CSC2515 Assignment 3

Kairui Wu

Q1

```
11314 train data points.
101631 feature dimension.
Most common word in training set is "the"
BernoulliNB baseline train loss = 18.7343114725
BernoulliNB baseline train accuracy = 0.5987272405868835
BernoulliNB baseline test loss = 23.5280138078
BernoulliNB baseline test accuracy = 0.4579129049389272
rf regression train loss = 4.65927169878
rf train accuracy = 0.89234576630723
rf test loss = 19.1143122677
rf test accuracy = 0.6340945300053107
gaussian naive bayes train loss = 4.92425313771
gaussian naive bayes train accuracy = 0.9025101643980908
gaussian naive bayes test loss = 24.7126925119
gaussian naive bayes test accuracy = 0.5533722782793414
logistic regression train loss = 1.43318013081
logistic regression train accuracy = 0.9653526604207177
logistic regression test loss = 22.9646840149
logistic regression test accuracy = 0.6071428571428571
most confused classes are [[15 19]]
```

1. The train and test losses of four algorithms.

	Bernoulli Naive-Bayes(Baseline)	Random Forest	Gaussian Naïve-Bayes	Logistic Regression
Train Loss	18.73	4.66	4.92	1.43
Test Loss	23.53	19.11	24.71	22.96
Train Accuracy	0.60	0.89	0.90	0.97
Test Accuracy	0.46	0.63	0.55	0.61

2. For picking best hyperparameters, I used the method of grid search to do cross validation for finding the optimal parameters. Firstly, choosing a wide range of values of the hyperparameters. If the optimal value is at the upper limited or the lower limited, the range would be changed (same width). Until finding that the optimal value is between the upper limited and the lower limited of the range we choose, we can narrow down the range to find the optimal hyperparameter.

3. The algorithms I choose are Random Forest, Gaussian Naïve Bayes and Logistic Regression. SVM algorithms are not popular in dealing with the dataset of which there are many features, but in our dataset, there are more than 100,000 features. Random Forest has a high efficiency in dealing with classification,

especially there are more than 10 labels. Gaussian Naïve-Bayes and Logistic Regression both are stable in dealing with classification problem. That's why I choose them. Although like my previous idea, Gaussian Naïve-Bayes is sort of slow due to high dimension data in this problem, Logistic Regression performs much faster than I thought. Gaussian Naïve-Bayes performs worse than I thought, that is probably because the high dimension of data causes the overfitting.

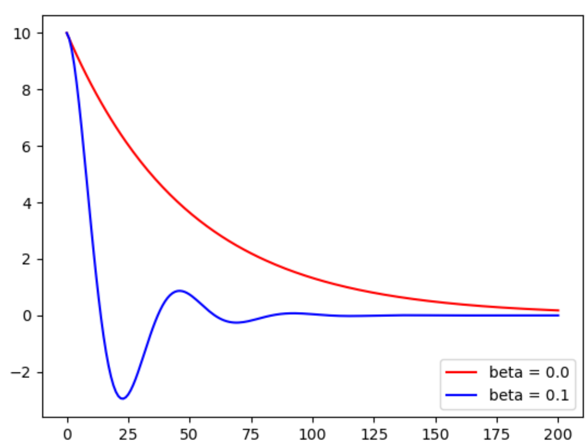
4. The best classifier is Random Forest. The confusion matrix is below.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	95.00	1.00	2.00	0.00	0.00	0.00	0.00	3.00	0.00	0.00	0.00	0.00	0.00	2.00	6.00	16.00	5.00	20.00	10.00	26.00
1	2.00	240.00	20.00	15.00	6.00	39.00	4.00	5.00	1.00	7.00	1.00	10.00	28.00	20.00	8.00	4.00	1.00	1.00	2.00	3.00
2	2.00	36.00	262.00	42.00	10.00	32.00	2.00	4.00	2.00	0.00	0.00	6.00	13.00	4.00	3.00	6.00	2.00	2.00	1.00	3.00
3	0.00	11.00	26.00	236.00	29.00	8.00	17.00	2.00	3.00	1.00	0.00	4.00	30.00	0.00	3.00	0.00	1.00	0.00	1.00	0.00
4	0.00	10.00	14.00	25.00	259.00	3.00	12.00	1.00	1.00	0.00	1.00	3.00	18.00	2.00	6.00	0.00	3.00	0.00	1.00	0.00
5	2.00	27.00	15.00	9.00	7.00	272.00	0.00	3.00	0.00	1.00	2.00	4.00	10.00	1.00	3.00	1.00	3.00	1.00	0.00	0.00
6	7.00	10.00	4.00	12.00	16.00	4.00	310.00	13.00	11.00	3.00	1.00	4.00	17.00	14.00	6.00	2.00	4.00	2.00	4.00	3.00
7	9.00	7.00	4.00	4.00	5.00	1.00	5.00	261.00	26.00	2.00	2.00	3.00	22.00	9.00	7.00	1.00	8.00	3.00	5.00	5.00
8	17.00	9.00	3.00	3.00	5.00	3.00	6.00	29.00	294.00	7.00	5.00	11.00	11.00	23.00	15.00	12.00	12.00	15.00	13.00	10.00
9	23.00	10.00	27.00	12.00	23.00	15.00	18.00	38.00	31.00	332.00	38.00	35.00	26.00	32.00	28.00	26.00	28.00	24.00	21.00	18.00
10	8.00	1.00	1.00	1.00	2.00	0.00	1.00	0.00	1.00	34.00	341.00	0.00	4.00	2.00	4.00	0.00	1.00	0.00	5.00	4.00
11	7.00	3.00	2.00	5.00	2.00	6.00	1.00	3.00	2.00	1.00	0.00	269.00	20.00	1.00	3.00	0.00	14.00	4.00	5.00	1.00
12	3.00	10.00	1.00	27.00	19.00	3.00	5.00	18.00	9.00	2.00	0.00	11.00	177.00	14.00	10.00	2.00	6.00	1.00	4.00	4.00
13	7.00	2.00	2.00	0.00	0.00	1.00	1.00	1.00	4.00	1.00	3.00	2.00	7.00	248.00	6.00	5.00	7.00	3.00	20.00	9.00
14	13.00	7.00	8.00	1.00	2.00	4.00	4.00	5.00	1.00	0.00	0.00	5.00	7.00	7.00	270.00	4.00	5.00	1.00	11.00	5.00
15	102.00	4.00	0.00	0.00	0.00	4.00	1.00	2.00	6.00	5.00	1.00	2.00	1.00	11.00	6.00	314.00	14.00	17.00	14.00	107.00
16	6.00	1.00	2.00	0.00	0.00	0.00	3.00	6.00	4.00	0.00	3.00	22.00	1.00	3.00	6.00	1.00	233.00	14.00	99.00	27.00
17	9.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	2.00	0.00	0.00	1.00	1.00	2.00	260.00	3.00	8.00
18	3.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	2.00	1.00	1.00	3.00	1.00	3.00	3.00	2.00	11.00	8.00	89.00	4.00
19	4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	4.00	0.00	2.00	14.00

Ignoring the diagonal data which indicates the correct classification. The maximum value is 107 when the predicted label is 15 but the real label is 19.

Q2

2.1 Below is the optimizer test graph.



2.3 For beta = 0:

Training Loss = 0.37 Test Loss = 0.40

Training Accuracy = 0.91 Test Accuracy = 0.91

For beta = 0.1:

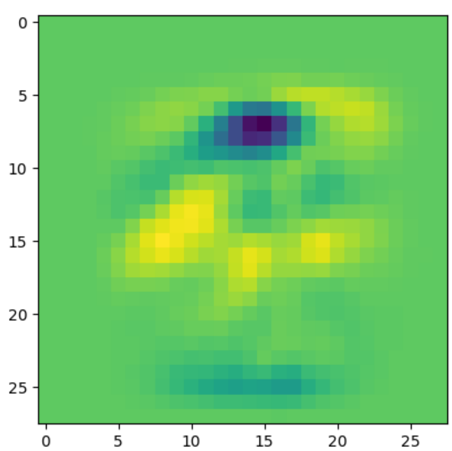
Training Loss = 0.36 Test Loss = 0.34

Training Accuracy = 0.90 Test Accuracy = 0.90

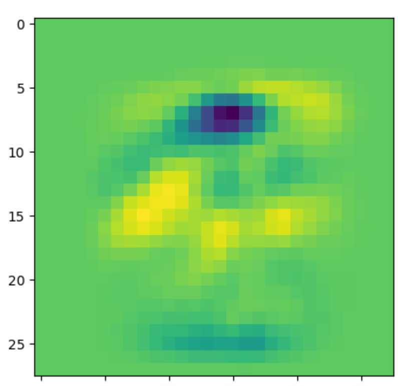
```
In [55]: runfile('/Users/wukairui/Documents/Courses UoT/CSC411/
HW3/q2.py', wdir='/Users/wukairui/Documents/Courses UoT/CSC411/
HW3')
Data Loaded
Train size: 11025
Test size: 2757
-------------------------------
The average hinge loss of train data with momentum of 0 is
0.396922546385
The average hinge loss of train data with momentum of 0.1 is
0.35614374504
The average hinge loss of test data with momentum of 0 is
0.400217682259
The average hinge loss of test data with momentum of 0.1 is
0.344133252814

The classification accuracy on the training set with momentum
of 0 is 0.913832199546
The classification accuracy on the training set with momentum
of 0.1 is 0.903945578231
The classification accuracy on the test set with momentum of 0
is 0.914762422923
The classification accuracy on the test set with momentum of
0.1 is 0.903155603917
```

W plot

1. Beta = 0



2. Beta = 0.1



3

3.1 $Kx = \lambda x$ (λ is Eigenvalues, X is Eigen vectors)

$x^T K x = x^T \lambda x = \lambda x^T x$

$x^T K x = \lambda x^T x$

for $x^T x = |x|^2$ and x is non-zero

we have $x^T x = |x|^2 > 0$

So, If $x^T K x > 0$

we have $\lambda > 0$

So, we have

① K is a symmetric matrix

② λ of K > 0

we get K is positive semidefinite

3.2

1. $K_{ij} = k(x^{(i)}, x^{(j)}) = \alpha$

since ① $K_{ij} > 0$ for all i, j

$\left\{\begin{array}{l} \text{eigenvalues of } K_{ij} > 0 \\ ② K_{ij} \text{ is symmetric} \end{array}\right.$

We get K_{ij} is positive semi-definite

Thus, $k(x, y) = \alpha$ is a kernel

3.2

2. Because $f: R^d \to R$

We get $f(x) \cdot f(y) = \langle f(x), f(y) \rangle$

Thus, $k(x,y) = \langle \phi(x), \phi(y) \rangle$ where $\phi(x) = f(x)$

we get $k(x,y) = f(x) \cdot f(y)$ is a kernel

3. $K_{ij} = a \cdot K_{1ij} + b \cdot K_{2ij}$

$$X^T K_{ij} X = a \cdot \underline{X^T K_{1ij} X} + b \cdot \underline{X^T K_{2ij} X} \qquad (a, b > 0)$$

We have, $X^T K_{1ij} X > 0$ and $X^T K_{2ij} X > 0$

Thus $X^T K_{ij} X > 0$

$\longrightarrow$ K_{ij} is positive semidefine

$\longrightarrow$ $k(x,y)$ is a kernel

4. $$\frac{k_1(x,y)}{\sqrt{k_1(x,x)}\sqrt{k_1(y,y)}} = \frac{\langle \phi(x), \phi(y) \rangle}{\sqrt{\langle \phi(x), \phi(x) \rangle}\sqrt{\langle \phi(y), \phi(y) \rangle}}$$

$$= \frac{\langle \phi(x), \phi(y) \rangle}{\|\phi(x)\| \, \|\phi(y)\|}$$

$$= \left\langle \frac{\phi(x)}{\|\phi(x)\|}, \frac{\phi(y)}{\|\phi(y)\|} \right\rangle$$

We get $k(x,y) = \langle \phi'(x), \phi'(y) \rangle$

where $\phi'(x) = \dfrac{\phi(x)}{\|\phi(x)\|}$ $\quad \phi'(y) = \dfrac{\phi(y)}{\|\phi(y)\|}$