CSci 5521: Fall'20
Introduction To Machine Learning

Homework 1

(Due Friday, October 02, 11:59 pm)

1. (25 points) Consider doing least squares regression based on a training set $\mathcal{Z}_{\text{train}} = \{(x_t, r_t), t = 1, \ldots, N\}$, where $x_t \in \mathbb{R}$ and $r_t \in \mathbb{R}$.

   (i) (10 points) Consider fitting a linear model of the form

   $$g_1(x) = w_1 x + w_0 \ ,$$

   with unknown parameters $w_1, w_0 \in \mathbb{R}$, which are selected so as to minimize the following empirical loss:

   $$E(w_1, w_0 | \mathcal{Z}_{\text{train}}) = \frac{1}{N} \sum_{t=1}^{N} (r_t - (w_1 x_t + w_0))^2 \ .$$

   Derive the optimal values of (w_1, w_0) clearly showing all steps of the derivation.

   (ii) (10 points) Consider fitting a polynomial model of the form

   $$g_2(x) = v_2 x^{20} + v_1 x^3 + v_0 \ ,$$

   with unknown parameters $v_2, v_1, v_0 \in \mathbb{R}$, which are selected so as to minimize the following empirical loss:

   $$E(v_2, v_1, v_0 | \mathcal{Z}_{\text{train}}) = \frac{1}{N} \sum_{t=1}^{N} (r_t - (v_2 x_t^{20} + v_1 x_t^3 + v_0))^2 \ .$$

   Derive the optimal values of v_2, v_1, v_0 clearly showing all steps of the derivation.[1]

   (iii) (5 points) For a given training set $\mathcal{Z}_{\text{train}}$, let (w_1^*, w_0^*) be the optimal values of (w_1, w_0) in (i) above, and let (v_2^*, v_1^*, v_0^*) be the optimal values of (v_2, v_1, v_0) in (ii) above. Professor Gopher claims that the following is true for any given $\mathcal{Z}_{\text{train}}$:

   $$E(v_2^*, v_1^*, v_0^* | \mathcal{Z}_{\text{train}}) \leq E(w_1^*, w_0^* | \mathcal{Z}_{\text{train}}) \ .$$

   Is Professor Gopher's claim correct? Clearly explain your answer.[2]

2. (15 points) Consider the following 4×4 matrix:

   $$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \end{bmatrix} \ .$$

[1] It is ok to leave the solution in terms of a linear system, say $A\mathbf{v} = \mathbf{b}$, where $A \in \mathbb{R}^{3\times 3}$, $\mathbf{b} \in \mathbb{R}^3$ are known, and $\mathbf{v} = [v_2 \ v_1 \ v_0]^T \in \mathbb{R}^3$ is a vector of the unknown parameters. If you choose to do this, please also mention your preferred approach to solve such a linear system.

[2] A correct answer with insufficient or incorrect explanation will not get any credit.

(i) (5 points) What are the values of $\text{tr}(A), \text{tr}(A^T), \text{tr}(A^TA)$, and $\text{tr}(AA^T)$.[3]

(ii) (5 points) From a geometric perspective, explain how the absolute value of $|A|$ (determinant of A) can be computed.

(iii) (5 points) Are the rows of A linearly independent? Clearly explain your answer.[2] (For this problem, you can use python libraries to arrive at your answer. If you do that, clearly explain what you did and why. There is a way to arrive at the answer without using python libraries.)

Programming assignments: The next two problems involve programming. We will be considering three datasets (derived from two available datasets) for these assignments:

(a) `Boston`: The Boston housing dataset comes pre-packaged with `scikit-learn` (sklearn.datasets.load_boston). The dataset has 506 points, 13 features, and 1 target (response) variable. You can find more information about the dataset here:
`https://www.kaggle.com/c/boston-housing/overview`

While the original dataset is for a regression problem, we will create two classification datasets for the homework. Note that you only need to work with the `response` r to create these classification datasets.

i. `Boston50`: Let τ_{50} be the median (50th percentile) over all r (response) values. Create a 2-class classification problem such that $y = 1$ if $r \geq \tau_{50}$ and $y = 0$ if $r < \tau_{50}$. By construction, note that the class priors will be $p(y = 1) \approx \frac{1}{2}, p(y = 0) \approx \frac{1}{2}$.

ii. `Boston25:` Let τ_{25} be the 25th percentile over all r (response) values. Create a 2-class classification problem such that $y = 1$ if $r \geq \tau_{25}$ and $y = 0$ if $r < \tau_{25}$. By construction, note that the class priors will be $p(y = 1) \approx \frac{3}{4}, p(y = 0) \approx \frac{1}{4}$.

(b) `Digits`: The `Digits` dataset comes prepackaged with `scikit-learn` (sklearn.datasets.load_digits). The dataset has 1797 points, 64 features, and 10 classes corresponding to ten numbers $0, 1, \ldots, 9$. The dataset was (likely) created from the following dataset:
`https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits`

The 2-class classification datasets from `Boston50`, `Boston25`, and the 10-class classification dataset from `Digits` will be used in the following two problems.

3. **(30 points)** We will consider three methods from `scikit-learn`: `LinearSVC`, `SVC`, and `LogisticRegression`. Use the following parameters for these methods:
```
LinearSVC: max_iter=2000
SVC: gamma='scale', C=10
LogisticRegression:  penalty='l2', solver='lbfgs', multi_class='multinomial',
max_iter=5000
```

(i) (15 points) Develop code for `my_cross_val(method,`X`,y,`k`)`, which performs k-fold cross-validation on $(X, \mathbf{y})$ using `method`, and returns the error rate in each fold. Using

[3]For this problem, you can use python libraries for the computations.

`my_cross_val`, report the error rates in each fold as well as the mean and standard deviation of error rates across folds for the three methods: `LinearSVC`, `SVC`, and `LogisticRegression`, applied to the three classification datasets: `Boston50`, `Boston25`, and `Digits`.

You will have to submit (a) **code** and (b) **summary of results** for `my_cross_val`:

(a) **Code**: You will have to submit code for `my_cross_val(method,`X`,`$\mathbf{y}$`,`k`)` (main file) as well as a wrapper code `q3i()`.

The main file has **input**: (1) `method`, which specifies the (class) name of one of the three classification methods under consideration, (2) X,$\mathbf{y}$, which is data for the 2-class or 10-class classification problem, (3) k, the number of folds for cross-validation, and **output**: (1) the test set error rates for each of the k folds.

The wrapper code has no input and is used to prepare the datasets, and make calls to `my_cross_val(method,`X`,`$\mathbf{y}$`,`k`)` to generate the results for each dataset and each method. Make sure the calls to `my_cross_val(method,`X`,`$\mathbf{y}$`,`k`)` are made in the following order and add a print to the terminal before each call to show which method and dataset is being used:

1. `LinearSVC` with `Boston50`; 2. `LinearSVC` with `Boston25`; 3. `LinearSVC` with `Digits`,

4. `SVC` with `Boston50`; 5. `SVC` with `Boston25`; 6. `SVC` with `Digits`,

7. `LogisticRegression` with `Boston50`; 8. `LogisticRegression` with `Boston25`; 9. `LogisticRegression` with `Digits`.

For example, the first call to `my_cross_val(method,`X`,`$\mathbf{y}$`,`k`)` with $k = 10$ should result in the following output:

```
Error rates for LinearSVC with Boston50:
Fold 1:  ###
Fold 2:  ###
...
Fold 10:  ###
Mean:  ###
Standard Deviation:  ###
```

(b) **Summary of results**: For each dataset and each method, report the test set error rates for each of the $k = 10$ folds, the mean error rate over the k folds, and the standard deviation of the error rates over the k folds. Make a table to present the results for each method and each dataset (9 tables in total). Include a column in the table for each fold, and add two columns at the end to show the overall mean error rate and standard deviation over the k folds. For example:

Error rates for LinearSVC with Boston50											
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Mean	SD
#	#	#	#	#	#	#	#	#	#	#	#

(ii) (15 points) Develop code for `my_train_test(method,`X`,`$\mathbf{y}$`,`π`,`k`)`, which performs random splits on the data $(X, \mathbf{y})$ so that $\pi \in [0, 1]$ fraction of the data is used for training using `method`, rest is used for testing, and the process is repeated k times, after which the code returns the error rate for each such train-test split. Using `my_train_test`, with $\pi = 0.75$ and $k = 10$, report the mean and standard deviation of error rate for the three methods: `LinearSVC`, `SVC`, and `LogisticRegression`, applied to the three classification datasets: `Boston50`, `Boston25`, and `Digits`.

You will have to submit (a) **code** and (b) **summary of results** for `my_train_test`:

(a) **Code**: You will have to submit code for `my_train_test(method,`X`,`$\mathbf{y}$`,`π`,`k`)` (main file) as well as a wrapper code `q3ii()`.
This main file has **input**: (1) `method`, which specifies the (class) name of one of the three classification methods under consideration, (2) X,$\mathbf{y}$, which is data for the 2-class or 10-class classification problem, (3) π, the fraction of data chosen randomly to be used for training, (4) k, the number of times the train-test split will be repeated, and **output**: (1) the test set error rates for each of the k folds printed to the terminal.
The wrapper code has no input and is used to prepare the datasets, and make calls to `my_train_test(method,`X`,`$\mathbf{y}$`,`π`,`k`)` to generate the results for each dataset and each method (9 combinations in total). Make sure the calls to `my_train_test(method,`X`,`$\mathbf{y}$`,`π`,`k`)` are made in the following order and add a print to the terminal before each call to show which method and dataset is being used:
1. `LinearSVC` with `Boston50`; 2. `LinearSVC` with `Boston25`; 3. `LinearSVC` with `Digits`,
4. `SVC` with `Boston50`; 5. `SVC` with `Boston25`; 6. `SVC` with `Digits`,
7. `LogisticRegression` with `Boston50`; 8. `LogisticRegression` with `Boston25`;
9. `LogisticRegression` with `Digits`.

(b) **Summary of results**: For each dataset and each method, report the test set error rates for each of the $k = 10$ runs with $\pi = 0.75$, the mean error rate over the k folds, and the standard deviation of the error rates over the k folds. Make a table to present the results for each method and each dataset (9 tables in total). Include a column in the table for each run, and add two columns at the end to show the overall mean error rate and standard deviation over the k runs.

4. **(30 points)** The problem considers a preliminary exercise in 'feature engineering' with focus on the `Digits` dataset. Represented as $(X, \mathbf{y})$, the `Digits` dataset has $X \in \mathbb{R}^{1797\times 64}$, i.e., 1797 training points, each having 64 features, and $\mathbf{y} \in \{0, 1, \ldots, 9\}^{1797}$, i.e., 1797 training labels with each $y_i \in \{0, 1, \ldots, 9\}$. We will consider three methods from `scikit-learn`: `LinearSVC`, `SVC`, and `LogisticRegression` for this problem. Use the following parameters for the different methods mentioned:

```
LinearSVC: max_iter=2000
SVC: gamma='scale', C=10
LogisticRegression:  penalty='l2', solver='lbfgs', multi_class='multinomial',
max_iter=5000
```

(i) (15 points) For the `Digits` dataset, starting with $X \in \mathbb{R}^{1797\times 64}$, you will create a new feature representation $\tilde{X}_1 \in \mathbb{R}^{1797\times 32}$ as follows: Construct a (random) matrix $G \in \mathbb{R}^{64\times 32}$ where each element $g_{ij} \sim N(0, 1)$, i.e., sampled independently from a univariate normal distribution, and then compute $\tilde{X}_1 = XG$. Using $(\tilde{X}_1, \mathbf{y})$, perform 10-fold cross-validation[4] using the three methods: `LinearSVC`, `SVC`, and `LogisticRegression`, and report the mean and the standard deviation of the 10-fold test set error rate.[5] The

[4]Please use your own code `my_cross_val` for this problem.

[5]Since G is a random matrix, every time you generate G and repeat the procedure, your results will be a bit different.

creation of $\tilde{X}_1$ will be done based on a function `rand_proj`(X, d), where $d = 32$ for this problem, and the function will return $\tilde{X}_1$.

(ii) (15 points) For the `Digits` dataset, starting with $X \in \mathbb{R}^{1797\times 64}$, you will create a new feature representation $\tilde{X}_2 \in \mathbb{R}^{1797\times 2144}$ as follows: For any training data $\mathbf{x}_i \in \mathbb{R}^{64}$, let the elements be $x_{ij}, j = 1, \ldots, 64$. The new feature set for $\tilde{x}_i \in \mathbb{R}^{2144}$ will include:

- all the original features $x_{ij}, j = 1, \ldots, 64$, total 64 features,
- squares of the original features $x_{ij}^2, j = 1, \ldots, 64$, total 64 features. and
- cross-products of all the original features $x_{ij}x_{ij'}, j < j', j = 1, \ldots, 64, j' = j + 1, \ldots, 64.$, total of $\binom{64}{2}$ features.

You should verify that the new features vector $\tilde{x}_i \in \mathbb{R}^{2144}$ and hence $\tilde{X}_2 \in \mathbb{R}^{1797\times 2144}$. Using $(\tilde{X}_2, \mathbf{y})$, perform 10-fold cross-validation[4] using the three methods: `LinearSVC`, `SVC`, and `LogisticRegression`, and report the mean and the standard deviation of the 10-fold test set error rate. The creation of $\tilde{X}_2$ will be done based on a function `quad_proj`(X), and the function will return $\tilde{X}_2$.

You will have to submit (a) **code** and (b) **summary of results** for all three parts:

(a) **Code**: You will have to submit code for `rand_proj`(X,d), `quad_proj`(X) as well as a wrapper code `q4()`.

`rand_proj`(X,d) has **input**: (1) X, which is data (features) for the classification problem, (2) d, the dimensionality of the projected features, and **output**: (1) $\tilde{X} \in \mathbb{R}^{1797\times d}$, the new data for the problem. This output array does not need to be printed to the terminal.

`quad_proj`(X) has **input**: X, which is data (features) for the classification problem, and **output**: (1) $\tilde{X}_2$, the new data with all linear and quadratic combinations of features as described above. This output array does not need to be printed to the terminal.

The wrapper code has no input and uses these above functions to execute all the classification exercises outlined in (i) and (ii) above and print the test set error rates for each of the k folds to the terminal. Make sure the exercises are executed in the following order and add a print to the terminal before each execution to show which method and dataset is being used:

1. `LinearSVC` with $\tilde{X}_1$; 2. `LinearSVC` with $\tilde{X}_2$,
3. `SVC` with $\tilde{X}_1$; 4. `SVC` with $\tilde{X}_2$,
5. `LogisticRegression` with $\tilde{X}_1$; 6. `LogisticRegression` with $\tilde{X}_2$.

(b) **Summary of results**: For each dataset, i.e., $\tilde{X}_1$ and $\tilde{X}_2$, and each method, report the mean error rate over the k folds, and the standard deviation of the error rates over the k folds. Make a table to present the results for each method and each dataset (6 tables in total). Include a column in the table for each fold, and add two columns at the end to show the overall mean error rate and standard deviation over the k folds.

Additional instructions: Code can only be written in Python (**not** IPython notebook); no other programming languages will be accepted. One should be able to execute all programs from python command prompt. Your code must be run on a CSE lab machine (e.g., csel-kh1260-01.cselabs.umn.edu). Please make sure you specify the version of Python you are using as well as instructions on how to run your program in the README file. Information on the size of the

datasets, including number of data points and dimensionality of features, as well as number of classes can be readily extracted from the datasets in `scikit-learn`. Each function must take the inputs in the order specified in the problem and display the output via the terminal or as specified.

For each part, you can submit additional files/functions (as needed) which will be used by the main file. Please put comments in your code so that one can follow the key parts and steps in your code.

Follow the rules strictly. If we cannot run your code, you will not get any credit.

- **Things to submit**

  1. hw1.pdf: A document which contains the solution to Problems 1, 2, 3, and 4 including the summary of results for 3 and 4. This document must be in PDF format (no word, photo, etc. is accepted). If you submit a scanned copy of a hand-written document, make sure the copy is clearly readable, otherwise no credit may be given.
  2. Python code (main.py and other necessary function files) for Problems 3 and 4.
  3. README.txt: README file that contains your name, student ID, email, instructions on how to run your code, any assumptions you are making, and any other necessary details.
  4. Any other files, except the data, which are necessary for your code.