Homework 3 KNN and SVM

Hsin-Yang Chen, Ying-Chuan Liao

1. KNN

   Preprocess:

   Since training KNN and SVM takes a lot of time. To fast up the training time, we first compress the data with LDA and shuffle them. Thus, for MNIST and COVERTYPE, the data has been compressed to 9 and 6 dimensions, respectively.

   Use cross validation to decide the best K. You are expected to do 2-fold, 5-fold and Leave-one-out.

   2-Fold:

   MNIST

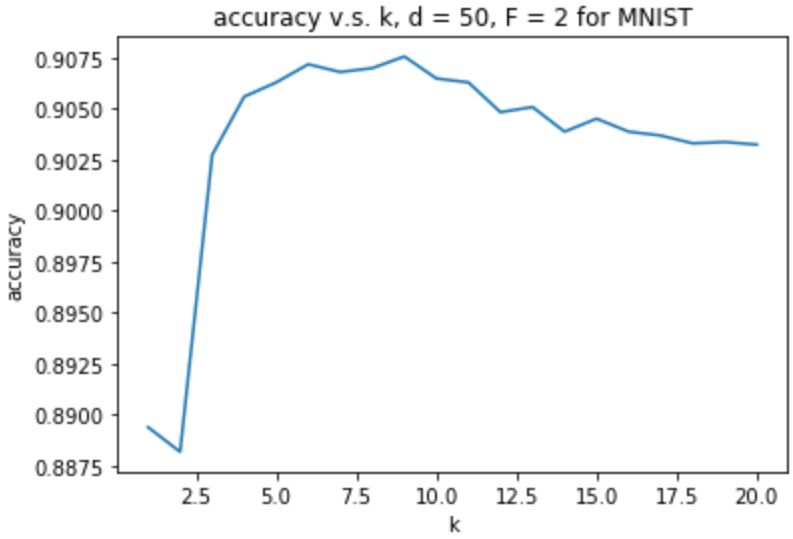


Figure 1-1 accuracy v.s. K, d = 50, F = 2 for MNIST

The figure shows that k = 8 gives the best accuracy around 0.907

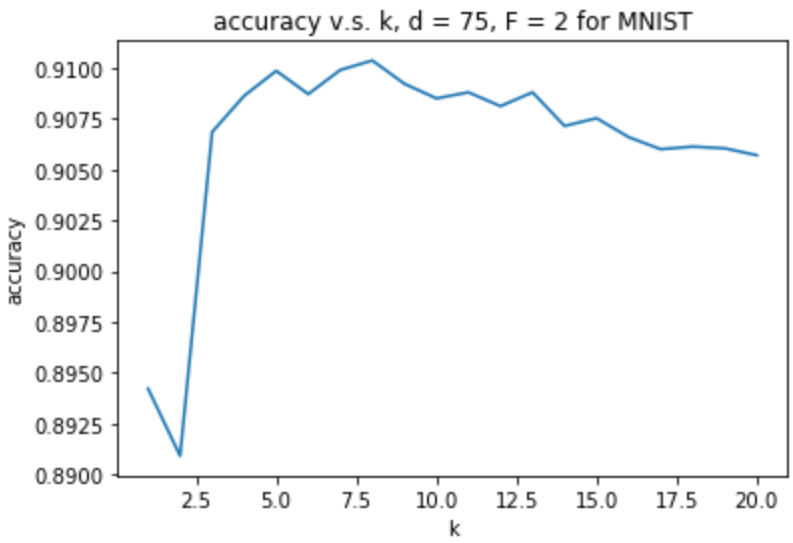


Figure 1-2 accuracy v.s. K, d = 75, F = 2 for MNIST

The figure shows that k = 8 gives the best accuracy around 0.911

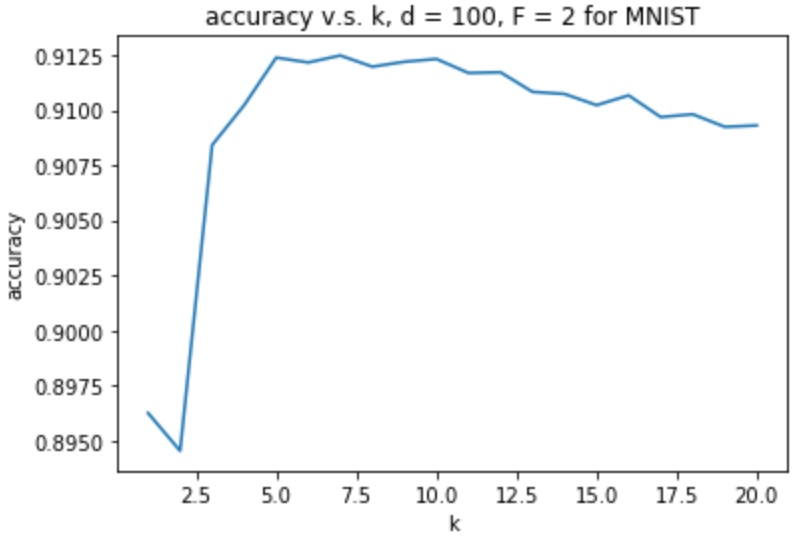


Figure 1-3 accuracy v.s. K, d = 100, F = 2 for MNIST

The figure shows that k = 7 gives the best accuracy around 0.912

COVERTYPE:

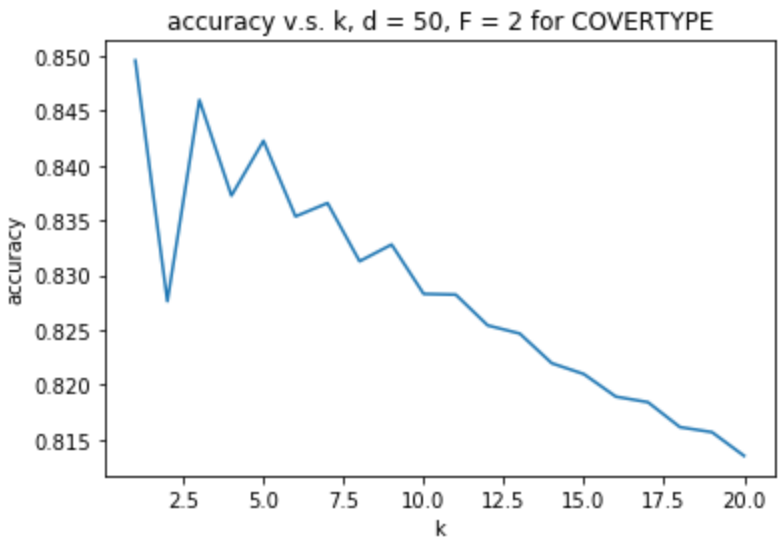


Figure 1-4 accuracy v.s. K, d = 50, F = 2 for COVERTYPE

The figure shows that k = 1 gives the best accuracy around 0.850

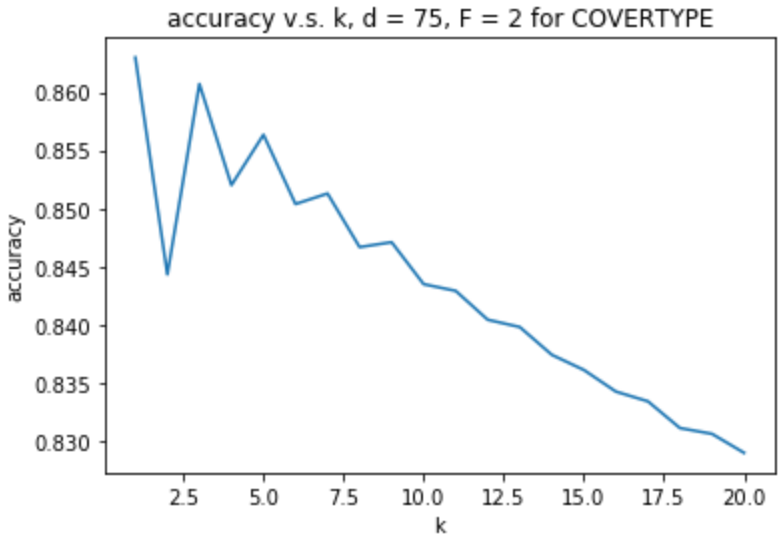


Figure 1-5 accuracy v.s. K, d = 75, F = 2 for COVERTYPE

The figure shows that k = 1 gives the best accuracy around 0.864

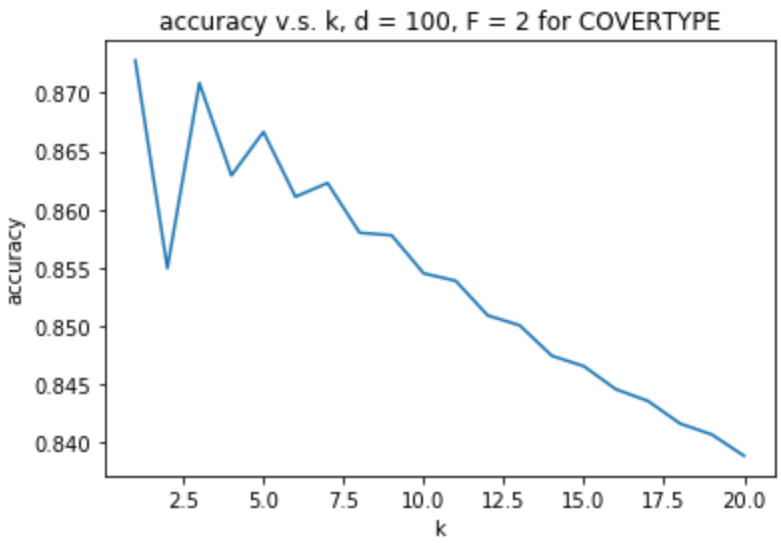


Figure 1-6 accuracy v.s. K, d = 100, F = 2 for COVERTYPE

The figure shows that k = 1 gives the best accuracy around 0.874

5-Fold:

MNIST

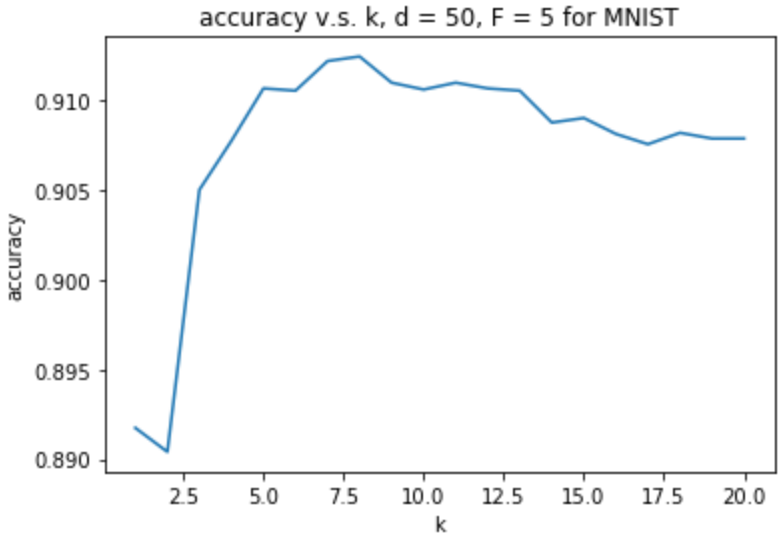


Figure 1-7 accuracy v.s. K, d = 50, F = 5 for MNIST

The figure shows that k = 8 gives the best accuracy around 0.913

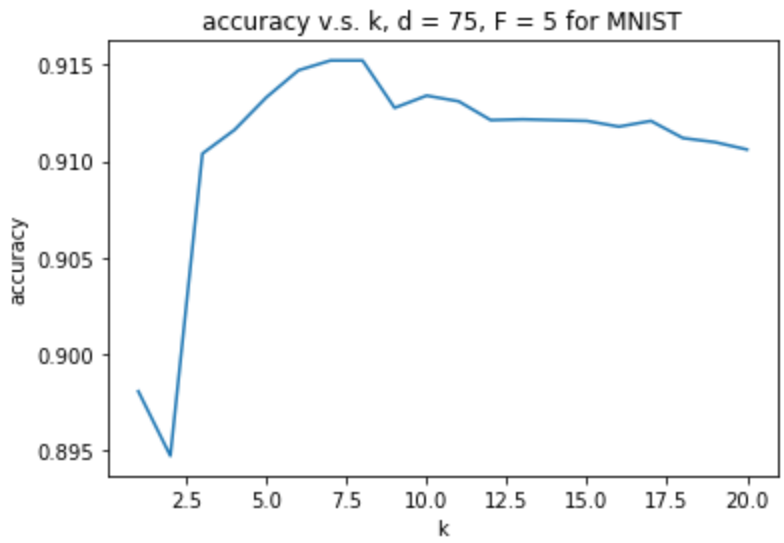


Figure 1-8 accuracy v.s. K, d = 75, F = 5 for MNIST

The figure shows that k = 8 gives the best accuracy around 0.914

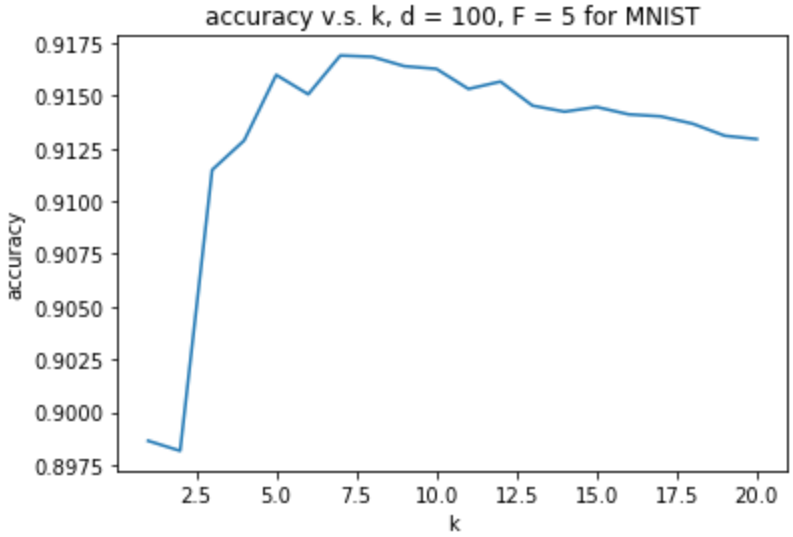


Figure 1-9 accuracy v.s. K, d = 100, F = 5 for MNIST

The figure shows that k = 8 gives the best accuracy around 0.917

COVERTYPE

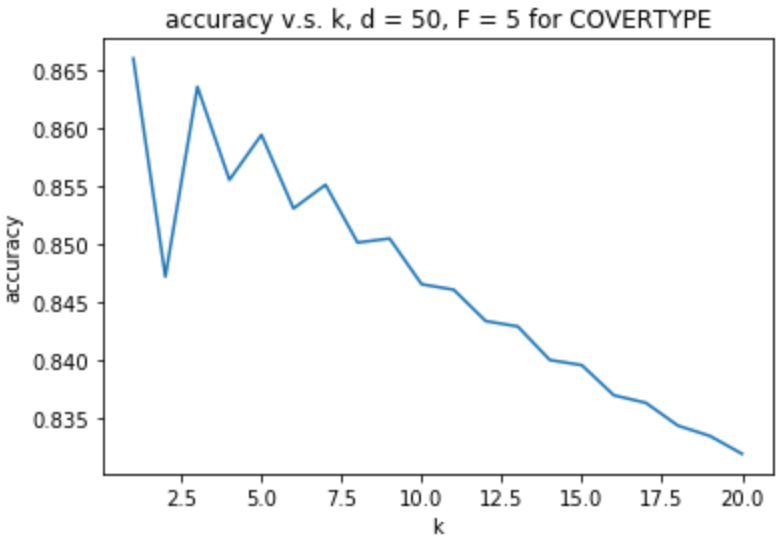


Figure 1-10 accuracy v.s. K, d = 50, F = 5 for COVERTYPE

The figure shows that k = 1 gives the best accuracy around 0.865

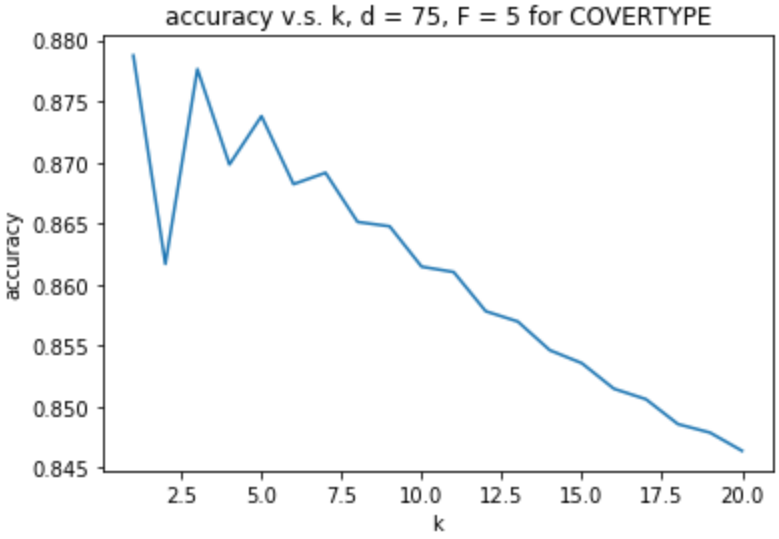


Figure 1-11 accuracy v.s. K, d = 75, F = 5 for COVERTYPE

The figure shows that k = 1 gives the best accuracy around 0.880

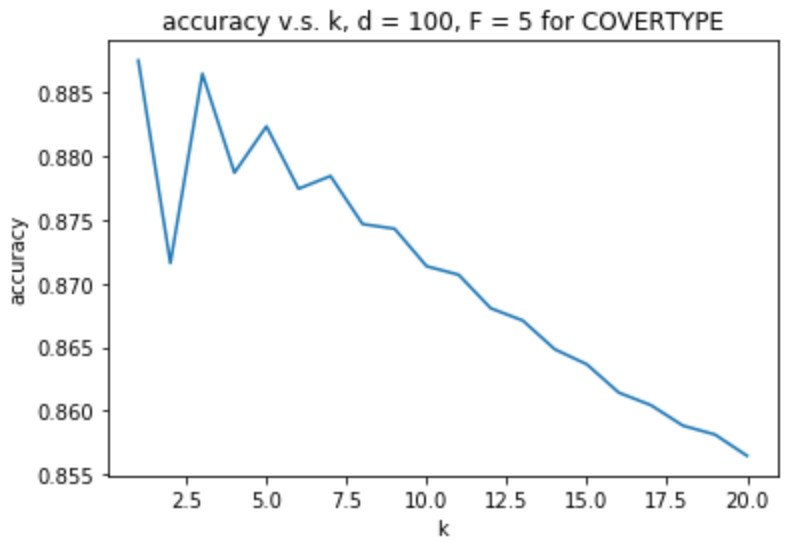


Figure 1-12 accuracy v.s. K, d = 100, F = 5 for COVERTYPE

The figure shows that k = 1 gives the best accuracy around 0.890

Leave-One-Out:

MNIST

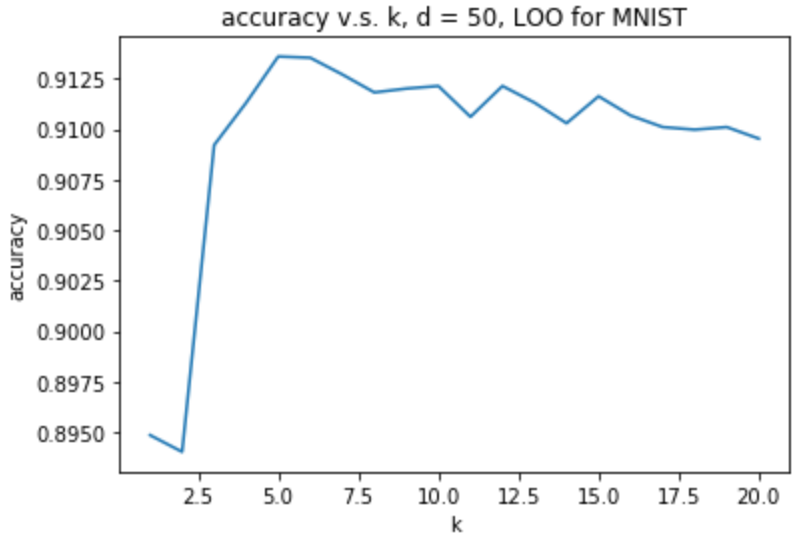


Figure 1-13 accuracy v.s. K, d = 50, LOO for MNIST

The figure shows that k = 7 gives the best accuracy around 0.914

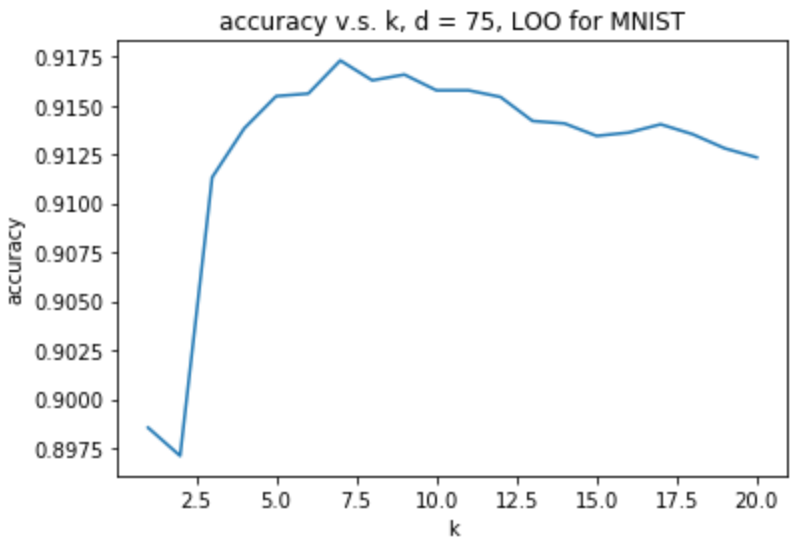


Figure 1-14 accuracy v.s. K, d = 75, LOO for MNIST

The figure shows that k = 8 gives the best accuracy around 0.916

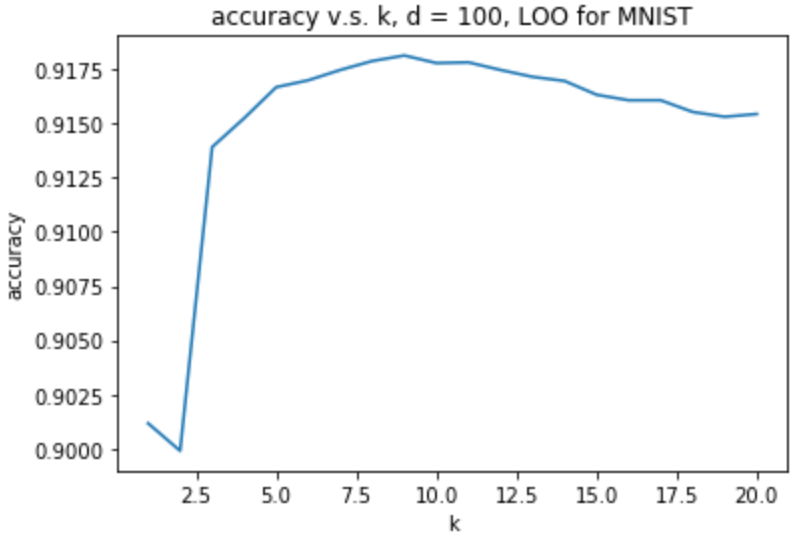


Figure 1-15 accuracy v.s. K, d = 100, LOO for MNIST

The figure shows that k = 9 gives the best accuracy around 0.918

COVERTYPE

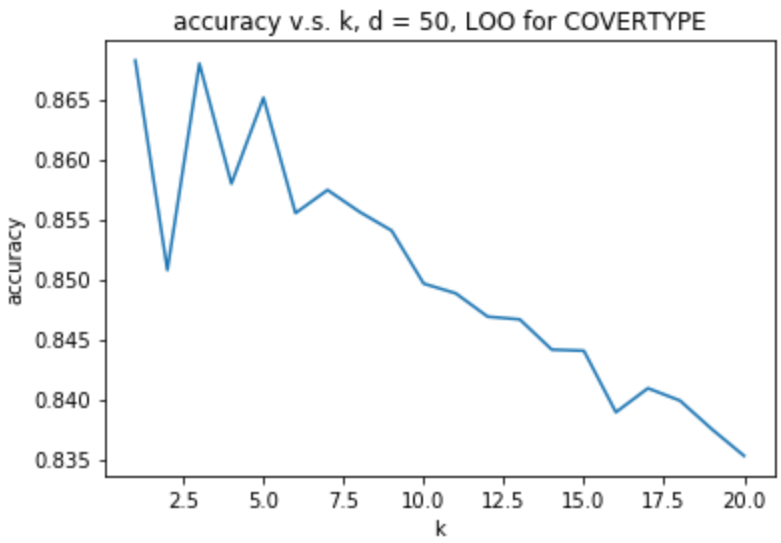


Figure 1-16 accuracy v.s. K, d = 50, LOO for COVERTYPE

The figure shows that k = 1 gives the best accuracy around 0.869

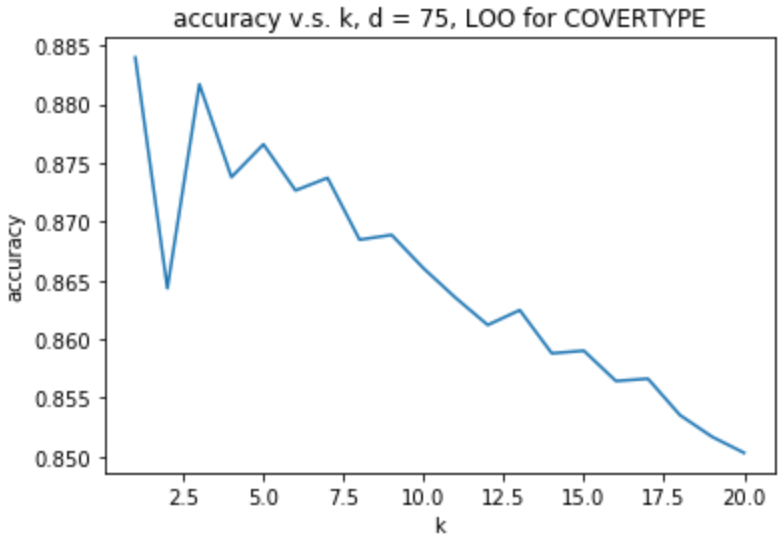


Figure 1-17 accuracy v.s. K, d = 75, LOO for COVERTYPE

The figure shows that k = 1 gives the best accuracy around 0.884

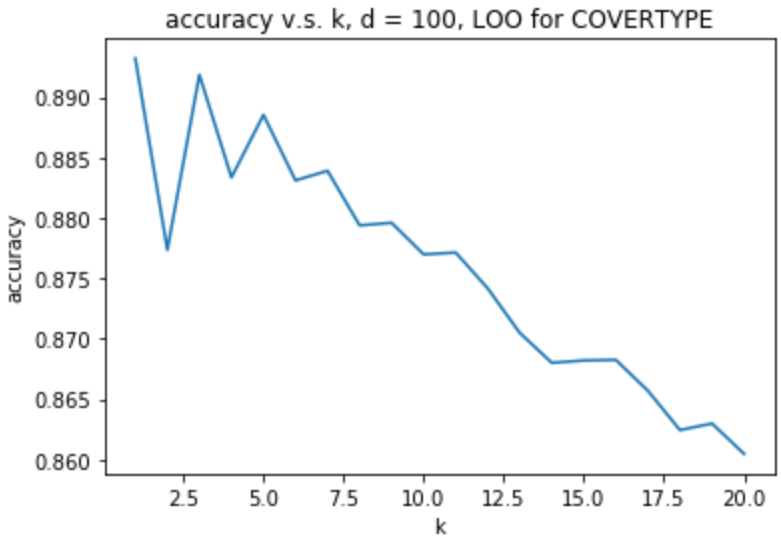


Figure 1-18 accuracy v.s. K, d = 100, LOO for COVERTYPE

The figure shows that k = 1 gives the best accuracy around 0.894

Report the CV errors for the best K.

Below the two tables showed the Best K and its associated error. Accuracies are already showed in above figures, and we can calculate those errors because error = 1 - accuracy.

d/F	2	5	LOO
50	8	8	7
75	8	8	8
100	7	8	9

Table1-1 best K under d% of training data, F-fold, data = MNIST

d/F	2	5	LOO
50	0.093	0.087	0.086
75	0.089	0.086	0.084
100	0.088	0.083	0.082

Table 1-2 error for the best K under d% of training data, F-fold, data = MNIST

d/F	2	5	LOO
50	1	1	1

75	1	1	1
100	1	1	1

Table1-3 best K under d% of training data, F-fold, data = COVERTYPE

d/F	2	5	LOO
50	0.150	0.135	0.131
75	0.136	0.120	0.116
100	0.124	0.110	0.106

Table1-4 error for the best K under d% of training data, F-fold, data = COVERTYPE

As we can see above, we can get better accuracy when d = 100, using LOO. This is because higher d means more training data, thus improved the accuracy. Also, by looking at table, we can pick k resulting in the best accuracy. Our Approach is to use the majority vote among F= 2,5 and LOO for k, so for MNIST dataset, we choose k = 8, k = 8, k = 8 for d = 50, d = 75 and d = 100, and for COVERTYPE dataset, we choose k = 1 for every d. The result is showed in the table.

data/d%	50	75	100
MNIST	8	8	8
COVERTYPE	1	1	1

Table1-5 best k given different d%

Test the TtD with the best K and its associated TD and compare the error rate to CV error.

50% training set:

accuracy for MNIST, d= 50 is : 0.91580952381

accuracy for COVERTYPE, d= 50 is : 0.871858068336

data/error	test	F =2	F = 5	LOO
MNIST	0.084	0.093	0.087	0.086
COVERTYPE	0.128	0.150	0.135	0.131

Table1-6 test error and error using F-folds or LOO, d = 50%

75% training set:

accuracy for MNIST, d= 75 is : 0.915333333333

accuracy for COVERTYPE, d= 75 is : 0.884739041534

data/error	test	F =2	F = 5	LOO
MNIST	0.089	0.086	0.084	0.085
COVERTYPE	0.115	0.136	0.120	0.116

Table1-7 test error and error using F-folds or LOO, d = 75%

100% training set:

accuracy for MNIST, d= 100 is : 0.91923809523

accuracy for COVERTYPE, d= 100 is : 0.89246349473

data/error	test	F =2	F = 5	LOO
MNIST	0.081	0.088	0.084	0.082
COVERTYPE	0.108	0.124	0.110	0.106

Table1-8 test error and error using F-folds or LOO, d = 100%

Report which F estimated the test error the best

First we look at the table and compare with our accuracy given the testing data. We can figure out that no matter what percentage the training data is, using Leave-One-Out always give the closest accuracy to the test accuracy for both MNIST and COVERTYPE. This is because Leave-One-Out method not only has a larger training set, which can represent the dataset better, but also trains and tests way more times than 2-fold or 5-fold, thus averaged random noises.

Report on the stability of the results with respect to the size of the TD

We use 10-Fold cross validation, k = 7 to test the model's stability given the percentage d of the training data = 50,75,100. We evaluate the stability by taking the variance of the accuracies from each fold. And below shows the result:

d	50	75	100
variance	8.27855883094e-05	1.89528595859e-05	9.59838750315e-06

Table1-9 Stability of MNIST with percentage = 50,75,100

d	50	75	100
variance	7.23361091328e-06	2.51654733183e-06	2.44966547212e-06

Table1-10 Stability of COVERTYPE with percentage = 50,75,100

Greater d results in the smaller variance. The reason is that when d = 50%, the sample is not sufficient enough to make prediction stable. That is, there exist a difference between each training model, so the accuracy fluctuates.

Report on your methodology for choosing the best K and distance metric.

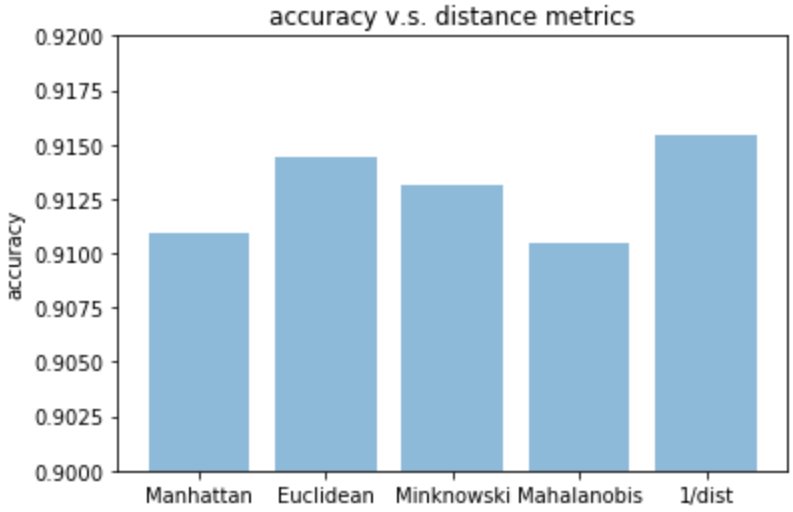


Figure 1-19 accuracy using different distance metric

Here we use MNIST, d = 50% as out data and try different approaches for comparison. We found that the 1/dist gives the best accuracy. It is not always the best choice for us to use 1/dist as our weight because it makes the prediction more dependent on the nearer data, since we assume the training data has a very good quality, using 1/dist is useful.

Report on the effect on bias and variance of the classifier w.r.t. values of K

By the result k = 7 gives the best prediction for MNIST. The reason is that when k is too low, the prediction boundary depends on the single data, so it is easily to get overfitted. In this case we might have low bias but high variance, so our model might not flexible enough to predict various test data. On the other side, when k is too high, the boundary is way too general because it consider lots of data for prediction, causing underfitted. We expected that there is a k which gives the minimum error by the equation: Error = Bias + Variance.

However, in the COVERTYPE the distribution is not uniform at all. By the figure, we can see that class1 and class2 dominate the entire test set. So if we increase k, there might be a chance that the test data would pick the training data belonged to dominated class. If the test data belongs to other class, the error occurs. So in this case, k should be small. Also, k should be the odd number by convention because we does not understand how the KNN work when two classes are tied, so prediction with even k is not valuable.

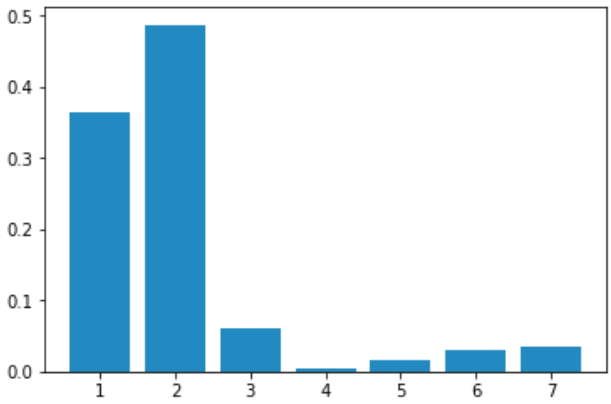


Figure1-20 distribution of the COVERTYPE

2. SVM

a. Cross Validation for different kernels, d, and F-fold

For MNIST:

Accuracy Table for d = 50:

	linear	poly	rbf	sigmoid
2-fold	0.896444	0.905079	0.915429	0.671810
5-fold	0.896762	0.911619	0.919175	0.657905

Accuracy Table for d = 75:

	linear	poly	rbf	sigmoid
2-fold	0.898370	0.909841	0.920042	0.655111
5-fold	0.899005	0.913693	0.922370	0.637206

Accuracy Table for d = 100:

	linear	poly	rbf	sigmoid
2-fold	0.897778	0.913429	0.920794	0.638349
5-fold	0.898762	0.915810	0.923079	0.631714

For COVERTYPE:

Accuracy Table for d = 50:

	linear	poly	rbf	sigmoid
2-fold	0.719239	0.712533	0.737409	0.579579

5-fold	0.719294	0.712758	0.739805	0.589717

Accuracy Table for d = 75:

	linear	poly	rbf	sigmoid
2-fold	0.718964	0.713356	0.739887	0.578178
5-fold	0.719023	0.714821	0.740379	0.588294

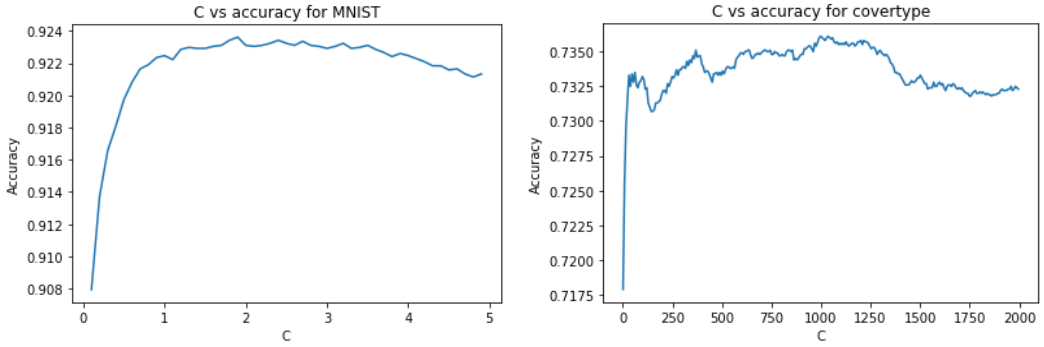
Accuracy Table for d = 100:

	linear	poly	rbf	sigmoid
2-fold	0.718897	0.714039	0.742273	0.577748
5-fold	0.720751	0.715394	0.747392	0.578231

From tables above, we can tell that for both MNIST and COVERTYPE dataset, ‘rbf’ is the best kernel which outperform others for at least 1% of accuracy. Also, ‘sigmoid’ kernel is the worst by only looking at the results.

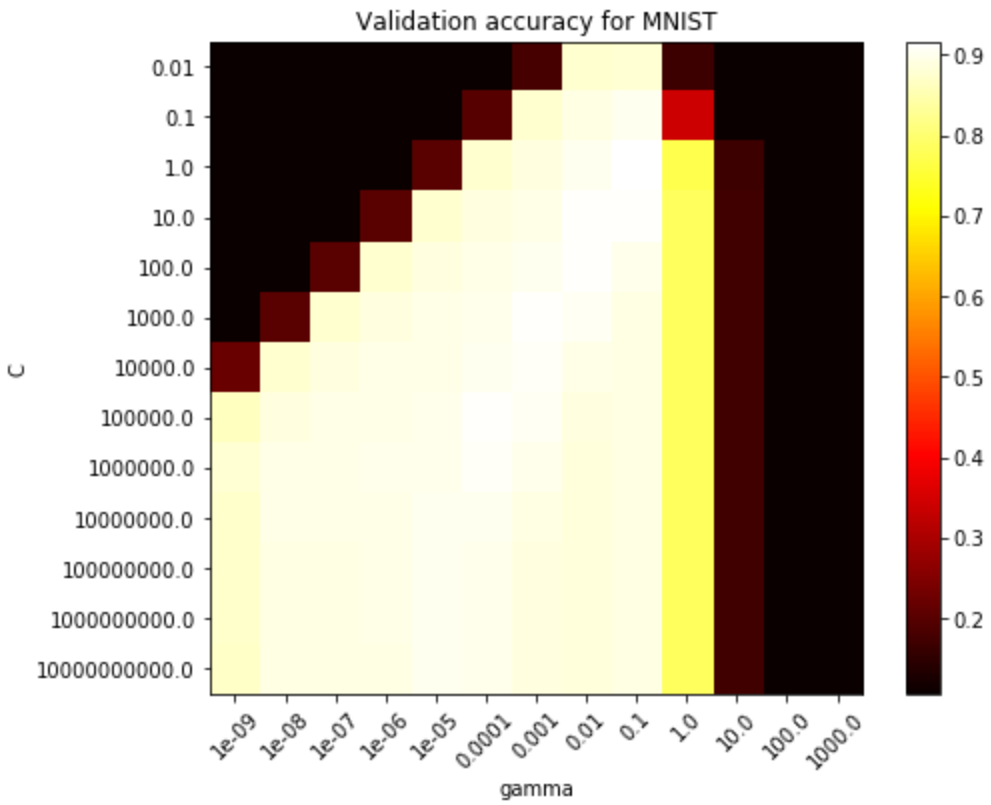
b. Finding parameters

For best performance, we decide to use the ‘rbf’ kernel. The SVM function in sklearn takes two parameters for ‘rbf’ kernel, C and gamma. C stands for the penalty constant of error terms, while gamma is the parameter of fitting the data. That is, increase C (as shown in the figures below) or increase gamma can help the model to fit the data better. We then can plot the cross-validation according to different C and gamma.

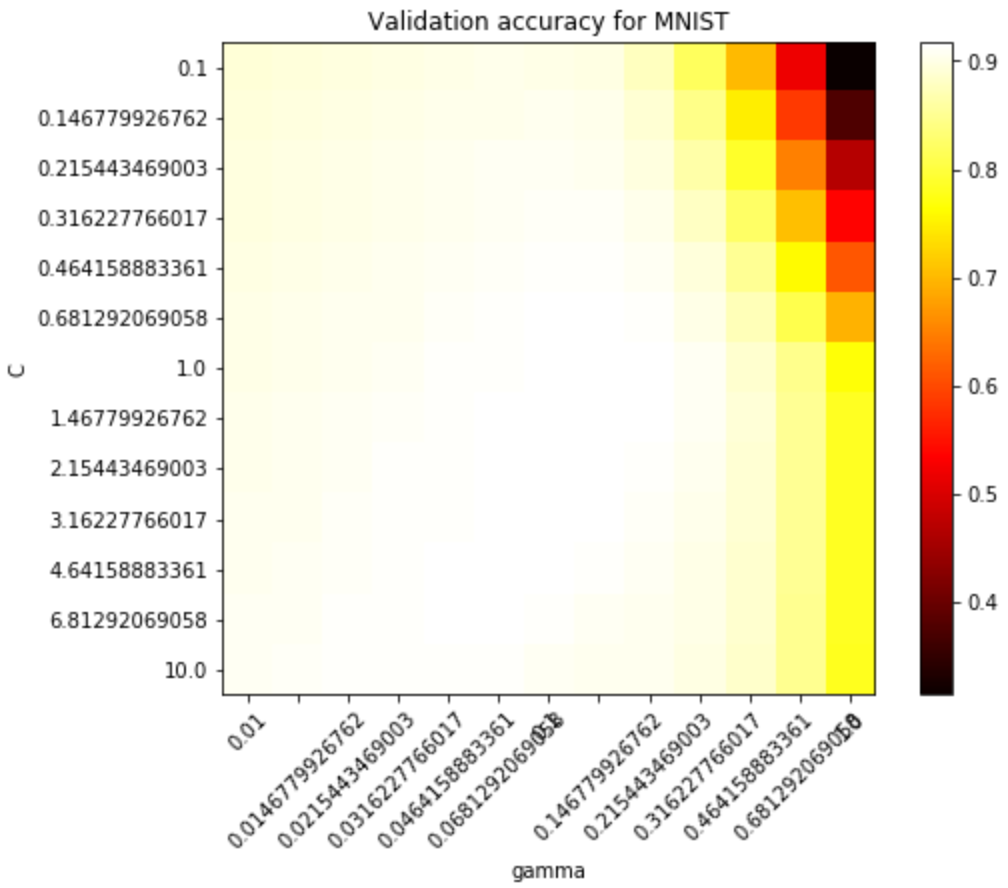


(Increasing C helps improving the accuracy but might over-fit the data if C is too large)

For MNIST:

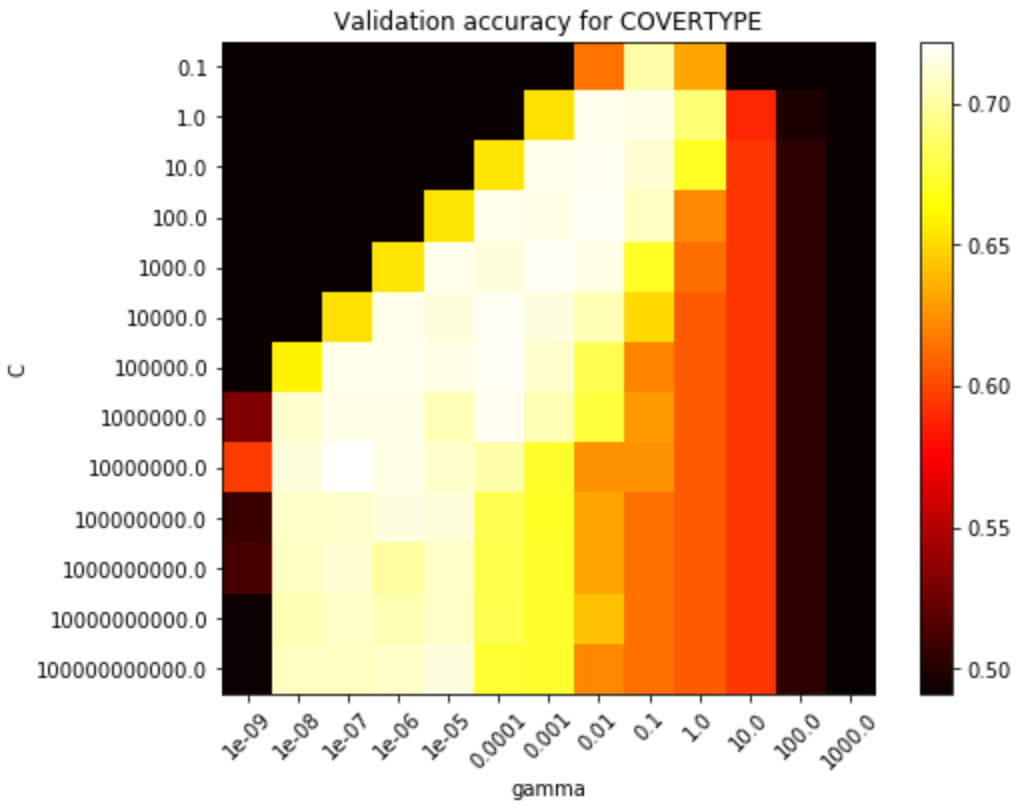


We know that in the up-left corner where C = 0.01 and gamma = 1e-09, the model is under-fitted with low parameter values. In the other hand, in the bottom-right corner of the figure, the model is over-fitted with parameter values too high. Therefore, in the middle, perpendicular to the diagonal line, exists a white belt-like area where C and gamma are balanced for the best accuracy. Let’s get a closer look to the white area.



We can see that when C = 1 and gamma = 0.1, the accuracy is the highest, thus we choose C as 1 and gamma as 0.1.

For COVERTYPE:



We can also get the same observation as MNIST form COVERTYPE. There exists a belt-like high-accuracy region in the middle of the chart. However, the different part of the two dataset is that classes of MNIST are nearly uniformly distributed, while in COVERTYPE, 80% of data belongs to classes 1 and 2. Therefore, we need a higher C to fit the data better. In this case, we choose C = 100 and gamma = 0.01.

c. Stability

For MNIST, we ran 10-Fold SVM on different d (size of TD). The variances of the results are:

d	50%	75%	100%
variance	2.94442e-05	2.54733e-05	8.51499e-06

For COVERTYPE, we also ran 10-Fold SVM on different d (size of TD). The variances of the results are:

d	50%	75%	100%
variance	4.19635e-06	8.61370e-07	1.10357e-07

We can see that same as KNN above, the variance reduces when d increases. The reason is also the same, larger training subset can represent the whole dataset better.

d. Results
Accuracy of MNIST with 'rbf' kernel and C = 1, gamma = 0.1:
50% training set: 0.922571428571
75% training set: 0.923619047619
100% training set: 0.923142857143
Accuracy of COVERTYPE with 'rbf' kernel and C = 100, gamma = 0.01:
50% training set: 0.722022953054
75% training set: 0.730718126304
100% training set: 0.73179

Also, from the results, we know that 5-fold estimate the error closer than 2-fold. It is because 5-fold train and test on the dataset three more times than 2-fold and has a larger training set, the same reasons in KNN.

e. Bias and Variance
We can see that when C and gamma are low, the model is under-fitted. As a result, the bias is high while the variance is low. However, when C and gamma are high, the model is overfitted. Thus, the bias is low while the variance is high.

3. Deliverables
   Comparison between KNN and SVM
   a. Classification accuracy

	MNIST	COVERTYPE
KNN	0.919	0.892
SVM	0.923	0.732

The performance of SVM is slightly better on MNIST, while KNN outperform SVM on COVERTYPE. Generally, SVM is more accurate than KNN because instead of using finite neighbor data, SVM actually find the boundaries separating the classes. Also, we can tune the parameters C and gamma to reduce the impact of outliers. However, in COVERTYPE, the distribution of the classes are not uniform, most of the classes are class 1 and class 2. In this case, the boundaries in SVM are dominated by class 1 and class 2, while with KNN, we can get votes from only the neighbor data, which will be less influenced by the major classes.

   b. Training and testing time

	MNIST	COVERTYPE
KNN	2.59 sec	10.41 sec
SVM	19.7 sec	5 h 25 min 25 sec

Apparently, SVM is way slower than KNN, especially on large dataset such as COVERTYPE. According to paper, “Fast Kernel Classifiers with Online and Active Learning”, written by Antoine Bordes et al., the time complexity of SVM with rbf kernel is O(train_size^2*n_features), while for KNN, the time complexity is O(train_size*test_size). Thus, SVM will be slower than KNN.

c. Fitted model size in memory
   Although we cannot find a proper function for measuring the memory usage, we found out that when we train the data on SVM, our CPU usage will always be above 90%, while our CPU usage will only stay below 20% when running KNN. This means that SVM takes more memory to process than KNN does.