CSC2515 Assignment 3

Tianxiang Chen 999473181

I. 20 Newsgroups predictions

For this problem, I tried many algorithms such as KNN, Decision Tree, Logistic Regression, RNN, SVM, multinomialNB, Neural Network.

For algorithm selection, I firstly just call each one mentioned above (please see q1.py) and check its training and test accuracy. It showed KNN, RNN and Decision Tree did not work well with this problem (test accuracy less than 0.5). Then, for the rest four, I wrote the cross-validation to pick the hyperparameter and check its test accuracy with the hyperparameter chosen from training set cross validation.

Eventually, I picked Logistic Regression, SVM and Neural Network(MLP Classifier) as the three algorithms to report. For this problem, we had about 10,000 input data but 100,000 features. So we need to try to avoid over-fitting. To achieve that, a linear model is preferred due to its simplicity. For example, for SVM, a more more complicated kernel will probably cause some over-fitting issues. Thus, the final model selection met my expectation.

Three models' accuracy along with the BernoulliNB are reported in the table below (the result here is with the hyperparameter tuned, which will be discussed later):

	Training Set Accuracy	Test Set Accuracy
BernoulliNB	0.5987	0.4579
Logistic Regression	0.9695	0.6891
Linear SVM	0.9725	0.6920
Neural Network	0.9747	0.7011

For hyperparameter, I chose one from each algorithm for tuning, which are:

- C, the inverse of regularization strength in Logistic Regression
- C, the penalty parameter C of the error term in SVM
- Number of layers in Neural Network

I used the code written for A1 for cross-validation and set K=10 for Logistic and SVM cases and swept C from 1 to 10 linearly with a step size equals 1. For Neural Network, since the computation is really slow, I only chose two candidates $\{50,100\}$ for the number of layers and set K=5 for a faster runtime. Still, the code took a long time for getting all the result.

The final hyperparameter is selected as:

C=9 for Logistic Regression
C=1 for SVM
Number of Layers = 100

Fig. 1 shows the Confusion Matrix for the Neural Network approach with 100 layers. In order to show the classification correctness clearly, I plotted the figure in gray-r mode. So, the darker the color is, the more predictions fall in this region.

For finding the most confusing two class, I added the transpose matrix to the origin matrix itself, since we want to find the most wrongly classified two class ($A \rightarrow B$ and $B \rightarrow A$ need to be sum up). For the new matrix, the maximum non-diagonal element is the one we are looking for. In this problem, the most confusing two classes are:

$$class17 : talk.politics.guns \text{ and } class19 : talk.politics.misc$$

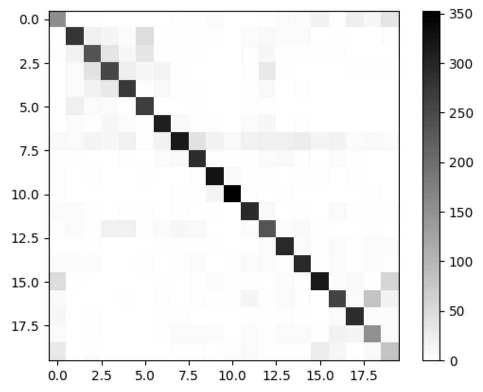

Fig. 1: Confusion Matrix for Neural Network with 100 Layers, in gray-r mode)

II. Training SVM with SGD

In this part, only the questions asked in the handout are answered. For details about the code implemented, please checked the q2.py file uploaded.

2.1 SGD With Momentum

In Fig. the two curves represent w_t for $\beta = 0.0$ and $\beta = 0.9$, respectively.

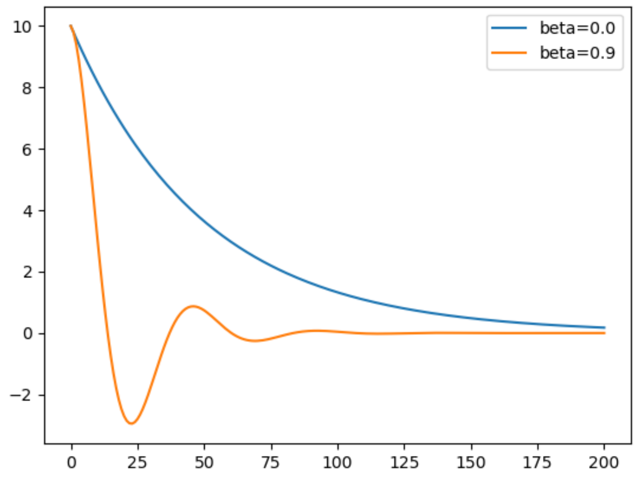

Fig. 2: w_t for $\beta = 0.0$ and $\beta = 0.9$

2.2 Training SVM

This part purely asks to write code for implementing the SVM training. So please check q2.py for detail.

2.3 Apply on 4-vs-9 digits on MNIST

The result is shown in the table below.

	$\beta = 0.0$	$\beta = 0.1$
The training loss (avg)	0.4035	0.3574
The test loss (avg)	0.4071	0.3453
Training set accuracy	0.9122	0.8998
Test set accuracy	0.9126	0.8981

The w as a 28×28 image for $\beta = 0.0$ and $\beta = 0.1$ are plotted below.

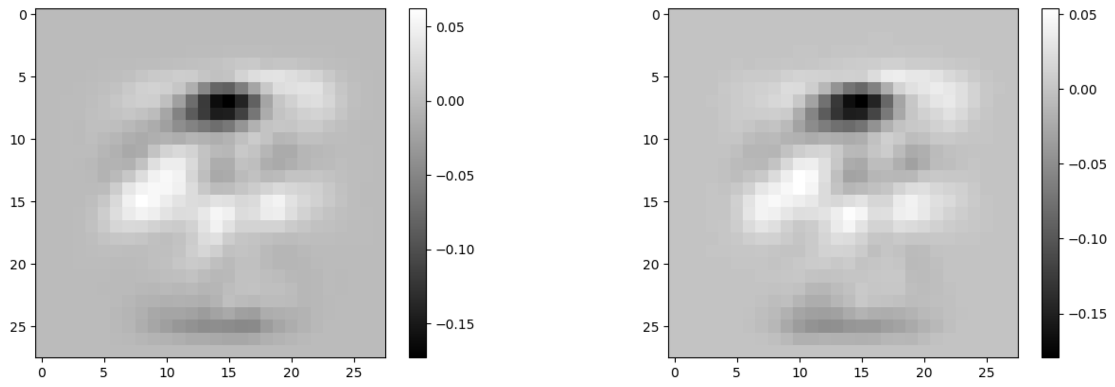

Fig. 3: w as a 28×28 image (left for $\beta = 0.0$, right for $\beta = 0.1$)

III. Kernels

3.1 Positive semidefinite and quadratic form

For this question, since it states ”iff”, we need to prove for both directions.

The proof for forward direction is:

Knowing matrix K $\in R^{d\times d}$ is symmetric and real, we can write $K = UDU^T$, where U is a real orthogonal matrix, and D is real and diagonal. Also, the entries on the diagonal of D are the eigenvalues of K which are non-negative, so we can write $D = C^2$ where C is also a diagonal matrix. Then we have $K = UCCU^T = UC(UC)^T$.

For every x $\in R^d$, we have $x^TKx = x^TUC(UC)^Tx = (x^TUC)(x^TUC)^T$

Since x $\in R^d$, U and C $\in R^{d\times d}$, x^TUC will be a $(1 \times d)$ matrix. So the equation above is a $(1 \times d)$ matrix multiple a $(d \times 1)$ matrix, which returns a scalar answer. Assuming the elements in the $(1 \times d)$ matrix is $\{a_0, a_1, ...a_d\}$, then multiplication can be expressed as: $a_0^2 + a_1^2 + ..a_d^2 \geq 0$

So, $(x^TUC)(x^TUC)^T \geq 0$.

Forward path proved.

Then, we proved the backward path.

We know to find the eigenvalue of K, we will have $Kx = \lambda x$, where λ and x are the eigenvalues and eigenvectors of K, respectively.

Multiple both side with x^T in the front, we get:

$x^TKx = x^T\lambda x = \lambda x^Tx$ Since λ is the eigenvalue (scalar).

We know $x^TKx \geq 0$, so $\lambda x^Tx \geq 0$. From the forward path, we know for such x $\in R^d$, $x^Tx \geq 0$, so λ is also greater or equal than 0 for all d eigenvalues, plus K is symmetric as given, the matrix K is positive semidefinite.

Backward path proved.

So, proved.

3.2 Kernel properties

1: Since function $k(x, y) = \alpha$ where $\alpha > 0$,The matrix K is a $(d \times d)$ matrix where all the elements equal to α.

It is obvious that matrix K is symmetric since all elements are α.

Also, it can be shown that the eigenvalues of this matrix will be $\lambda_1 = d \cdot \alpha$ and $\lambda_{2...d} = 0$

So, K is positive semidefinite. Proved.

2: Since f is a function: $R^d \to R$, f() just return a scalar value.

So, $k(x, y) = f(x)(y) =< \phi(x), \phi(y) >$ where $\phi(x) = f(x)$ and $\phi(y) = f(y)$

Proved.

3: Let K_1, K_2 be the Gram matrix associated with kernel function k_1 and k_2, respectively. The Gram matrix associated with $c_1k_1 + c_2k_2$ is just $K = c_1K_1 + c_2K_2$.

Since K1 and K2 are positive semidefinite ($v^TK_1v \geq 0$ and $v^TK_2v \geq 0$), we get:

$$v^T(c_1K_1 + c_2K_2)v = c_1(v^TK_1v) + c_2(v^TK_2v) \geq 0$$

Proved.

4:

$$\begin{aligned}
\frac{k_1(x,y)}{\sqrt{k_1(x,x)}\sqrt{k_1(y,y)}} &= \frac{< \phi^1(x), \phi^1(y) >}{\sqrt{(\phi_1^1(x) + ...\phi_1^N(x)) \cdot (\phi_1^1(x) + ...\phi_1^N(x))} \cdot \sqrt{(\phi_1^1(y) + ...\phi_1^N(y)) \cdot (\phi_1^1(y) + ...\phi_1^N(y))}} \\
&= \frac{(\phi_1^1(x) + ...\phi_1^N(x)) \cdot (\phi_1^1(y) + ...\phi_1^N(y))}{\sqrt{\phi_1^1(x)^2 + ...\phi_1^N(x)^2} \cdot \sqrt{\phi_1^1(y)^2 + ...\phi_1^N(y)^2}} \\
&= \Big(\frac{\phi_1^1(x)}{||\phi^1(x)||} + ...\frac{\phi_N^1(x)}{||\phi^1(x)||}\Big) \cdot \Big(\frac{\phi_1^1(y)}{||\phi^1(y)||} + ...\frac{\phi_N^1(y)}{||\phi^1(y)||}\Big)
\end{aligned}$$

So, the two brackets can represent a new $\phi^{'}(x)$ and $\phi^{'}(y)$ and overall the result is just their inner product.

Proved.