Machine Learning Lab 4

Naïve Bayesian classifier

In this lab, you will use Naïve Bayesian classifier to solve the problem of handwritten digit recognition. As for the dataset, we still use MNIST which we used in our previous labs. Note that using Naïve Bayesian, we can directly do multi-class classification similar to KNN. You should finish the following tasks.

1. Consider each dimension of digits is conditionally independent given the digit label, and assume a uniform prior, please implement a Naïve Bayesian classifier as described in the lecture notes. You should use *m*-estimate with *m* at your choice. For example, try *m=3* and *m=10* to compare the performance (error rate, f1, etc).
   Hints:
   - P(d|X) = P(X|d)P(d) / P(X)
   - P(X|d) = P(x0|d)P(x1|d)...P(xn|d)
   - P(dk|X) = max(P(d1|X), P(d2|X), ... , P(d10|X))
   - Priori Probability - P(d)
   - Posterior Probability - P(X|d)

784个digit的概率累成，会超出float的精度
可以取log，把乘法转换成加法

```
def Bayes_train(train_x, train_y, m):
    # prior probability - P(d)
    totalNum = train_x.shape[0]
    classNum = Counter(train_y)
    # P(d=i) = total number of class i / total number of all classes
    prioriP = np.array([classNum[i] / totalNum for i in range(10)])

    # posterior probability - P(X|d), also a set of P(xi|d)
    # create empty array with shape of (10, 784), 10 refers to the number
    of classes and 784 refers to the number of features
    posteriorNum = np.empty((10, train_x.shape[1]))
    posteriorP = np.empty((10, train_x.shape[1]))
```

2. In the lecture notes, the likelihood *P(x|d)* is estimated by calculating the frequency of a value at a given dimension (pixel location) for a given digit label. This may not be the best choice even with *m*-estimate. This is because for a given pixel location, there may be many different intensity values in the training data for a given digit label. A more conventional way to estimate the likelihood term is to assume that the values at each dimension (pixel location) follow Gaussian distribution http://en.wikipedia.org/wiki/Normal_distribution. The mean and standard deviation of the Gaussian function at the given pixel location for a given digit label can be easily estimated in Python using `mean` and `std` function from `numpy`. Assuming `x` is the variable, `mu` (μ) and `sd` (σ) are the mean and standard deviation, you can calculate the Gaussian using the following code:

```
1/((2*pi)**0.5*sd)*exp(-(x-mu)**2/(2*sd**2))
```

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Implement Naïve Bayesian classifier using our new way to likelihood estimation, and compare the performance with the version in task 1 and see which gives better results.