HOMEWORK 3: KNN, PERCEPTRON LINEAR REGRESSION

10-301/10-601 Introduction to Machine Learning (Summer 2024)
https://www.cs.cmu.edu/~hchai2/courses/10601/
OUT: Thursday, May 23rd
DUE: Tuesday, June 4th

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section on the course site for more information: `https://www.cs.cmu.edu/~hchai2/courses/10601/#Syllabus`
- **Late Submission Policy:** See the late submission policy here: `https://www.cs.cmu.edu/~hchai2/courses/10601/#Syllabus`
- **Submitting your work:**
  - **Programming:** You will submit your code for programming questions on the homework to Gradescope (`https://gradescope.com`). After uploading your code, our grading scripts will autograde your assignment by running your program on a virtual machine (VM). When you are developing, check that the version number of the programming language environment (e.g. Python 3.9.12) and versions of permitted libraries (**only** `numpy` 1.23.0 for this assignment) match those used on Gradescope. You have a **total of 10 Gradescope programming submissions.** Use them wisely. In order to not waste code submissions, we recommend debugging your implementation on your local machine (or the linux servers) and making sure your code is running correctly first before any Gradescope coding submission.
  - **Written:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using Gradescope (`https://gradescope.com/`). Please use the provided template. Submissions must be written in LaTeX. Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. Each derivation/proof should be completed on a separate page. For short answer questions you **should not** include your work in your solution. If you include your work in your solutions, your assignment may not be graded correctly by our AI assisted grader.
- **Materials:** The data that you will need in order to complete this assignment is posted along with the writeup and template on the course website.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For LaTeX users, replace `\choice` with

`\CorrectChoice` to obtain a shaded box/circle, and don't change anything else.

Instructions for Specific Problem Types

For "Select One" questions, please fill in the appropriate bubble completely:

Select One: Who taught this course?

● Henry Chai

○ Marie Curie

○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

Select One: Who taught this course?

● Henry Chai

○ Marie Curie

~~●~~ Noam Chomsky

For "Select all that apply" questions, please fill in all appropriate squares completely:

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

□ I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

~~■~~ I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

Fill in the blank: What is the course number?

$\boxed{\text{10-601}}$ $\boxed{\text{10-}\cancel{6}\text{301}}$

Written Problems (48 points)

1 Decision Trees and *k*NNs (5 points)

1. Consider a binary classification problem using 1-nearest neighbors with the Euclidean distance metric. We have N 1-dimensional training points $x^{(1)}, x^{(2)}, \dots x^{(N)}$ and corresponding labels $y^{(1)}, y^{(2)}, \dots y^{(N)}$, with $x^{(i)} \in \mathbb{R}$ and $y^{(i)} \in \{0, 1\}$.

   Assume the points $x^{(1)}, x^{(2)}, \dots x^{(N)}$ are in ascending order by value. If there are ties during the 1-NN algorithm, we break ties by choosing the label corresponding to the $x^{(i)}$ with lower value.

   (a) (2 points) **Select one:** Is it possible to build a decision tree that behaves exactly the same as the 1-nearest neighbor classifier? Assume that the decision at each node takes the form of "$x \leq t$" or "$x > t$", where $t \in \mathbb{R}$.

   ◯ Yes

   ◯ No

   If your answer is yes, please explain how you will construct the decision tree. If your answer is no, explain why it's not possible.

   Your Answer:

   (b) (3 points) **Select all that apply:** Let's add a dimension! Now the training points are 2-dimensional where $\mathbf{x}^{(i)} = \left(x_1^{(i)}, x_2^{(i)}\right) \in \mathbb{R}^2$. For which of the following training sets is it possible to build a decision tree that behaves exactly the same as a 1-nearest neighbor classifier? Assume that the decision at each node takes the form of "$x_j \leq t$" or "$x_j > t$", where $t \in \mathbb{R}$ and $j \in \{1, 2\}$.

   ☐ Points: $\{(-5, 5), (5, 5)\}$, Labels: $\{0, 1\}$

   ☐ Points: $\{(0, -9), (0, 10)\}$, Labels: $\{0, 1\}$

   ☐ Points: $\{(3, 4), (-3, -4)\}$, Labels: $\{0, 1\}$

   ☐ Points: $\{(0, 0), (0, 1), (1, 0)\}$, Labels: $\{0, 0, 1\}$

   ☐ Points: $\{(0, 0), (0, 1), (1, 0)\}$, Labels: $\{0, 1, 1\}$

   ☐ Points: $\{(1, 2), (3, 4), (5, 6)\}$, Labels: $\{0, 1, 1\}$

   ☐ None of the above

2 *k* Nearest Neighbors (12 points)

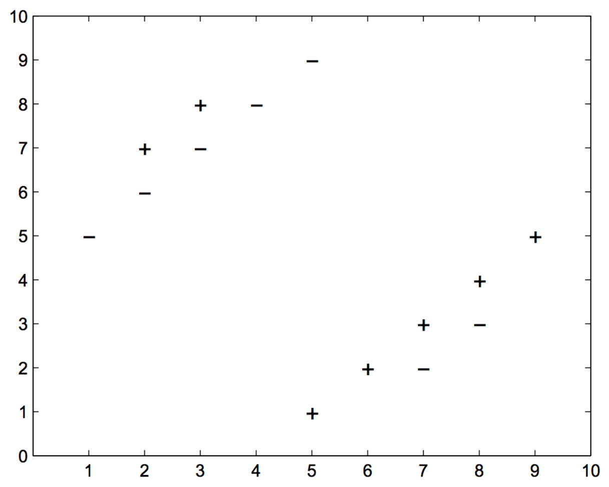


Figure 1: kNN Dataset

1. Consider a k nearest neighbors (kNN) binary classifier which assigns the class of a test point to be the class of the majority of the k nearest neighbors, according to the Euclidean distance metric. Assume that ties are broken by selecting one of the labels uniformly at random.

   (a) (2 points) Using Figure 1 shown above to train the classifier and choosing $k = 6$, what is the training error rate?

Your answer:

(b) (2 points) **Select all that apply:** Let's say that we have a new test point (not present in our training data) $\mathbf{x}^{\text{new}} = [3, 9]^T$ that we would like to apply our kNN classifier to, as seen in figure 2.

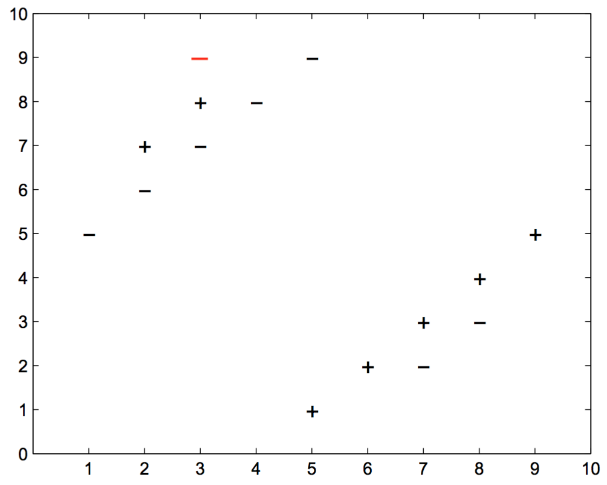


Figure 2: kNN Dataset with Test Point

For which values of k is this test point correctly classified by the kNN algorithm?

- ☐ k = 1
- ☐ k = 5
- ☐ k = 9
- ☐ k = 12
- ☐ None of the above

2. **Select one:** Assume we have a large labeled dataset that is randomly divided into a training set and a test set, and we would like to classify points in the test set using a kNN classifier.

   (a) (1 point) In order to minimize the classification error on this test set, we should always choose the value of k which minimizes the training set error.

   ◯ True

   ◯ False

   (b) (2 points) **Select one:** Instead of choosing the hyperparameters by merely minimizing the training set error, we instead consider splitting the training-all data set into a training and a validation data set, and choose the hyperparameters that lead to lower validation error. Is choosing hyperparameters based on validation error better than choosing hyper-parameters based on training error?

   ◯ Yes, lowering validation error instead of training error is better because lowering training error will not help generalize our model and may lead to overfitting.

   ◯ Yes, lowering validation error is better for the model because cross-validation guarantees a better test error.

   ◯ No, lowering training error instead of validation error is better because lowering validation error will not help generalize our model and may lead to overfitting.

   ◯ No, lowering training error is better for the model because we have to learn the training set as well as possible to guarantee the best possible test error.

   (c) (2 points) **Select one:** Your friend Sally suggests that instead of splitting the original training set into separate training and validation sets, we should instead use the test set as the validation data for choosing hyperparameters. Is this a good idea? Justify your opinion with no more than 3 sentences.

   ◯ Yes

   ◯ No

   Your answer:

3. (3 points) **Select all that apply:** Which of the following is/are correct statement(s) about kNN models?

   ☐ A larger k tends to give a smoother decision boundary.

   ☐ To reduce the impact of noise or outliers in our data, we should increase the value k.

   ☐ If we make k too large, we could end up overfitting the data.

   ☐ We can use cross-validation to help us select the value of k.

   ☐ We should never select the k that minimizes the error on the validation dataset.

□ None of the above.

3 Perceptron & Linear Regression (21 points)

1. (1 point) **Select one:** Consider running the online perceptron algorithm on some sequence of examples S (an example is a data point and its label). Let S' be the same set of examples as S, but presented in a different order.

   True or False: The online perceptron algorithm is guaranteed to make the same number of mistakes on S as it does on S'.

   ◯ True

   ◯ False

2. (2 points) **Select all that apply:** Suppose we have a perceptron whose inputs are 2-dimensional vectors and each feature vector component is either -1 or 1, i.e., $x_i \in \{-1, 1\}$. The prediction function is $y = \text{sign}(w_1 x_1 + w_2 x_2 + b)$, and

$$\text{sign}(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ -1, & \text{otherwise.} \end{cases}$$

   Which of the following functions can be implemented with the above perceptron? That is, for which of the following functions does there exist a set of parameters w, b that correctly define the function.

   □ AND function, i.e., the function that evaluates to 1 if and only if all inputs are 1, and -1 otherwise.

   □ OR function, i.e., the function that evaluates to 1 if and only if at least one of the inputs are 1, and -1 otherwise.

   □ XOR function, i.e., the function that evaluates to 1 if and only if the inputs are not all the same. For example

$$\text{XOR}(1, -1) = 1, \text{ but } \text{XOR}(1, 1) = -1.$$

   □ None of the above.

3. (1 point) **Select one:** Suppose we have a dataset $\left\{\left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(N)}, y^{(N)}\right)\right\}$, where $\mathbf{x}^{(i)} \in \mathbb{R}^M$, $y^{(i)} \in \{+1, -1\}$. We would like to apply the perceptron algorithm on this dataset. Assume there is no intercept term. How many parameter values is the perceptron algorithm learning?

   ◯ N

   ◯ $N \times M$

   ◯ M

4. (2 points) **Select one:** The following table shows a data set and the number of times each point is misclassified during a run of the perceptron algorithm. What is the separating plane $\boldsymbol{\theta}$ found by the algorithm, i.e. $\boldsymbol{\theta} = [b, \theta_1, \theta_2, \theta_3]$? Assume that the initial weights are all zero.

x_1	x_2	x_3	y	Times Misclassified
2	1	5	1	10
5	3	3	1	5
1	6	2	1	8
7	2	1	-1	2
3	2	6	-1	3

○ $[18, 25, 14, 34]$

○ $[18, 30, 63, 61]$

○ $[16, 56, 18, 47]$

○ $[18, 52, 19, 47]$

5. (2 points) **Select one:** Suppose we have data whose examples are of the form $[x_1, x_2]$, where $x_1 - x_2 = 0$. We do not know the label for each element. Suppose the perceptron algorithm starts with $\boldsymbol{\theta} = [3, 5]$; which of the following values will $\boldsymbol{\theta}$ never take on in the process of running the perceptron algorithm on the data?

○ $[-1, 1]$

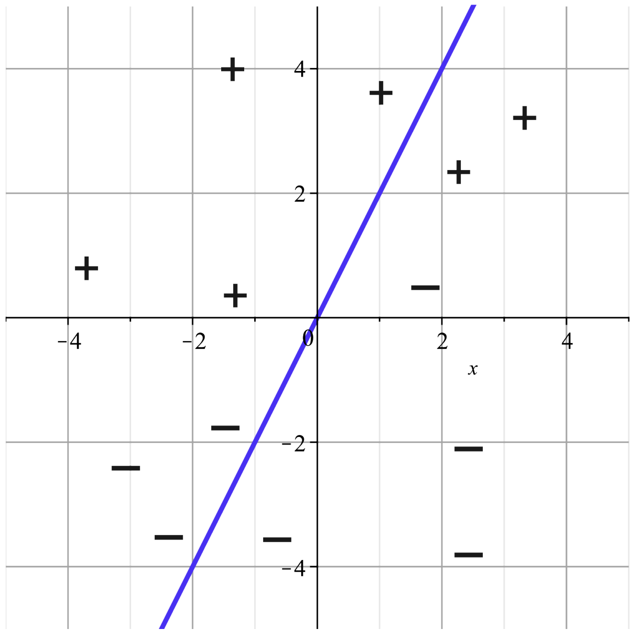
○ $[4, 6]$

○ $[-3, 0]$

○ $[-6, -4]$

6. (2 points) **Select all that apply:** Consider the linear decision boundary below and the dataset shown. Which of the following are valid weights $\boldsymbol{\theta}$ and their corresponding error on this dataset? (Note: Assume the decision boundary is fixed and does not change while evaluating error.)

   - ☐ $\boldsymbol{\theta} = [-2, 1]$, error = 5/13
   - ☐ $\boldsymbol{\theta} = [2, -1]$, error = 8/13
   - ☐ $\boldsymbol{\theta} = [2, -1]$, error = 5/13
   - ☐ $\boldsymbol{\theta} = [-2, 1]$, error = 8/13
   - ☐ None of the above.



7. Consider a dataset $\mathcal{D}_1 = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$. Assume the linear regression model that minimizes the mean-squared error on $\mathcal{D}_1$ is $y = w_1 x + b_1$.

   (a) (2 points) **Select one:** Now, suppose we have the dataset $\mathcal{D}_2 = \{(x^{(1)} + \alpha,\ y^{(1)} + \beta), \ldots, (x^{(N)} + \alpha,\ y^{(N)} + \beta)\}$ where $\alpha > 0, \beta > 0$ and $w_1\alpha \neq \beta$. Assume the linear regression model that minimizes the mean-squared error on $\mathcal{D}_2$ is $y = w_2 x + b_2$. Select the correct statement about w_1, w_2, b_1, b_2 below. Note that the statement should hold no matter what values α, β take on within the specified constraints.

   - ○ $w_1 = w_2, b_1 = b_2$
   - ○ $w_1 \neq w_2, b_1 = b_2$
   - ○ $w_1 = w_2, b_1 \neq b_2$
   - ○ $w_1 \neq w_2, b_1 \neq b_2$

   (b) (2 points) We decide to ask a friend to analyze $\mathcal{D}_1$; however, he makes a mistake by duplicating a subset of the rows in $\mathcal{D}_1$. Explain why the linear regression parameters that minimize mean-squared error on the duplicated data *may* differ from the parameters learned on $\mathcal{D}_1$, i.e. w_1 and b_1.

Your answer:

8. We wish to learn a linear regression model on the dataset $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$ where $\boldsymbol{x} \in \mathbb{R}^k$. For this question, define the *log-cosh loss* as

$$\ell(\hat{y}, y) = \log(\cosh(\hat{y} - y))$$

In particular, for a given point $\boldsymbol{x}^{(i)}$, the log-cosh loss of a model with parameters $\boldsymbol{w}$ is

$$\ell^{(i)}(\boldsymbol{w}) = \log\left(\cosh\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)}\right)\right)$$

We are interested in minimizing loss over our training data, so we minimize the average log-cosh loss over all points in $\mathcal{D}$.

(a) (2 points) What is the objective $\ell(\boldsymbol{w})$ in this setting?

Your answer:

(b) (3 points) What is the partial derivative of $\ell^{(i)}(\boldsymbol{w})$ with respect to the j^{th} parameter, w_j? It may be helpful to know that $\frac{d}{dx}\cosh(x) = \sinh(x)$.

Your answer:

(c) (2 points) What is the gradient of $\ell^{(i)}(\boldsymbol{w})$ with respect to the entire parameter vector $\boldsymbol{w}$?

Your answer:

4 Empirical Questions (10 points)

The following questions should be completed as you work through the programming portion of this assignment.

1. (3 points) For the `iris` dataset, create a *computer-generated* plot showing error rate on the y-axis against k on the x-axis for k going from 1 to 90 (the number of training data points). Use the Euclidean distance metric for this question.

   On a single set of axes, include *both* training error rate and validation error rate, clearly labeling which is which. That is, for each possible value of k from 1 to 90, you should report the error rate of a kNN classifier's predictions on both the training dataset and validation datasets given to you. You should include an image file below using the provided, commented out code in LaTeX, switching out `iris.png` to your file name as needed.

Plot

2. (4 points) Using the `iris` dataset, compare the effect of "training" your kNN model on just the training dataset vs. using both the training and validation datasets by computing the test error rate in both settings using Euclidean distance. Populate the table below for the pre-specified values of k. (Please round each number to the fourth decimal place, e.g. 0.1234)

   NOTE: you should *absolutely* not do something like this in practice; comparing different values of k should only be done using the validation dataset. If you were to use a table like this to inform your selection of k, your reported test error would no longer be a reliable estimate of your kNN classifier's true error; a good rule of thumb is that a test dataset should only be used *once* in the entire pipeline. That being said, we are asking you to complete this task because it illustrates the fact that test error rate tends to decrease when models are trained on more data, regardless of the value of k.

k	**Test Error Rate (Train Only)**	**Test Error Rate (Train + Validation)**
1		
20		
40		
60		

3. (3 points) For the `iris` dataset, create a *computer-generated* plot showing 10-fold cross-validation error on the y-axis against k on the x-axis for k going from 1 to 90 (the number of training data points). For this question, you will use both the Euclidean distance *and* the Manhattan distance to compute the cross-validation errors.

   On a single set of axes, include the 10-fold cross-validation computed using *both* the Euclidean distance and the Manhattan distance, clearly labeling which is which. That is, for each possible value of k from 1 to 90, you should split the training dataset into 10 equal-sized chunks or "folds". Then compute the average error rate when each fold is used as the validation dataset for a kNN model "trained" on the remaining 9 folds with the Euclidean distance metric. Repeat this procedure with the Manhattan distance metric. You should include an image file below using the provided, commented out code in LaTeX, switching out `crossval.png` to your file name as needed.

Plot

5 Collaboration Questions

After you have completed all other components of this assignment, report your answers to these questions regarding the collaboration policy. Details of the policy can be found here.

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details.
3. Did you find or come across code that implements any part of this assignment? If so, include full details.

Your Answer

6 Programming (88 points)

Your goal in this assignment is to implement a classifier based on distance between points - specifically a kNN model. In addition, we will ask you to run some end-to-end experiments selecting the best value for k using the classic Fisher iris dataset. You will write one program: `knn.py`. The program you write will be automatically graded using Gradescope.

6.1 The Tasks and Datasets

Materials Download the zip file from the course website. The zip file will have a handout folder that contains all the data that you will need in order to complete this assignment.

Starter Code The handout will contain a preexisting `knn.py` file that itself contains some starter code for the assignment. While we do not require that you use the starter code in your final submission, we *heavily* recommend building upon the structure laid out in the starter code.

Datasets The handout contains one dataset, which has already been split into training, validation and test datasets. The first line of each `.csv` file contains the name of each attribute, and *the class label is always the last column*.

The task associated with this dataset is to predict the species of an iris, based on physical measurements. Note that in this dataset, there are three possible labels: *iris setosa* (0), *iris versicolor* (1), and *iris virginica* (2). The attributes (aka. features) are:

1. `sepal length`: The length of the sepal in cm.
2. `sepal width`: The width of the sepal in cm.
3. `petal length`: The length of the petals in cm.
4. `petal width`: The width of the petals in cm.

The training data is in `iris_train.csv`, the validation data is in `iris_val.csv`, and the test data is in `iris_test.csv`.

6.2 Program #1: kNN (88 points)

In `knn.py`, you will implement a k nearest neighbors classifier.

Your implementation must satisfy the following requirements:

- Use Euclidian or Manhattan distance (as specified) to calculate distances between points.
- Return predictions for a range of k values as specified on the command line.
- Compute the error of a set of predictions given the true labels.
- Perform validation and 10-fold cross-validation; **NOTE:** the number of folds is not a command line argument, you may hard code in a value of 10 in your implementations.
- Break ties in distance between training data points in favor of data points that appear *earlier* in the training dataset i.e., ones with lower row indices. For example, if a data point is equally close to the 4th and the 10th training data point, then its nearest neighbor would be the 4th training data point.
- Break ties in labels of the nearest neighbors in favor of lower label values. For example, if the 5 nearest neighbors to some data point have labels 0, 0, 1, 2, and 2, then the predicted label would be 0.
- Do not hard-code any aspects of the datasets into your code. We may autograde your programs on hidden datasets that include different attributes and output labels.

Careful planning will help you to correctly and concisely implement your kNN model. Here are a few *hints* to get you started:

- Write and test helper functions to calculate the two distance metrics.
- Be sure to correctly handle the case where the specified number of neighbors is greater than the total number of points.
- When calling the cross-validation method, you should combine the training and validation datasets.
- Also be sure to correctly handle the case where the number of folds does not divide evenly into the number of data points.
- Look under the FAQ post on Piazza for more useful clarifications about the assignment.

6.3 Command Line Arguments

The autograder runs and evaluates the output from the files generated, using the following command:

```
$ python knn.py [args...]
```

Where above `[args...]` is a placeholder for eleven command-line arguments, described in detail below:

1. `<train input>`: path to the training input `.csv` file (see Section 6.1)
2. `<val input>`: path to the validation input `.csv` file (see Section 6.1)
3. `<test input>`: path to the test input `.csv` file (see Section 6.1)
4. `<val type>`: whether we use “normal” validation (0) or cross-validation (1).
5. `<dist metric>`: whether we use Euclidean (0) or Manhattan (1) distance.
6. `<min k>`: The smallest value of k to consider.
7. `<max k>`: The largest value of k to consider.
8. `<train out>`: path of output `.txt` file to which the predictions on the *training* data should be written (see Section 6.4)
9. `<val out>`: path of output `.txt` file to which the predictions on the *validation* data should be written (see Section 6.4)
10. `<test out>`: path of output `.txt` file to which the predictions on the *test* data should be written (see Section 6.4)
11. `<metrics out>`: path of the output `.txt` file to which metrics such as train and test error should be written (see Section 6.5)

As an example, the following command line would run your program on the iris dataset and learn a kNN classifier with the Manhattan distance metric, using 10-fold cross-validation to find the best of k from 10 to 20. The train predictions would be written to `iris_10to20_train.txt`, the validation predictions to `iris_10to20_val.txt`, the test predictions to `iris_10to20_test.txt`, and the metrics to `iris_10to20_metrics.txt`.

```
$ python knn.py iris_train.csv iris_val.csv iris_test.csv 1 1 10 20 \
iris_10to20_train.txt iris_10to20_val.txt iris_10to20_test.txt \
iris_10to20_metrics.txt
```

6.4 Output: Labels Files

Your program should write *either two or three* output `.txt` files depending on the `<val type>` argument.

1. If `<val type>` is 0 ("normal" validation), then you should output the predictions of your model on the training dataset (`<train out>`), validation dataset (`<val out>`), and test dataset (`<test out>`). `<train out>` and `<val out>` should contain an array where the rows correspond to the input values of k and the columns correspond to the predicted labels for each data point. `<test out>` should contain a single list of predictions on the test dataset corresponding to the best value of k i.e., the one with the lowest validation error rate. Columns in a row should be comma separated. Use '`\n`' to create a new line.

   For example, if you set `<min k>` to 10 and `<max k>` to 20, then `<train out>` would have 11 rows and 90 columns (as there are 90 training data points); the 3rd entry in the 2nd row would be the prediction of an 11NN model on the 3rd training data point.

   The first few rows and columns of a sample `<train out>` output file are given below for `<val type>` 0, where k ranges from 1 to 5 and the Euclidean distance metric is used:

   ```
   2,1,2,0,...
   2,1,1,0,...
   2,1,2,0,...
   ...
   ```

2. If `<val type>` is 1 (10-fold cross-validation), then you should output the predictions of your model on each fold of the combined training and validation datasets when that fold is held out (`<val out>`) as well as the predictions on the test dataset (`<test out>`). `<val out>` should contain an array where the rows correspond to the input values of k and the columns correspond to the predicted labels for each data point when that data point is not used to train the model. `<test out>` should contain a single list of predictions on the test dataset corresponding to the best value of k i.e., the one with the lowest cross-validation error rate. Columns in a row should be comma separated. Use '`\n`' to create a new line.

   For example, if you set `<min k>` to 10 and `<max k>` to 20, then `<val out>` would have 11 rows and 120 columns (as there are 120 total training and validataion data points); the 3rd entry in the 2nd row would be the prediction of an 11NN model on the 3rd data point when the last 108 data points (the 2nd through 10th folds or all but the 1st fold) are used as the training dataset.

   The first few rows of a sample `<test out>` output file are given below for `<val type>` 1, where k ranges from 1 to 10 and the Manhattan distance metric is used:

   ```
   1
   1
   0
   1
   ...
   ```

Your labels should exactly match those of a reference kNN implementation — this will be checked by the autograder by running your program and evaluating your output file against the reference solution.

6.5 Output: Metrics File

Generate another file where you report the error rates for the various kNN classifiers your program implements. This file should be written to the path specified by the command line argument `<metrics out>`. Again, the contents of this file will depend on `<val type>`.

1. If `<val type>` is 0 ("normal" validation), then you should output the following error rates in this order:

   - the training error rates for each value of k (each on their own line),
   - the validation error rates for each value of k (again, each on their own line),
   - the test error rate corresponding to just the best value of k (the one that minimizes the validation error rate) *where only the training dataset is used to make predictions*, and
   - the test error rate corresponding to just the best value of k (the one that minimizes the validation error rate) *where the training and validation datasets are first combined and then used to make predictions*.

   Use '`\n`' to create a new line.

   A few lines of a sample `<metrics out>` output file are given below for `<val type>` 0, where k ranges from 1 to 5 and the Euclidean distance metric is used:

   ```
   k=1 training error rate: 0.0
   k=2 training error rate: 0.044444444444444446
   k=3 training error rate: 0.044444444444444446
   ...
   k=1 validation error rate: 0.06666666666666667
   k=2 validation error rate: 0.06666666666666667
   k=3 validation error rate: 0.03333333333333333
   ...
   test error rate (train): 0.0
   test error rate (train + validation): 0.0
   ```

2. If `<val type>` is 1 (10-fold cross-validation), then you should output the 10-fold cross-validation error rates for each value of k (each on their own line) followed by the error rate for just the best value of k (the one that minimizes the cross-validation error rate). Use '`\n`' to create a new line.

   A few lines of a sample `<metrics out>` output file are given below for `<val type>` 1, where k ranges from 1 to 10 and the Manhattan distance metric is used:

   ```
   k=1 cross-validation error rate: 0.06666666666666667
   k=2 cross-validation error rate: 0.09166666666666666
   k=3 cross-validation error rate: 0.05
   ...
   test error rate: 0.0
   ```

Again, your reported error rates should be within 0.0001 of the reference solution. You do not need to round your reported numbers! The autograder will automatically incorporate the right tolerance for float comparisons.

At this point, you should be able to go back and answer questions 1-3 in the "Empirical Questions" section of this handout. Write your solutions in the template provided.

6.6 Submission Instructions

Programming Please ensure you have completed the following file for submission.

`knn.py`

When submitting your solution, make sure to select and upload the correct file. **Any other files will be deleted.** Ensure the file has the exact same spelling and letter casing as above. You can either directly zip the file (by selecting the file and compressing it – do not compress the folder containing the file) or directly drag it to Gradescope for submission.

Written Questions Make sure you have completed all questions from Written component (including the collaboration policy questions) in the template provided. When you have done so, please submit your document in **PDF format** to the corresponding assignment slot on Gradescope.