, so even if many zero probabilities exit in all 3 datasets, padding doesn't change a lot because the **trigger isn't met often**.

③ **Different ways of choosing 'm'**

Here we try 2 different ways to choose m. (1) **Do validation.** I randomly choose 10% training set as validation set (result shown in Q3). (2) Setting m as a **constant** (here we choose 1). Here is the result of constant m: **'av2_c2.mat'**: (0.8887, 0.8909), **'av3_c2.mat'**: (0.8921, 0.8935) and **'av7_c3.mat'** (0.8649, 0.8626), **related error 0.0017**. We see accuracy may **decrease when m is constant and it's not stable**.

However, validation **training time is much longer than constant m** because for every zero probability you need to **use different m and run test function many times**. What's worse, we **can't do any vectorization**. If you care about accuracy more, you choose validation If you care about training time more, you had better choose constant m.

④ **Best m**

M's candidates are [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 15, 20]. I check validation result and find best m always comes from **[0.01, 0.05] or [10, 15, 20]**, which are **smallest and largest** parts, and mostly the **smallest one**. It's reasonable because zero padding only matters when losing value exists in test set. **When it's rare in test set, m is small** because it should not change **relative probability distribution** so much when **avoiding cascade effect**. When missing value appears **frequently, m should be large**.

⑤ **Naïve bayes classifier can't deal with unbalanced dataset very well**

'av7_c3.mat' is an unbalanced dataset with 1285 class-0 objects, 769 class-1 and 246 class-2 in test set and approximately **same rate** in training set. The result shows **92.9%** class-0 test objects are correctly classified, **82.4%** for class-1 and only **67.1%** for class-2 from **confusion matrix**.I don't think unbalanced result comes from over fitting. Naïve Bayes is a generative model based on **probability distribution**. Unbalanced training data will affect the **prior probability** for each class and restrict the prediction.

⑥ **Normal distribution or uniform distribution——variance equal 0**

In 'avc_c2.mat', I find one feature in class-1 in training set is constant so variance is 0 and prediction is always 0 first. It's not normal distribution any more if everyone is the same. So I check first if **variance is 0 and x is the average, probability (density actually) remains. Or it will be set to 0**. It improves **'avc_c2.mat' from 63% to 78%**.