Software Project Assignment Details

Background

One learning outcome of this class is the ability to implement techniques related to pattern recognition. In this project, you will implement a pattern recognition algorithm and compare its performance to another pattern recognition algorithm. You will also gain experience in preparing a concise technical report.

Project description

In this project, you will write software to implement a new classification algorithm called kthperclass (see separate file for algorithm details). The algorithm's performance will be evaluated by taking an existing dataset (the Iris dataset) and splitting the data into training and test vectors. Performance will then be compared to the kNN classifier. You do not need to write software for the kNN classifier. You may use an existing software implementation.

So that we all get the same results, the kNN classifier needs to use the same distance metric and the same tie-breaking rule. Use the "city block" or "Manhattan" distance metric (this distance metric was chosen because it can be easily computed manually from a scatter plot of one or two features). With this metric, the distance between vectors [a b] and [c d] is given by

$$\text{Distance} = |a-c| + |b-d|.$$

As for ties, if we assign the classes setosa, versicolor, and virginica the numbers 1, 2, and 3, then when a tie occurs, the decision goes with the class that was assigned a lower number. Thus, for k=1, if the two closest points have the same distance to the test vector and correspond to setosa and virginica, then setosa is the decided (detected) class.

The Iris dataset

You will work with the classic Iris dataset created by Fisher. The iris is a type of flower which has petals and sepals. The original dataset and documentation can be found at

Iris - UCI Machine Learning Repository

Note: The file iris.data contains the 150 lines corresponding to the 150 data points (variables separated by commas like a csv file). The file iris.names provides documentation, including typographical errors in the data that need to be corrected in iris.data.

There are 3 classes and 50 data points per class. For initial testing, you will use the entire set for training. For final classification performance, you will create a training set by keeping the first 30 points from each class. The remaining 60 points (20 per class) will be used for test.

Each feature vector consists of 4 features: sepal length [cm], sepal width [cm], petal length [cm], and petal width [cm]. For milestone one you will consider all 4 features. **However, for milestone two and the final report, you will only consider two features: petal length and petal width (3rd and 4th features)**.

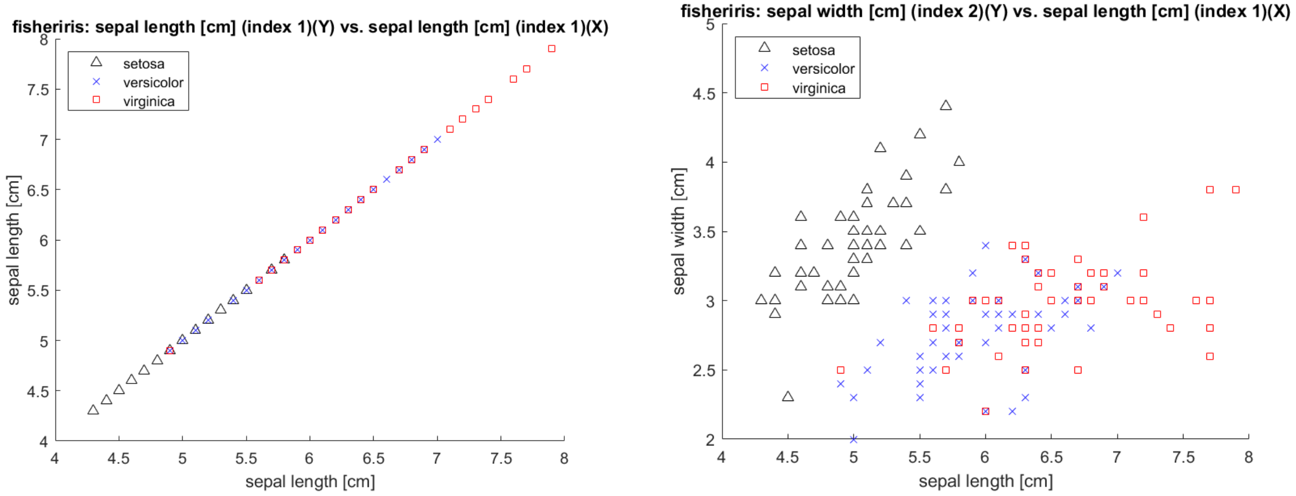
Milestone one: initial dataset evaluation

The project includes intermediate milestones. The first milestone involves taking a look at the data using scatter plots (specified as y value vs. x value). For the 4 features in the Iris dataset, provide scatter plots for 1 vs. 1, 2 vs. 1, 3 vs. 1, 4 vs. 1, 2 vs. 2, 3 vs. 2, 4 vs. 2, 3 vs. 3, 4 vs. 3, and 4 vs. 4 (10 plots). Use different markers for different classes. Use the plotting ranges in Table A.

Table A. Plotting ranges for Iris dataset features.

Feature	Lowest value	Highest value
Sepal length [cm]	4	8
Sepal width [cm]	2	5
Petal length [cm]	1	7
Petal width [cm]	0	3

Make sure you label each axis and provide a legend of what the markers correspond to. The title should at least include the dataset name. Examples of the first two scatter plots are given below in Fig. A. Observe from Fig. Aa that sepal length cannot be used to perfectly separate any two classes because for each pair of classes there exists feature values that are the same.



(a)

(b)

Figure A. Scatter plots for pairs of features in iris dataset.

In addition to providing the 10 scatter plots, answer the following questions with regards to the data.

1. Can you perfectly separate the setosa and versicolor classes using only one feature? If so, which feature or features would work?
2. Can you perfectly separate the setosa and virginica classes using only one feature? If so, which feature or features would work?
3. Can you perfectly separate the versicolor and virginica classes using only one feature? If so, which feature or features would work?

Milestone two: kNN classification evaluation

The second milestone involves evaluating the kNN classifier for k=1 and k=3. For the remainder of the project, only the third and fourth features (petal length and petal width) will be used. For milestone two, all 150 datapoints will be used as training data. The test vectors to be used are given in Table B.

Table B. Iris dataset test vectors (petal length and petal width) for milestone two.

Test case	Petal length	Petal width	True Class	Pred Class k=1	Pred Class k=3
1	2.0	0.8	Setosa		
2	4.0	0.8	Versicolor		
3	6.5	2.5	Virginica		
4	4.5	1.7	Virginica		
5	4.8	1.8	Virginica		
6	5.0	1.8	Versicolor		
7	5.0	1.5	Virginica		

Provide the following.

1. Complete Table B with the kNN classifier results for k=1 and k=3.
2. For both k=1 and k=3, provide confusion matrices.
3. For both k=1 and k=3, fill in Table C, providing the overall probability of classification error as well as conditional classification error probabilities, conditioned on the true class.
4. Answer the following question. **For test case 7, why did the k=3 classifier choose versicolor?** The scatter plot in Fig. B suggests that for k=3, it should choose virginica. Hint: take a closer look at the dataset.

Note: test case 5 can be used to verify the tie-breaking rule. For k = 1, there is a tie between versicolor and virginica. According to the "smallest" rule, versicolor (2nd class) should be selected instead of virginica (3rd class).

Note: test case 6 reveals a roundoff error problem with the kNN classifier in MATLAB® (Python may not have this). For k = 1, there should be a tie, in which case Versicolor should be the detected (predicted) class. However, due to roundoff, you may get Virginica as the detected class. To ensure that roundoff error does not bypass the tie-breaking rule, **you should feed the classifier integer features** (e.g. feature_int = round(10*feature)).

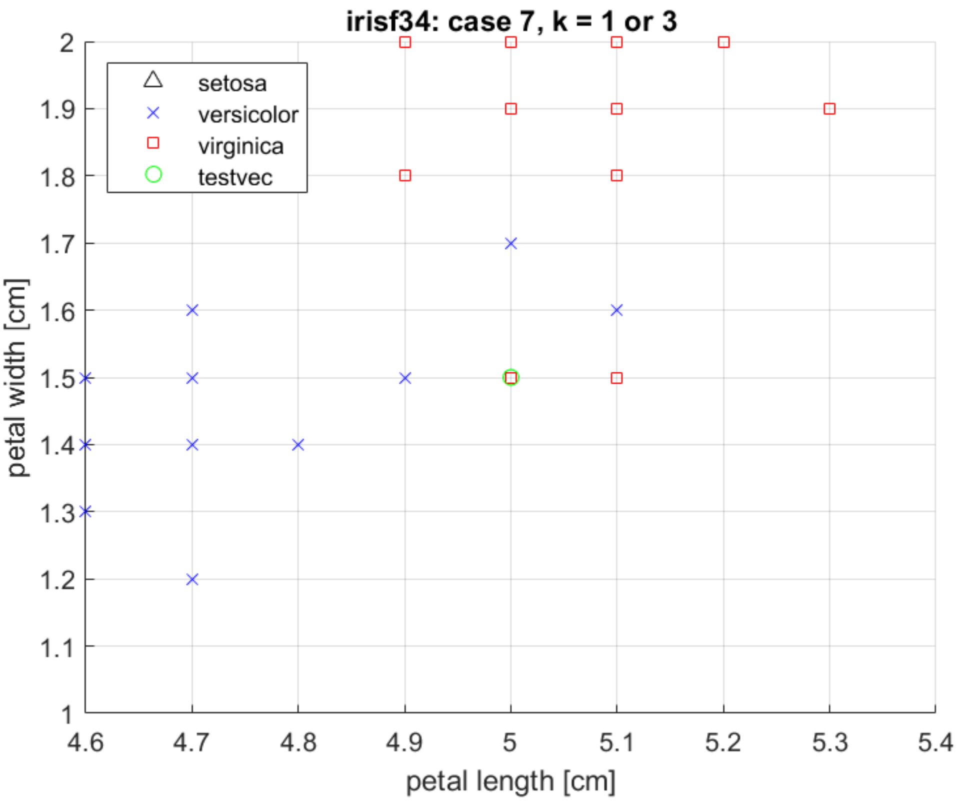


Figure B. Scatter plot of fisheriris dataset (features petal length and petal width only) and test vector 7.

Table C. Overall & conditional classification error probabilities for kNN classification trained with the irisf34 dataset.

kNN k value	Overall Pe	Pe | setosa	Pe | versicolor	Pe | virginica
1				
3				

Final report

The final report involves testing the kthperclass classifier that you implemented and then comparing performance with the kNN classifier.

First, the kthperclass classifier is tested for **k=1** and **k=2**, using all 150 datapoints as training data. The test vectors to be used are given in Table D.

Table D. Iris dataset test vectors and results for kthperclass classification.

Test case	Petal length	Petal width	True Class	Pred Class k=1	Pred Class k=2
1	2.0	0.8	Setosa		

2	4.0	0.8	Versicolor		
3	6.5	2.5	Virginica		
4	4.5	1.7	Virginica		
5	4.8	1.8	Virginica		
6	5.0	1.8	Versicolor		
7	5.0	1.5	Virginica		

Provide the following for the test results.

1. Complete Table D with the kthperclass classifier results for k=1 and k=2.
2. For both k=1 and k=2, provide confusion matrices.
3. For both k=1 and k=2, fill in Table E, providing the overall probability of classification error as well as conditional classification error probabilities, conditioned on the true class.

Table E. Overall & conditional classification error probabilities for kthperclass classification trained with the irisf34 dataset.

kthperclass k value	Overall Pe	Pe | setosa	Pe | versicolor	Pe | virginica
1				
2				

Second, compare the classification error performance for the kNN and kthperclass classifiers. You will need to write two functions

1. **Splitdata**: it will take the original data set and split it into a training set and a test set. For the Iris dataset, this will involve taking the 50 vectors of each class and using the first Nt for training and the remaining 50-Nt for test. Warning: because of the order of the vectors and labels in the dataset, one cannot simply keep the first 90 vectors as this would give 50 setosa vectors and 40 versicolor vectors.
2. **Kthperclass**: it will take a training set (features and labels) and a test set (feature vectors) and produce a set of detected (predicted) labels.

You will also need to write an overall script or set of scripts, using these and other functions, to evaluate the probability of classification error as a function of k. Use this script with Nt = 30 (30 training vectors per class, 20 test vectors per class) to compare

1. kNN classification, k=1 through 17, and
2. kthperclass classification, k=1 through 17

Results for Nt = 35 are shown in Fig. C.

Question: Explain why the probability of classification error appears to take on only certain values.

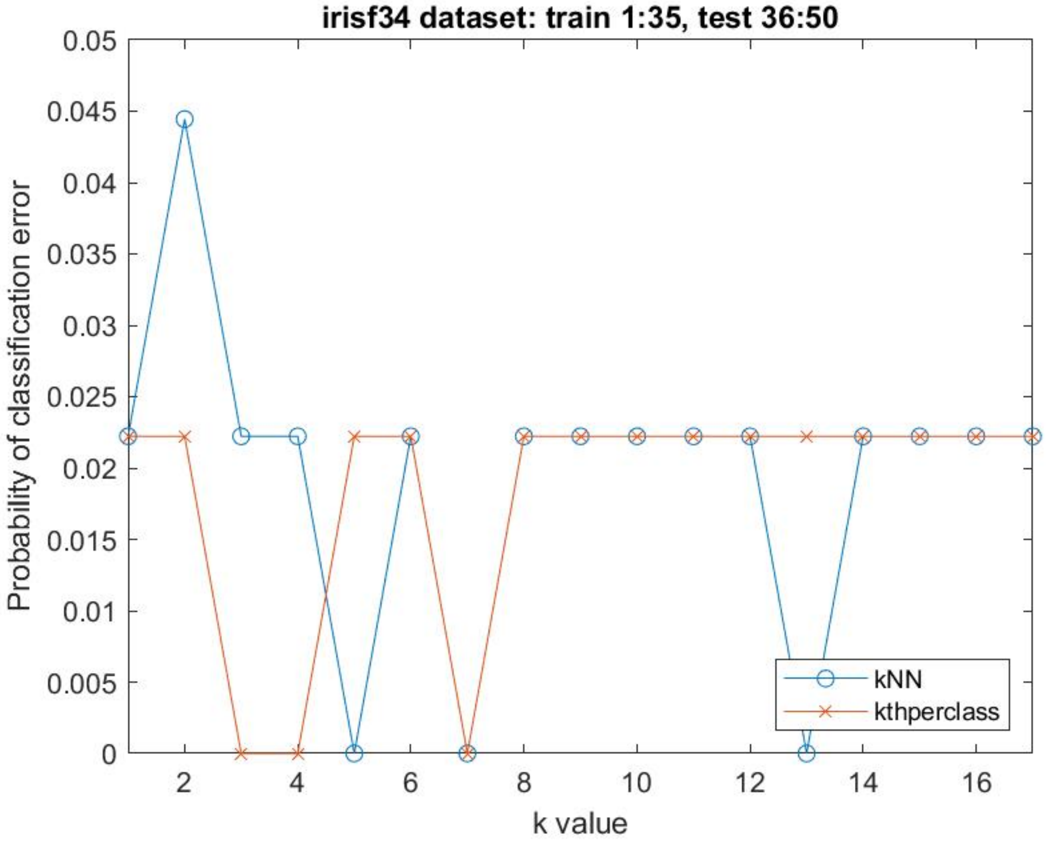


Figure C. Performance results for 35 training vectors per class, 15 test vectors per class.

Project deliverables

Report results using the template provided. The milestones will be given as homework assignments, so you will upload those results in Moodle for those homework assignments.

For the software project assignment item in Moodle, upload two items.

1. The final report providing the results requested using the template provided.
2. A zip file containing the source code you developed. For MATLAB®, this should include the main script as well as any functions you wrote. If using Python, please export the environment containing the packages and their version numbers and include this in your zip file.

Grading rubric

Grading will be based on following directions, level and quality of effort, and effectiveness in communicating results.

Using MATLAB®

The Iris database is built into the statistics and machine learning toolbox in MATLAB®. MATLAB® is freely available to NC State students, see

MATLAB(R) at NC State

The Iris dataset and a kNN classifier are included in the statistics and machine learning toolbox.

Statistics and Machine Learning Toolbox Documentation (mathworks.com)

For those interested in learning more about MATLAB®, see

Get Started with MATLAB (mathworks.com)

The command “load fisheriris” can be used to load a matrix of numerical feature values (meas) and the corresponding class labels (species). The kNN classifier can be trained using the fitcknn function, and test vectors can be evaluated using the predict function. While it is unclear from the documentation, it appears that if your training and test sets consist of setosa vectors followed by versicolor vectors followed by virginica vectors, then it internally assigns 1 to setosa, 2 to versicolor, and 3 to virginica.

In a zip file to be provided, the instructor has written an example (test_knn) on how to use the built-in kNN classifier as well as additional code you may find useful (e.g. the test vectors for milestone 2, meta-data, a plotting example).

Note: MATLAB® stores the true class labels in a 150x1 cell array called `species`. Each cell (e.g. `species{1})` consists of a character array, e.g. `‘setoma’`, not be confused with the string `“setoma”`.

Cell arrays involving character arrays can be tricky to work with. For example, to create an Nx1 cell array `results` to store class values returned by the predict function, you would use something like `results{testcase,1} = predictedclass{1}`. Also, if you collect results in columns of a cell array, then to extract the kth column you use `results(:,k)` not `results{:,k}`.

Using Python

If you use Python, you will need to find a way to get the Iris dataset into Python. Also, to compare the performance of the kthperclass classifier to the kNN classifier, you will need to find a Python implementation of the kNN classifier. Such an implementation needs to be flexible enough to use the distance measure and tie-breaking rule specified.

Note: if you use the original dataset (iris.data file), you need to correct two of the feature vectors (see the iris.names file).

Guidance and hints

Some guidance and hints as you do this project.

1. Plan before writing code. How will you partition the tasks into functions that can be easily tested? What parts of each function will you encode first, so you can test along the way?
2. What additional outputs will a function have so that debugging at a higher level is easier? For example, you may want record not only which class was best, but what the distance metric was and what training vector (or training vector index) gave that distance metric.
3. When debugging a function, you might print intermediate results to the screen. It is convenient to do so by setting a variable (e.g. `verbose`) to one and printing to screen only if this variable is set. That allows for debug mode to be turned back on if new errors occur later. Similarly, you may want to use a variable `plotflag` to plot scatter plots of the training data and single test vector to see what is happening.
4. When possible, look at a result where you know the answer.
    a. Use a simple test dataset where you can manually compute the expected results. Milestone 2 provides one such dataset designed to test the kNN classifier.
    b. Use equivalences to check your code against other code. For example, when k = 1, the kNN classifier and the kthperclass classifer should give the same results (when the same distance metric and tie breaking rule are used).
    c. Use extreme cases to check your code. For example, for the kNN classifier, check that the result is what you would expect when k = total number of training vectors (vectors per class x number of classes).