A2Q4

March 2, 2020

1 CMPT 423/820

1.1 Assignment 2 Question 4

- Seyedeh Mina Mousavifar
- 11279515
- sem311

1.1.1 Theoretically

$$P(w_j|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|w_j)P(w_j)}{P(\mathbf{x}_i)}$$

where
$\mathbf{x}_i$ is the feature vector of sample i, $i \in 1, 2, ..., n$,
w_j is the notation of class j, $j \in 1, 2, ..., m$,
$P(\mathbf{x}_i|w_j)$ is the probability of observing sample $\mathbf{x}_i$ given that it belongs to class w_j.
So the decision rule is:

$$predicted\ class\ label \leftarrow arg\ max\ P(w_j|\mathbf{x}_i) \quad for\ j = 1, ..., m$$

Further, the class condition probilities of individual features d are as follows because of *naive* conditional independence assumption.

$$P(\mathbf{x}|w_j) = P(x_1|w_j)...P(x_d|w_j) = \Pi_{k=1}^{d} P(x_k|w_j)$$

Another assumption of *Naive Bayes* is that $P(x_i = b|w_j)$ is drawn from a particular distribution, which is the reason of calling *Naive Bayes* a *generative model*.

Bernoulli Model We use the *Bernoulli distribution* to compute the likelihood of a binary variable. For example, we could estimate $P(x_k = 1|w_j)$ via *MLE* as the frequency of occurences in the training set:

$$\theta = P(x_k = 1|w_j) = N_{x_k,w_j} / N_{w_j}$$

This means, number of training samples in class w_j that have the property $x_k = 1$ noted by N_{x_k,w_j} divided by by all training samples in j, noted by N_{w_j}.

$$P(\mathbf{x}|w_j) = \Pi_{k=1}^{d} P(x_k|w_j) \quad (1.1)$$

$$P(\mathbf{x}|w_j) = \Pi_{k=1}^{d} (\theta^{x_k}(1-\theta)^{1-x_k})$$

Gaussian Model Typically, the *Gaussian Naive Bayes* model is used for variables on a continuous scale, assuming that the variables are normally distributed.

$$P(x_k|w_j) = \frac{1}{\sqrt{2\pi\sigma^2_{w_j}}} exp(-\frac{(x_k - \mu_{w_j})^2}{2\sigma^2_{w_j}})$$

Therefore, we need to estimate mean μ of the samples associated with class w_j and the variance σ^2 associated with class w_j.

$$P(\mathbf{x}|w_j) = \Pi_{k=1}^{d} P(x_k|w_j) \quad (1.2)$$

Mixed Model Since *Naive Bayes* assumes conditional independence between features, it can be written as:

$$P(\mathbf{x}|w_j) = \Pi_{k=1}^{d} P(x_k|w_j)$$

By assuming that features $a, ..., b$ are from *Bernoulli* distribution and the rest are from *Guassian* distribution, we can write the above equation as:

$$P(\mathbf{x}|w_j) = \Pi_{k=a}^{b} P(x_k|w_j) \Pi_{k=b+1}^{d} P(x_k|w_j)$$

So considering equation (1.1), and (1.2) the equation would be:

$$P(\mathbf{x}|w_j) = P(\mathbf{x}_{bernoulli}|w_j) P(\mathbf{x}_{guassian}|w_j)$$

This can also be applied to *Multinoulli* instead of *Bernoulli* distribution.

1.1.2 Practically

For this part, I'll use *Forest covertypes* dataset.

In this dataset, there are seven covertypes, making this a multiclass classification problem. Each sample has 54 features, described here. Some of the features are boolean indicators, while others are discrete or continuous measurements.

It has 54 columns of data. 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables.

```
In [59]: from sklearn.datasets import fetch_covtype
         import pandas as pd

         covtype = fetch_covtype()

         data = pd.DataFrame.from_records(covtype.data)
         labels = covtype.target

         print(covtype.DESCR)
```

```
.. _covtype_dataset:

Forest covertypes
-----------------
```

```
The samples in this dataset correspond to 30Œ30m patches of forest in the US,
collected for the task of predicting each patch's cover type,
i.e. the dominant species of tree.
There are seven covertypes, making this a multiclass classification problem.
Each sample has 54 features, described on the
`dataset's homepage <https://archive.ics.uci.edu/ml/datasets/Covertype>`__.
Some of the features are boolean indicators,
while others are discrete or continuous measurements.

**Data Set Characteristics:**

    =================   ============
    Classes                        7
    Samples total             581012
    Dimensionality                54
    Features                     int
    =================   ============

:func:`sklearn.datasets.fetch_covtype` will load the covertype dataset;
it returns a dictionary-like object
with the feature matrix in the ``data`` member
and the target values in ``target``.
The dataset will be downloaded from the web if necessary.
```

In [157]:
```python
from sklearn.model_selection import train_test_split

# Set aside  data as a part of test set
tpropn = 0.2
X_train, X_test, Y_train, Y_test = train_test_split(data,
                                                    labels,
                                                    test_size=tpropn)


# continues part of train data
data_con = X_train.values[:, :10]
test_con = X_test.values[:, :10]

# discrete part of train data
data_cat = X_train.values[:, 10:]
test_cat = X_test.values[:, 10:]
```

I independently fit a *Gaussian NB model* on the continuous part of the data and a *Multinomial NB model* on the categorical part. Then transform all the dataset by taking the class assignment probabilities, with *predict_proba* method. Then I multiply these probabilities and find the most probability for each record, as predicted label.

```
In [161]: from sklearn.naive_bayes import CategoricalNB, GaussianNB
          import numpy as np

          # fitting model for categorical part
          clf_cat = CategoricalNB()
          clf_cat.fit(data_cat, Y_train)

          # fitting model for continuous part
          clf_con = GaussianNB()
          clf_con.fit(data_con, Y_train)

Out[161]: GaussianNB(priors=None, var_smoothing=1e-09)

In [162]: def mixed_predictor(model_cat,
                              model_con,
                              data_in_cat,
                              data_in_con):
              """
              :purpose: This function find class labels with
               Naive Bayes classifier on mixed class.
              :param model_cat: learned categorical NB
              :param model_cat: learned continuous NB
              :param data_in_cat: categorical part of dataset
              :param data_in_con: continues part of dataset
              :return: labels on the dataset
              """
              prob_cat = model_cat.predict_proba(data_in_cat)
              prob_con = model_con.predict_proba(data_in_con)
              new_feature = prob_cat*prob_con

              # prediction based on probabilites
              new_feature = pd.DataFrame(new_feature,
                           columns=range(1,8))

              new_feature['pred_label'] = new_feature.idxmax(axis=1)

              return new_feature.pred_label.values
```

Evaluation For evaluation I compare this model with *Guassian NB* and *Categorical NB* classifier.

```
In [166]: from sklearn.metrics import accuracy_score
          from sklearn.metrics import f1_score

          # on training set
          y_predicted = mixed_predictor(clf_cat,
                                        clf_con,
                                        data_cat,
                                        data_con)
```

```python
f1 = f1_score(Y_train,
              y_predicted,
              average='macro')

acc = accuracy_score(Y_train,
              y_predicted)

# printing result in tabular format
print('\033[1m' + 'Multi-class NB classifier' + '\033[0m')
print('{:<15} {:<15} {:<15} {:<15}'.format('Predictor',
                            'Accuracy',
                            'F1-score',
                             'type'))

print('{:<15} {:<15} {:<15} {:<15}'.format('Mixed',
                             round(acc,5),
                             round(f1,5),
                             'training set'))

# on test set
y_predicted = mixed_predictor(clf_cat,
                              clf_con,
                              test_cat,
                              test_con)
f1 = f1_score(Y_test,
              y_predicted,
              average='macro')

acc = accuracy_score(Y_test,
              y_predicted)

print('{:<15} {:<15} {:<15} {:<15}'.format('Mixed',
                             round(acc,5),
                             round(f1,5),
                             'test set'))

# Guassian Naive Bayes Predictor
# on training set
y_predicted = clf_con.predict(data_con)

f1 = f1_score(Y_train,
              y_predicted,
              average='macro')

acc = accuracy_score(Y_train,
              y_predicted)

# printing result in tabular format
```

```python
print('{:<15} {:<15} {:<15} {:<15}'.format('Guassian',
                                   round(acc,5),
                                   round(f1,5),
                                   'training set'))

y_predicted = clf_con.predict(test_con)

f1 = f1_score(Y_test,
              y_predicted,
              average='macro')

acc = accuracy_score(Y_test,
              y_predicted)

# printing result in tabular format
print('{:<15} {:<15} {:<15} {:<15}'.format('Guassian',
                                   round(acc,5),
                                   round(f1,5),
                                   'test set'))

# Categorical Naive Bayes Predictor
# on training set
y_predicted = clf_cat.predict(data_cat)

f1 = f1_score(Y_train,
              y_predicted,
              average='macro')

acc = accuracy_score(Y_train,
              y_predicted)

# printing result in tabular format
print('{:<15} {:<15} {:<15} {:<15}'.format('Categorical',
                                   round(acc,5),
                                   round(f1,5),
                                   'training set'))

y_predicted = clf_cat.predict(test_cat)

f1 = f1_score(Y_test,
              y_predicted,
              average='macro')

acc = accuracy_score(Y_test,
              y_predicted)

# printing result in tabular format
print('{:<15} {:<15} {:<15} {:<15}'.format('Categorical',
```

```
                                              round(acc,5),
                                              round(f1,5),
                                              'test set'))
```

```
Multi-class NB classifier
Predictor        Accuracy         F1-score         type
Mixed            0.68043          0.48956          training set
Mixed            0.67989          0.48803          test set
Guassian         0.63047          0.44924          training set
Guassian         0.62923          0.44694          test set
Categorical      0.63226          0.48263          training set
Categorical      0.63312          0.48605          test set
```

So we can see that we have better accuracy and f1-score for mixed classifier.