CSC2515： Assignment #3
Due: 10pm, 6th December

YIFAN ZHANG

Student Number: 1004245952

December 5, 2017

Problem 1: 20 Newsgroups predictions

In this question, I choose to work with tf-idf feature representation.
There are 11314 train data points and 101631 feature dimensions in this dataset.
The most common word in training set is “the”.

1.1 Four models implemented on the dataset:

1) **Bernoulli Naïve Bayes (Baseline):** parameters as default.

2) **Multinomial Naïve Bayes:**
We tune the hyperparameters alpha of Multinomial Naïve Bayes with 5fold cross-validation in order to find the best accuracy. Alpha is a additive (Laplace/Lidstone) smoothing parameter.
When alpha = 0.2, the model has the best accuracy.

Hyperparameter: alpha	*Test accuracy(5fold cross-validation)*
0	0.13 (+/- 0.01)
0.2	0.72 (+/- 0.01)
0.4	0.69 (+/- 0.02)
0.6	0.67 (+/- 0.02)
0.8	0.65 (+/- 0.02)
1.0	0.64 (+/- 0.02)

(Other parameters: fit_prior=True, class_prior=None)

3) **Logistic Regression:**
We tune the hyperparameters C with 5fold cross-validation. C is inverse of regularization strength, smaller values specify stronger regularization
When alpha = 1.0, the model has the best accuracy.

Hyperparameter: alpha	*Test accuracy(5fold cross-validation)*
0.2	0.65 (+/- 0.02)
0.4	0.69 (+/- 0.02)
0.6	0.71 (+/- 0.02)
0.8	0.72 (+/- 0.02)
1.0	0.72 (+/- 0.01)

(Other parameters: penalty='l2', dual=False, tol=0.0001, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1)

4) **SVM:**

We tune the hyperparameters of SVM in order to find the best accuracy. We use cross validation (split in 10 fold) to tune alpha of SVM. Alpha is a Constant that multiplies the regularization term.
When alpha = 1e-4, the performance of SVM is the best.

Hyperparameter: alpha	*Train accuracy*	*Teat accuracy*
1e-03	0.915827	0.73590408
1e-04	0.95731934	0.76462948
1e-05	0.97316992	0.73590517
1e-06	0.97306188	0.71416021
1e-07	0.97315026	0.71142224

(Other parameters: loss='hinge', penalty='l2', l1_ratio=0.15, fit_intercept=True, max_iter=None, tol=None, shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=42, learning_rate='optimal', eta0=0.0, power_t=0.5, class_weight=None, warm_start=False, average=False, n_iter=5)

5) In conclusion, the final train and test accuracy for each model is showed as following:

	Train accuracy	*Test accuracy*	*5-folds cross-validation*
Bernoulli NB	0.5987272405868835	0.4579129049389272	0.47(+/-0.03)
Multinomial NB	0.9051617465087503	0.6712692511949018	0.72 (+/- 0.01)
LR	0.8957044369807319	0.6775092936802974	0.72(+/-0.01)
SVM	0.9537740852041718	0.6938396176314392	0.76(+/-0.02)

1.2 Explain why you picked these 3 methods, did they work as you thought? Why/Why not?

Multinomial NB: we can take the idea of Bernoulli NB one step further and instead of "word occurs in the document" we have "count how often word occurs in the document", which could improve the accuracy of prediction. In fact, Multinomial NB did show a better performance than Bernoulli NB.

Logistic Regression: this model is the most familiar to me, which is robust to small noise and has a low variance. Since it can't handle large number of categorical features/variables well, it has a worse performance than SVM on this dataset.

SVM: High accuracy, nice theoretical guarantees regarding overfitting, and especially popular in text classification problems where very high-dimensional spaces are the norm. SVM showed the best performance in this question which is as my prediction.

1.3 Confusion Matrix

It's obvious that SVM has the best accuracy among those four models. The confusion matrix of SVM is showed as following, which is a k*k matrix where is the number of test examples belonging to class j that were classified as i.

```
[[ 158.   2.   1.   2.   0.   1.   4.   5.  13.   3.   1.   8.
    6.  11.  11.  52.   7.  13.   4.  17.]
 [   6. 276.  18.  15.   6.  16.   5.   2.   8.   4.   0.  10.
    7.   3.  10.   2.   0.   1.   0.   0.]
 [   6.  22. 234.  39.  13.  14.   4.   2.  17.   4.   1.   7.
    1.   6.  11.   1.   3.   1.   6.   2.]
 [   0.  12.  30. 272.  21.   5.  11.   2.   9.   0.   1.   5.
   20.   0.   1.   0.   0.   0.   2.   1.]
 [   2.   4.   9.  34. 267.   7.  10.   6.  20.   0.   1.   5.
    9.   0.   6.   1.   3.   0.   1.   0.]
 [   0.  45.  32.   7.   6. 278.   4.   0.   6.   2.   0.   3.
    4.   0.   6.   1.   0.   1.   0.   0.]
 [   0.   2.   1.  13.  12.   0. 317.   6.  13.   2.   3.   1.
    9.   0.   3.   1.   5.   1.   1.   0.]
 [   7.   0.   2.   5.   1.   1.  12. 276.  40.   7.   2.   1.
   15.   2.   7.   1.   7.   5.   3.   2.]
 [   3.   3.   0.   2.   1.   1.   7.  16. 318.   4.   4.   2.
    7.   6.   6.   4.   6.   1.   5.   2.]
 [   4.   1.   0.   1.   1.   1.   9.   2.  21. 316.  24.   0.
    1.   3.   1.   5.   1.   0.   6.   0.]
 [   2.   2.   1.   1.   1.   0.   0.   3.   9.  10. 356.   1.
    1.   2.   4.   2.   2.   1.   0.   1.]
 [   3.   4.   6.   3.   4.   5.   5.   2.  20.   3.   3. 295.
    5.   3.   5.   5.  13.   4.   7.   1.]
 [   2.  13.   6.  34.  10.   7.  17.  10.  21.   6.   3.  22.
  211.  13.  11.   1.   2.   2.   2.   0.]
 [   8.   9.   2.   1.   1.   0.   2.   2.  16.   2.   6.   2.
    6. 309.   6.   8.   4.   5.   6.   1.]
 [   6.  10.   4.   1.   2.   1.   3.   6.  21.   1.   3.   3.
    9.  10. 297.   4.   6.   2.   5.   0.]
 [  23.   2.   2.   1.   0.   2.   1.   0.  15.   3.   2.   1.
    1.   7.   4. 322.   0.   1.   2.   9.]
 [   3.   2.   5.   4.   1.   1.   3.   3.  16.   4.   0.  10.
    2.   6.   6.   8. 253.   9.  21.   7.]
 [  26.   1.   2.   3.   1.   1.   1.   2.  12.   4.   1.   7.
    3.   2.   2.  10.   8. 280.  10.   0.]
```

```
[ 16.   0.   0.   4.   2.   0.   0.   5.  13.   2.   5.   6.
   1.   4.  12.   4.  93.   4. 135.   4.]
[ 38.   3.   3.   2.   0.   1.   2.   4.   9.   1.   3.   2.
   1.  11.   6.  70.  24.   8.   7.  56.]]
```

1.4 What were the two classes your classifiers was most confused about?

Using *metrics.classification_report* on SVM prediction:

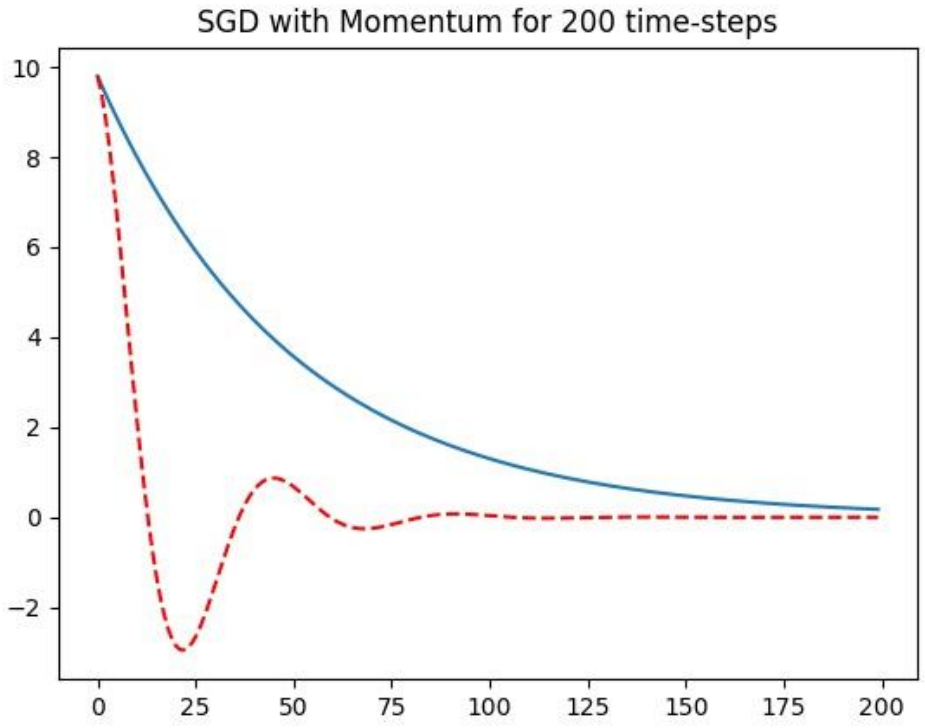
	precision	recall	f1-score	support
1	0.50	0.50	0.50	319
2	0.67	0.71	0.69	389
3	0.65	0.59	0.62	394
4	0.61	0.69	0.65	392
5	0.76	0.69	0.73	385
6	0.81	0.70	0.75	395
7	0.76	0.81	0.79	390
8	0.78	0.70	0.74	396
9	0.52	0.80	0.63	398
10	0.84	0.80	0.82	397
11	0.85	0.89	0.87	399
12	0.75	0.74	0.75	396
13	0.66	0.54	0.59	393
14	0.78	0.78	0.78	396
15	0.72	0.75	0.73	394
16	0.64	0.81	0.72	398
17	0.58	0.70	0.63	364
18	0.83	0.74	0.78	376
19	0.61	0.44	0.51	310
20	0.54	0.22	0.32	251
Avg / total	0.70	0.69	0.69	7532

In the report above, we could see that when test examples belonging to class 1(alt.atheism) and class 20(talk.religion.misc) have the lowest F1 scores. In this sense, class 1 and class 20 were the most confused class.

Problem 2: Training SVM with SGD

2.1 SDG with momentum

Find the minimum of $f(w) = 0.01w^2$ using gradient descent. The blue line denotes w_t the using β=0.0 and red line denotes the w_t using β=0.9.

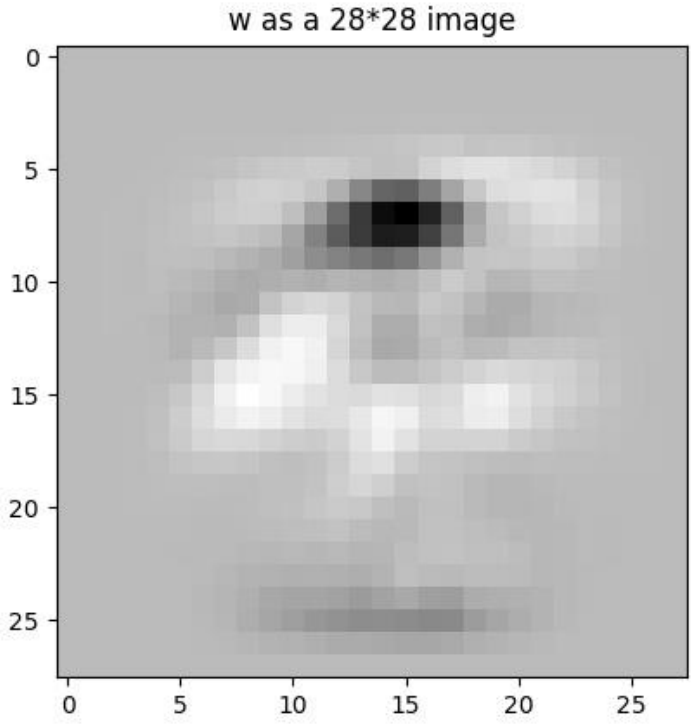


2.3 Apply on 4-vs-9 digits on MNIST

$\alpha = 0.05$, a penalty of C=1.0, minibatch sizes of $m = 100$, and $T = 500$ total iterations.

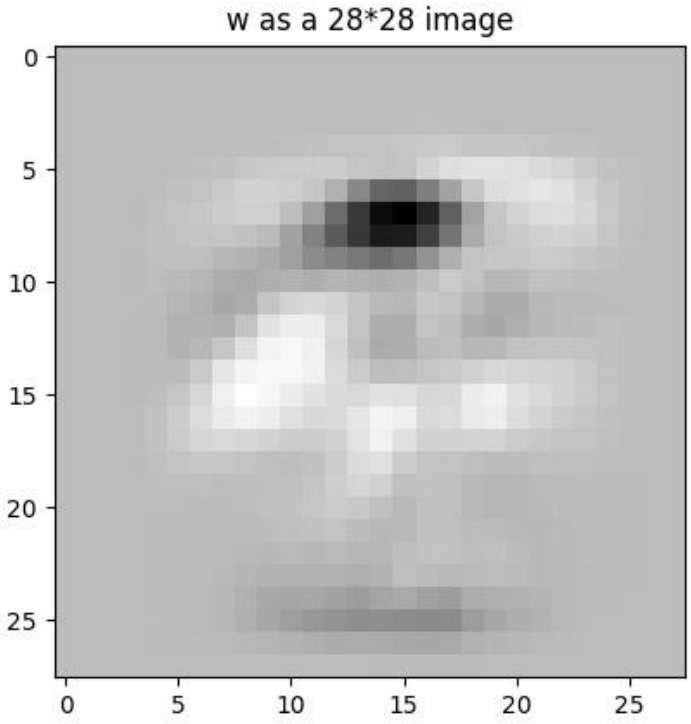
1) β=0:

Training loss = 0.593255121335
Test loss = 0.596883466246
Train accuracy = 0.9121995464852608
Test accuracy = 0.9125861443598113



2) β=0.1:

Training loss = 0.564242212128
Test loss = 0.564848859346
Train accuracy = 0.9254421768707483
Test accuracy = 0.9249183895538629


w as a 28*28 image
0
5
10
15
20
25
0
5
10
15
20
25

Problem 3: Kernels

3.1 Positive semidefinite and quadratic form

1. Prove that a symmetric matrix $K \in \mathbb{R}^{d\times d}$ is positive semidefinite iff for all vectors $x \in \mathbb{R}^d$ we have $\mathbf{x}^T\mathbf{K}\mathbf{x} \geq 0$.

Suppose $u_1, u_2, \ldots u_n$ be the linearly independent eigenvectors which correspond to the eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_n$ of the real symmetric matrix K .

Since K is positive semidefinite, we have $\lambda_1, \lambda_2, \ldots \lambda_n$ are positive.

Also, K is a real symmetric matrix, it can be rewrite as following:

$$K = P \cdot D \cdot P^{-1} = P \cdot D \cdot P^T$$

where the columns of P contain the right hand eigenvectors of matrix and P contain the left hand eigenvectors as its row. Thus, if u_i 's are the right hand eigenvectors, then u_j 's are the left hand eigenvectors of K . Then, it holds:

$$u_i^T \cdot u_j = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$$

Let $x = c_1u_1 + c_2u_2 + \ldots + c_nu_n$ be a random $n \times 1$ real vector.

$$\mathbf{x}^T\mathbf{K}\mathbf{x} = (c_1u_1^T + c_2u_2^T + \ldots + c_nu_n^T)(\mathrm{P} \cdot \mathrm{D} \cdot \mathrm{P}^{-1})(c_1u_1 + c_2u_2 + \ldots + c_nu_n)$$

$$= (c_1\|u_1\|_2^2 \quad c_2\|u_2\|_2^2 \quad \ldots \quad c_n\|u_n\|_2^2) \cdot \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \lambda_n \end{pmatrix} \cdot \begin{pmatrix} c_1\|u_1\|_2^2 \\ c_2\|u_2\|_2^2 \\ \ldots \\ c_n\|u_n\|_2^2 \end{pmatrix}$$

$$= \lambda_1c_1^2 + \lambda_2c_2^2 + \ldots + \lambda_nc_n^2 \geq 0$$

3.2 Kernel properties

Prove the following properties:

1. The function $k(\mathbf{x}, \mathbf{y}) = \alpha$ is a kernel for $\alpha > 0$

Suppose that: $\phi(x)=\sqrt{\alpha}$

Then: $k(\mathbf{x},\mathbf{y})=\langle\phi(x),\phi(y)\rangle=\alpha$

2. $k(\mathbf{x},\mathbf{y})=f(\mathbf{x})\cdot f(\mathbf{y})$ is a kernel for all $f:\mathbb{R}^d\to\mathbb{R}$.

Suppose that: $\phi(x)=f(x)$

Since there is just one feature defined by $f()$: $k(\mathbf{x},\mathbf{y})=\langle\phi(x),\phi(y)\rangle=f(x)\cdot f(y)$

3. If $k_1(\mathbf{x},\mathbf{y})$ and $k_2(\mathbf{x},\mathbf{y})$ are kernels then $k(\mathbf{x},\mathbf{y})=\alpha k_1(\mathbf{x},\mathbf{y})+\beta k_2(\mathbf{x},\mathbf{y})$ for $a,b>0$ is a kernel.

Suppose that:

$k_1(\mathbf{x},\mathbf{y})=\langle\phi^{(1)}(x),\phi^{(1)}(y)\rangle$

$k_2(\mathbf{x},\mathbf{y})=\langle\phi^{(2)}(x),\phi^{(2)}(y)\rangle$

$\phi(x)=[\sqrt{a}\phi^{(1)}(x),\sqrt{b}\phi^{(2)}(x)]$

Then:

$k(\mathbf{x},\mathbf{y})=\langle\phi(x),\phi(y)\rangle$

$=a\langle\phi^{(1)}(x),\phi^{(1)}(y)\rangle+b\langle\phi^{(2)}(x),\phi^{(2)}(y)\rangle$

$=\alpha k_1(\mathbf{x},\mathbf{y})+\beta k_2(\mathbf{x},\mathbf{y})$

4. If $k_1(\mathbf{x},\mathbf{y})$ is a kernel then $k(\mathbf{x},\mathbf{y})=\dfrac{k_1(\mathbf{x},\mathbf{y})}{\sqrt{k_1(\mathbf{x},\mathbf{x})}\sqrt{k_1(\mathbf{y},\mathbf{y})}}$ is a kernel(hint: use the features ϕ such that $k(\mathbf{x},\mathbf{y})=\langle\phi(x),\phi(y)\rangle$)

Suppose that:

$k_1(\mathbf{x},\mathbf{y})=\langle\phi^{(1)}(x),\phi^{(1)}(y)\rangle$

$\phi(x)=\dfrac{\phi^{(1)}(x)}{\left\|\phi^{(1)}(x)\right\|}$

Then:

$$k(\mathbf{x},\mathbf{y}) = \langle \phi(x), \phi(y) \rangle = \left\langle \frac{\phi^{(1)}(x)}{\left\|\phi^{(1)}(x)\right\|}, \frac{\phi^{(1)}(y)}{\left\|\phi^{(1)}(y)\right\|} \right\rangle$$

$$= \frac{k_1(\mathbf{x},\mathbf{y})}{\sqrt{k_1(\mathbf{x},\mathbf{x})}\sqrt{k_1(\mathbf{y},\mathbf{y})}}$$