Report on HW3

ZHANG Qianhao / 1004654377

1 20-Newsgroups predictions (6%)

> You need to report the results of three different algorithms and the baseline, each result reported should include train and test loss.

1. BernoulliNB baseline train accuracy = 0.5981085380943963
   BernoulliNB baseline test accuracy = 0.46587891662241104
2. RandomForest train accuracy = 0.9742796535266042
   RandomForest test accuracy = 0.6411311736590547
3. SVM train accuracy = 0.9482057627717871
   SVM test accuracy = 0.5452734997344663
4. Logistic train accuracy = 0.9211596252430617
   Logistic test accuracy = 0.6428571428571429

> Explain in your report how you picked the best hyperparameters.

By making use of sklearn's GridSearchCV, for a specific classifier, I can do a grid search for the hyper-parameters. For random forest, the size of the forest will need to be tuned to make sure the variance has been minimized. We know ensembling decision trees cannot help with the bias, so when the test accuracy does not significantly increase with the size of forest, we can stop. For SVM, the grid searches over different kinds of kernels and different values of penalty coefficient. The accuracy in practice depends heavily on the type of kernel, and linear kernel outperforms the other 2.
For logistic regression, the grid searches over the type of penalty and the value of penalty coefficient. It turns out that logistic regression has the highest test accuracy even though without the best training accuracy.

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

1. One intuition is that texts belonging to different news groups should have different keywords, where 'keywords' are some words that mostly appear in a specific news context, such as 'soccer' to sports, 'bond' to finance. These words can be strong indicators about the class of the text.
   This naturally relates to the split of a decision tree. Given that there are 20 classes, and each class could have many keywords, I used a random forest with a large amount of trees.
   And it works decently in the training set (97% accuracy).
   But it surprises me that random forest, even with 1200 trees, only gets a mediocre performance (64% accuracy) in the test set. And it does not help much to increase the forest size ever since 200.
   One possible explanation is that the dimensionality of feature is still significantly larger than the number of training cases available in this example. This could result in many unseen indicators appearing in the test set, and thus confuses the model.

2. Another intuition is that some words, though not exclusively used in one class, may have significantly more occurrences in some classes. Such as 'investigation' for President-Trump-related topics. Several such words combined would make a good boundary for some news groups.
   To make the distinction even better, I implemented the function 'preprocess' to remove puncutations and stop words and stemmize the words to reduce the noise from non-indicative but frequent words such as 'the', and make different forms for the same word such as 'investigate' and 'investigation' count together.
   SVM and logistic regression immediately pops out as good models to fit the boundary. As it turns out, logistic regression does indeed have the best performance.
   One thing I did not expect is that, since I was expecting the boundary to be not very linear, the best kernel for SVM is actually linear and logistic regression could outperform random forest (especially when it takes much less time to train!).
   My guess is that the distinction between classes can only depend on a linear combination of a few words' frequency, thus linear bounary is good enough.

> For your best classifier compute and show the confusion matrix. What were the two classes your classifier was most confused about?

As seen in the matrix, the model has the worst accuracy on class 18 and class 19. They are both often misclassified as class 8 (almost 50% chance). But for pairwise confusion, class 7 and class 8 are most likely to be confused with each other.

2 Training SVM with SGD

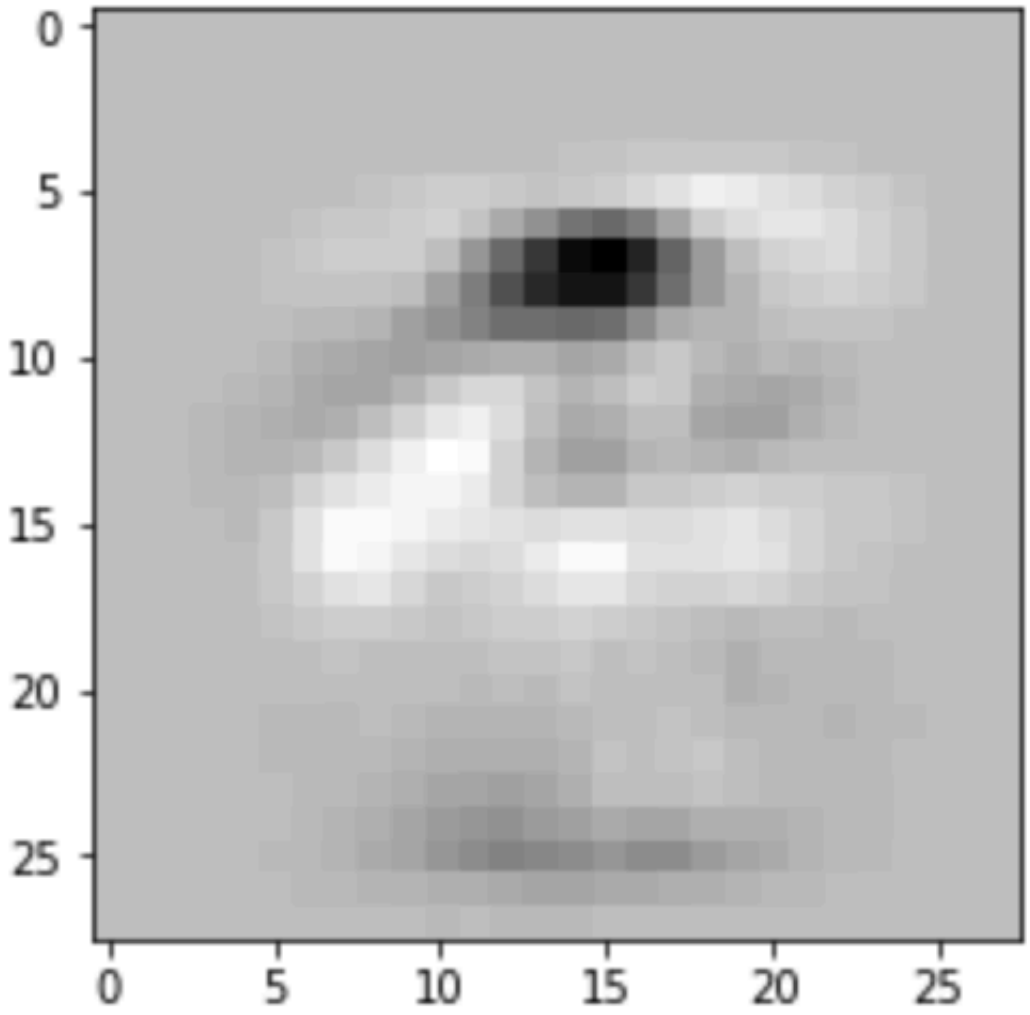
> Verify your implementation, find the minimum of $f(w) = 0.01w^2$ using gradient descent.

lr = 1.0, beta = 0.0, w0 = 10.0, steps = 200, final w: 0.17587946605721566

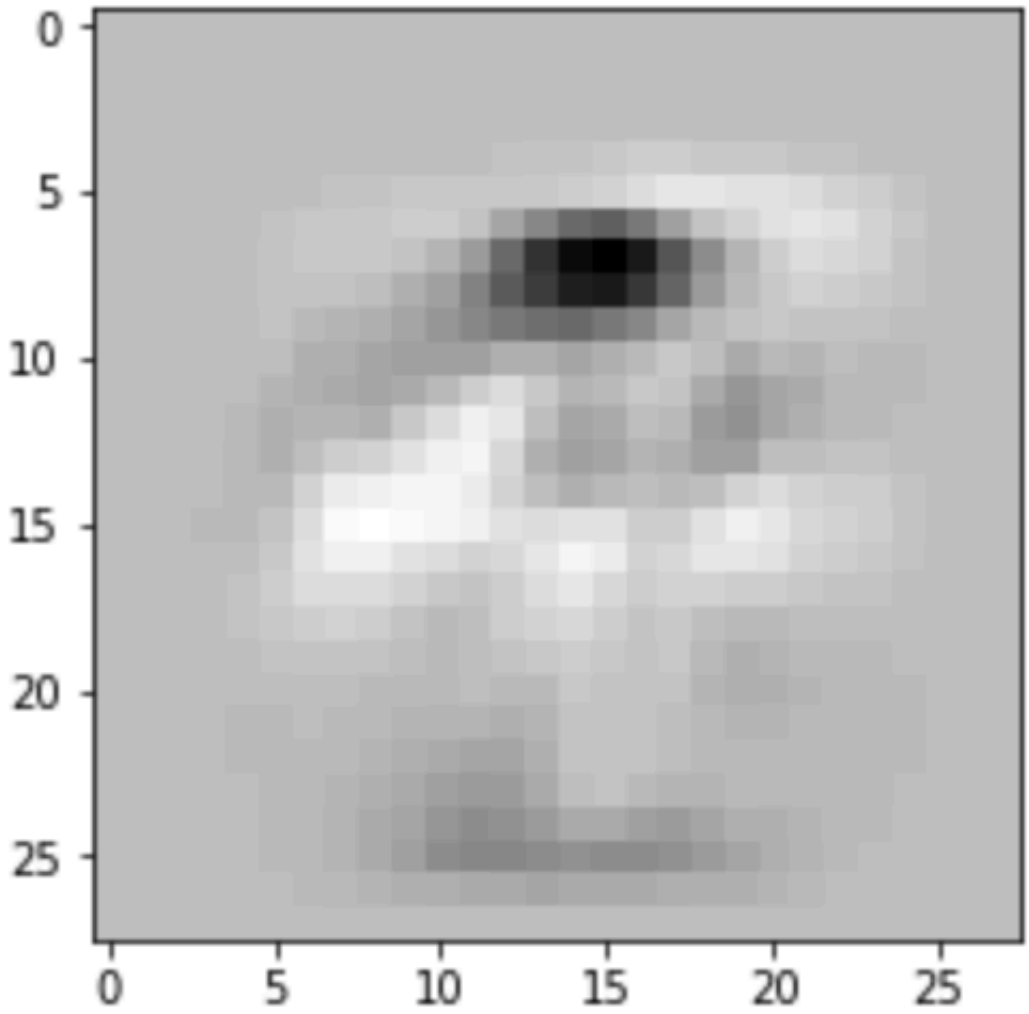
lr = 1.0, beta = 0.9, w0 = 10.0, steps = 200, final w: -2.0665349231570765e-05

> Train two SVM models using gradient descent with a learning rate of 0.05, a penalty of 1.0, minibatch sizes of 100, and 500 total iterations. For the first model use momentum 0 and for the second use momentum 0.1 For both of the trained models report the following: The training loss. The test loss. The classification accuracy on the training set. The classification accuracy on the test set. Plot w as a 28×28 image.

alpha = 0.05, C = 1.0, m = 100, T = 500, beta = 0.0 training accuracy: 0.9172789115646258, test accuracy: 0.9133115705476967 training loss: 0.30070056132858464, test loss: 0.30457613737347705



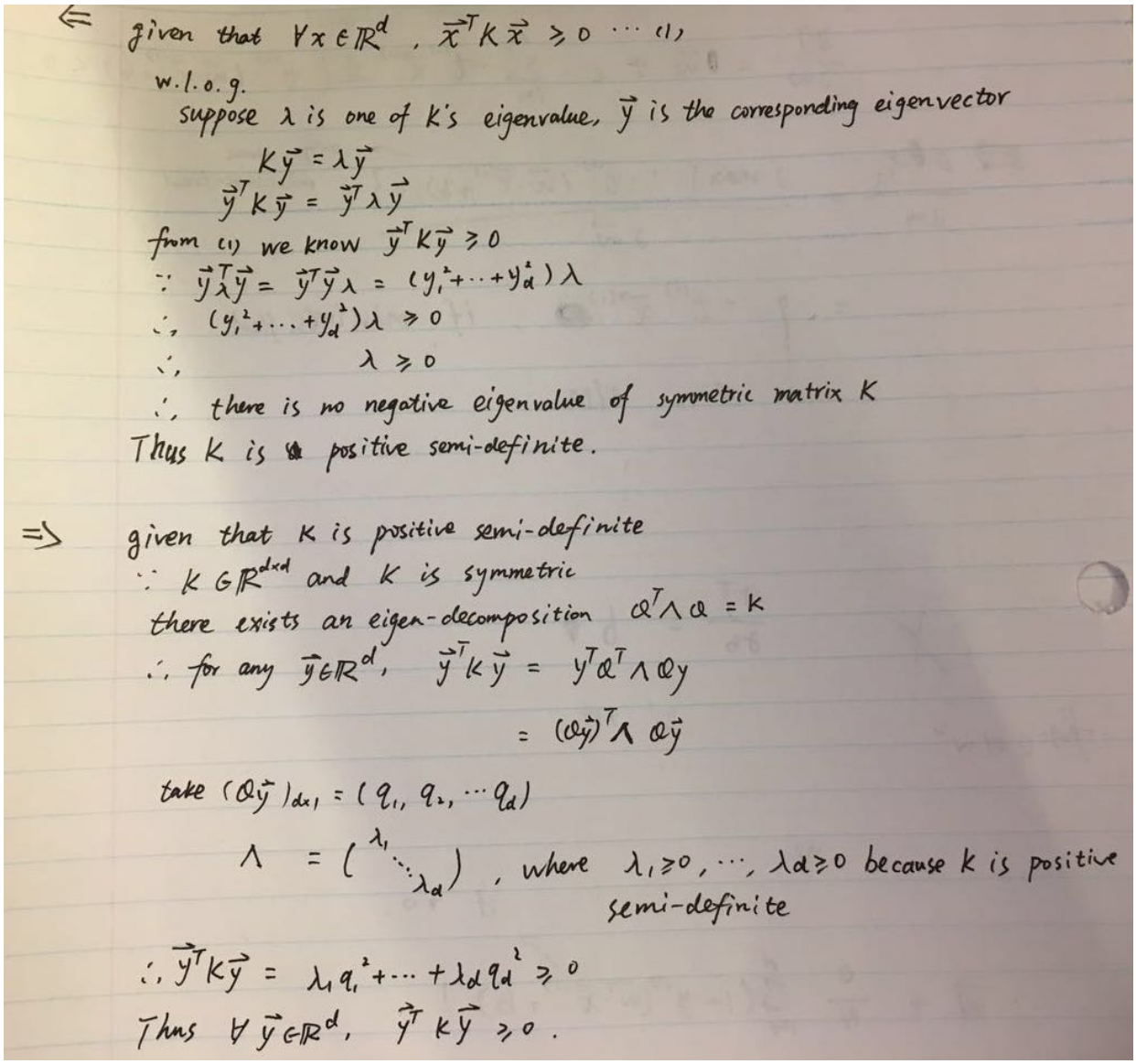
alpha = 0.05, C = 1.0, m = 100, T = 500, beta = 0.1 training accuracy: 0.927891156462585, test accuracy: 0.9220166848023214 training loss: 0.29247342549848304, test loss: 0.29536689916718345



3 Kernels

3.1 Positive semidefinite and quadratic form

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



3.2 Kernel properties

> Prove the following properties: The function $k(x, y) = \alpha$ is a kernel for $\alpha > 0$.
> $k(x, y) = f(x) \cdot f(y)$ is a kernel for all $f : R^d \to R$ If $k1(x, y)$ and $k2(x, y)$ are kernels then $k(x, y) = a \cdot k1(x, y) + b \cdot k2(x, y)$ for $a, b > 0$ is a kernel.

1. There exists φ: $\forall \vec{x} \in \mathbb{R}^d$, $\varphi(\vec{x}) = \sqrt{\alpha}$

s.t. $k(\vec{x}, \vec{y}) = \alpha = \langle \varphi(\vec{x}), \varphi(\vec{y}) \rangle$

Thus k is a kernel.

2. $\forall f: \mathbb{R}^d \to \mathbb{R}$

$k(\vec{x}, \vec{y}) = f(\vec{x}) \cdot f(\vec{y}) = \langle f(\vec{x}), f(\vec{y}) \rangle$

Thus k is a kernel.

3. for all $\vec{x}^{(1)}, \cdots, \vec{x}^{(n)}$

the gram matrix $K_{ij} = k(\vec{x}^{(i)}, \vec{x}^{(j)})$ $(i, j \in \{1, \cdots, n\})$

$= a \cdot k_1(\vec{x}^{(i)}, \vec{x}^{(j)}) + b \cdot k_2(\vec{x}^{(i)}, \vec{x}^{(j)})$

$\because$ k_1, k_2 are kernels, $a > 0$, $b > 0$

$\therefore$ $a \cdot k_1(\vec{x}^{(i)}, \vec{x}^{(j)})$ is positive semi-definite

$b \cdot k_2(\vec{x}^{(i)}, \vec{x}^{(j)})$ is positive semi-definite

$\therefore$ $k(\vec{x}^{(i)}, \vec{x}^{(j)})$ is positive semi-definite

Thus k is also a kernel.

4. If $k1(x, y)$ is a kernel then $k(x, y) = \frac{k1(x,y)}{\sqrt{k1(x,x)}\sqrt{k1(y,y)}}$ is a kernel (hint: use the features ϕ such that $k1(x, y) = \langle \phi(x), \phi(y) \rangle$).

4. Let φ be the embedding for kernel k_1

$$\therefore\ k(\vec{x},\vec{y}) = \frac{k_1(\vec{x},\vec{y})}{\sqrt{k_1(\vec{x},\vec{x})}\cdot\sqrt{k_1(\vec{y},\vec{y})}}$$

$$= \frac{\langle \varphi(\vec{x}), \varphi(\vec{y})\rangle}{\|\varphi(\vec{x})\|\cdot\|\varphi(\vec{y})\|}$$

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

thus $\phi(\vec{x}) = \frac{\varphi(\vec{x})}{\|\varphi(\vec{x})\|}$ is the embedding for k

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

Thus k is also a kernel.