Assignment 2: Machine Learning

Building on our knowledge of data processing and feature extraction, this assignment focuses on machine learning as it relates to classification and evaluation to support sketch recognition.

Part A

In part a, you will use the features you generated from the set of sketched letters from Assignment 1. We will first clean up the feature list to make it usable for machine learning, and then we will practice running different algorithms with a couple software tools.

Preparing features.csv

Previously, you created *features.csv* which contained data in the following format:

sketch	f01	f02	…
a_1	-0.78	-0.62	…
…	…	…	…
z_20	1.0	0.0	…

We retained the individual sketch labels only for the purposes of evaluation. In a practical application, we don't care which instance of the sketch the example is, only that it is a sketch of a certain type. That is, to train a machine learning classifier, we need to pass labels like "a", "b", "c", and so on rather than numbered instances.

Before proceeding, you'll first need to remove the file labels and retain only the sketch type (called a "label" or "class").

f01	f02	…	**label**
-0.78	-0.62	…	**a**
…	…	…	…
1.0	0.0	…	**z**

You should end up with a set of 520 examples, with 20 of each type of letter. Note that it doesn't matter if you wish to use the title "sketch" or "label", nor does it matter if you keep the column at the beginning or move it to the end. Conventionally, we often put the class type in the final column, and it is easier to use some tools following this convention (for instance, Weka will auto-select the class if you do move it to the last column, saving some effort setting it manually).

You can easily complete this step in Excel or with a short script.

Exercise 1: Weka

About

Weka is a machine learning toolkit developed by the machine learning group at Waikato University in New Zealand. It is a Java-based graphical user interface that makes it easy to generate models and analyze features. You can find more details at the project's home page.

Note: Being that Weka is Java-based and built around more traditional machine learning algorithms, it is not as widely used today for deep learning or production-ready model building. It does support deep learning, including convolutional and memory-based networks, but you will find that tools like scikit-learn, TensorFlow, and PyTorch are more popular. We use it as a gentle introduction to practice different machine learning activities.

Setting Up

Download Weka using the links from the manual, where there are many other good resources.

You may need to install the Java Runtime Environment (JRE) or Java Development Kit (JDK) to use Weka. The JDK is more developer-focused but includes a runtime environment. Weka recommends you have at least version 8, but it will work fine with more recent versions like Java 17 or 18.

Using Weka

The following 10 steps explain all the relevant portions of the interface for our purposes.

Loading Data

1. When you launch Weka, it begins in the Chooser, which gives you a variety of options. We will use the "Explorer" for this assignment.
2. In the Explorer window, you start in the "Preprocess" tab. Here you can load datasets for classification, so click "Open file…" and select your new *features.csv* file.
    a. Make sure to specify the CSV format. ARFF is the default, and is a Weka format for storing feature and label values.
    b. If you ordered the file so that labels are in the last column, Weka will automatically use the correct labels!
3. In the left-hand pane, Weka will show all the attributes, and in the right-hand pane, it will preview their values relative to the class labels, if you put labels in the last column. This is an easy way to glance at a few features, although it's not necessarily that useful on its own.

For Mac users, depending on your version of macOS and the location of your data files, you may need to give Weka disk permissions. This can be done in System Preferences under the Security & Privacy pane in the Privacy tab.

Creating Classifiers

4. Click the "Classify" tab and "Choose" to select a classifier.
    a. You'll see many different types, such as bayesian, function-based, trees, etc.
    b. You can select "ZeroR" under the "rule" classifiers to start with the majority classifier.
5. 10-fold cross-validation is a good way to handle the train/test split.
    a. You can also load a separate file if you manually split the data.
    b. You can also specify a percentage split (80% for training and 20% for testing is common).
6. Click "Start" to build the classifier and get the results.

   a. You may need to set the nominative label to your column containing class names if it was not the last column.
   b. The results appear in the right-hand pane. Different classifiers show different output; for instance, trees will print the tree structure.
7. At the bottom of the right-hand pane will be all the performance metrics and a confusion matrix.
   a. You can select and copy text directly from this pane.
   b. You can right-click on the classifier in the "Result list" on the left to get additional options.
      i. "Save result buffer" saves a file with the text results.
      ii. Weka generates Java objects which can also be saved and imported into other Java programs using the "Save model" feature, although we won't use that in this assignment.

Other Useful Actions

8. In the "Select Attributes" tab, you can select feature subset selection, or other feature selector algorithms.
   a. Using cross-validation here will give you the percentage of times a feature was selected across all runs; otherwise you receive a list of top features.
   b. You may need to select the nominative label (class name).
   c. Right-clicking in the result list gives you the option to visualize the data over the reduced feature set.
9. In the "Cluster" tab, you can use unsupervised learning to group and visualize data.
   a. In addition to being able to choose the algorithm, clicking on its name gives you options to specify parameters like number of clusters.
   b. The results will report how many of each label ended up in the different clusters.
   c. You can also right-click to visualize the cluster assignments. Consider using the top two features from feature selection as the axes if you do!
10. The "Visualize" tab shows all the features plotted against one another.
    a. Clicking on one makes it larger.
    b. This can be useful for quick review of your data, but like the "Preprocess" tab, it's not necessarily that useful by itself.

Requirements for Exercise 1

For this part, you should use Weka to load your *features.csv* file.

Create at least **six** classifiers:

- Majority Classifier (ZeroR)
- Decision Tree
- Random Forest
- Three more of your choosing, with
  - At least 1 function type
  - At least 1 probabilistic (bayes) type

Select attributes to get a list of selected or ranked features.

For each of the classifiers and the feature selection output, save the result buffer to a file named as the classifier or the feature selector you chose.

Exercise 2: Scikit-Learn

About

Scikit-learn (sklearn) is a Python library for machine learning. It uses the very popular NumPy, SciPy, and matplotlib libraries to handle complex mathematical operations and provide quick visualizations. Sklearn is very popular for data science and machine learning; Python in general is widely used in this field because of its good blend of speed and ease of use. Learn more at sklearn's website.

Note: We selected sklearn as a popular module that is easy to use in order to constrain this assignment and provide very relevant practice with hands-on machine learning. It is true that other frameworks like PyTorch or Keras may be more common for deep learning, and there are a plethora of other services and languages with their own libraries for this task. However, we believe sklearn strikes a good balance for our purposes.

Setting Up

The Installation Guide provides many options for setting up sklearn. You can use a downloadable Python distribution and install there, or you can use your OS-installed version (if applicable) to install sklearn into.

For that matter, there are many ways to run Python code with sklearn online or through separate applications, including Google CoLab, OneCompiler, CoCalc, and others. It's also very easy to setup JupyterLab Desktop since it's just a downloadable app which installs a standalone Python environment. You can install sklearn inside JupyterLab using the sklearn instructions.

Using Scikit-learn

Sklearn is nearly as straightforward as a click-button interface since it is so structured and standardized. You can do all the same activities as Weka and a lot more, owing to being built on the more popular Python (for data science and machine learning) and being more actively developed.

Here is an example script showcasing some sklearn functionality on the iris dataset. Below we explain the main operations you are likely to use.

Loading Data with Pandas

The easiest way to handle CSV files in Python is with the *pandas* library. Pandas provides documentation for installing if you need to. It's already installed in JupyterLab environments, and in most others, it's as simple as running "pip install pandas".

Once installed, you can import a CSV file with the `read_csv()` function. If you have a header row, you can access columns using those labels.

It's a good idea to shuffle the data after loading in case your classes are all grouped (to avoid removing only one class in a train/test split).

Creating Classifiers

Sklearn documentation does not provide a simplified list of all classifiers, but you can find a list in this Stack Overflow answer. Dozens of machine learning models are available in their respective subpackages (much like weka had folders for "trees", "functions", and so on).

To build a decision tree, you would do the following:

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
```

This creates the object called "clf" to be of type Decision Tree. Training the classifier is simple: you just call "clf.fit(X,y)" where X and y are your set of features and labels, respectively.

The standardization of sklearn can save a lot of time and effort. Every classifier implements the `.fit()` method, so training any other classifier is as easy as importing that classifier definition and instantiating the object to use it. All the other syntax is unchanged!

Note that the majority classifier in sklearn is called the DummyClassifier and may be accessed with "from sklearn.dummy import DummyClassifier".

Also, you can use either a train/test split with your data, via train_test_split from model_selection, or cross validation, via cross_validate from model_selection. There may be a small syntax change depending on the method you choose.

Other Useful Actions
Sklearn.metrics contains all the scoring capabilities. In general, you'll call `.predict()` on the test data to generate the labels that the model thinks are correct. These predicted labels can be compared with the actual labels to determine true positive, false positive, and other values and their associated metrics. The "classification_report" is an easy way to print a lot of information in one line, as is printing or displaying the confusion matrix.

There are many methods for feature selection available. The example code uses a tree classifier to determine the selected features and get their relative importance directly from the model, but there's no particular reason why it was used over any other.

Requirements for Exercise 2

For this part, you should use sklearn to load your *features.csv* file.

Create at least **seven** classifiers:

- Majority Classifier (dummy)
- Decision Tree
- Random Forest
- Neural Net (MLPClassifier)
- Naive Bayes (GaussianNB)
- Two more of your choosing

Select the top features by removing less useful ones or compute the relative feature importances.

You will use the "pickle" module to save your trained models to files for the autograder. Reference the Summary and How to Submit section at the end for details.

Part B

Part b will consist of several exercises practicing specific considerations when doing machine learning.

Exercise 3: Extended Feature Set

Extended Feature Set Download.

The download contains a CSV with the letter data including 45 features in total. Each feature is labeled for your reference. It is essentially a subset of the feature library listed in the *The Power of Automatic Feature Selection: Rubine on Steroids* paper. A number of features have been removed since the letter data lack pressure information, and being single-stroke, have no multi-stroke locality to consider.

Run at least 3 different classifiers on this data and perform feature selection. List the classifiers used and their f-measures, as well as the top 3 selected features in your report.

How does this result compare to your findings from part a?

Read the article "Overfitting, Variance, Bias, and Model Complexity in Machine Learning." Be aware that there is an error with the top chart on that page; it is supposed to be on the bottom. For more information about overfitting, Amazon has helpful documentation. Microsoft has a nice article on feature selection as well, although only the first section, "Why Do Feature Selection?", is useful in this case since the rest is documentation for their proprietary software.

Based on those readings, discuss what your results for this exercise relative to the earlier results mean. In particular, address the issue of more features vs. data (note that we have the same amount of data, just more features), and the impact of model complexity on overfitting or underfitting behaviors. Give one or two options to balance these considerations.

Exercise 4: Gesture Recognizer

Music Player Gesture Features Download.

Let's imagine that you are working on software for an interactive, gesture-based music player. The player supports a few basic gestures to trigger different commands:

- Swipe back means go back one song.
- Swipe forward means skip to the next song.
- Swipe down is play and pause.
- A circular gesture means to repeat the current track.

We define the back and forward directions based on the behavior of touch screen gestures: movement from left to right is back, while movement from right to left is forward. This UI design has a long history, likely rooted in English reading order being left to right. New content loads in from the right, so on a touch screen, manipulating that object off the screen to move back to old content means moving it out from the left.

For the purposes of this exercise, it doesn't actually matter the modality of the recognizer. Perhaps it runs on a touchscreen with 2D gestures, or it uses a camera or motion tracker to capture 3D gestures made in the air. In any event, the original coordinates are not provided, only the resulting features.

Run at least 3 different classifiers on this data and perform feature selection. List the classifiers used and their f-measures, as well as the top 3 selected features in your report.

Watch this 5-minute video on Machine Learning vs. Heuristics.

In that context, discuss your observations for how the classifiers performed for this data.

Notice that we did not provide the actual gesture labels, just gesture1, gesture2, etc. Can you identify which gesture is which action (back, forward, pause, repeat)? Use your knowledge of sketch attributes like angles and time! What heuristics could you employ to build this recognizer without running a full machine learning algorithm?

Exercise 5: Children's Shape Drawings in Sklearn

Shape Features Download.

For this exercise, you are given features for a set of children's simple shape drawings. Even though you are not given the original data, these are very basic geometric shapes that we have encountered in previous activities like worksheets, so you can hypothesize about what the shapes are just based on Rubine feature values!

The task for this exercise is to use sklearn to create the best classifier that you can. You'll want to use shuffling and either cross validation or train/test splits. Once you are satisfied with your classifier, you'll return it in the specified format, and the autograder will run your classifier on the hidden validation set.

You may remove features to improve model accuracy, but be sure to set the feature mask array so that the autograder can remove the same features.

In your report, mention what classifiers you tried and ultimately chose, as well as any feature selection or other optimization you tried. Based on the feature values, what do you guess the shapes to be for shape1, shape2, and so on up to shape5?

You will use the "pickle" module to save your trained model to a file for the autograder. Reference the Summary and How to Submit section at the end for details.

Summary and How to Submit

In this assignment, you are building on your work from Assignment 1 to practice machine learning. If you want to practice more, consider Google's Machine Learning Courses. Google provides a really good collection of materials for using TensorFlow for deep learning, with many helpful resources for general machine learning.

You can work in groups of up to 4. The same guidance applies as in the previous assignment with regards to including a contribution breakdown in the report and submitting a peer evaluation at the end.

Each member of a team submission must also complete a peer evaluation form. Link TBD.

Part A

Part a includes two exercises with the features.csv from the last assignment:

- For exercise 1, use Weka to create 6 classifiers and apply feature selection/ranking. Save the results buffer from each into a folder called "ex1".
- For exercise 2, use sklearn to create 7 classifiers and apply feature selection/ranking. Save the source code that generates your models, and use the "pickle" module to save each model inside a folder called "ex2". For your classifiers, name them according to the sklearn class. Below is an example of how to save the DummyClassifier to a file. Effectively, all you need to do is use pickle.dump() on each classifier. This saves the Python object to a file.

```
# assume you have a DummyClassifier object:
majority = DummyClassifier()
# after training with .fit() or using cross validation, save it with:
import pickle
pickle.dump(majority, open('DummyClassifier.sav','wb'))
```

Part B

Part b is a collection of activities practicing related concepts.

- In exercise 3, you're given an expanded feature set for the letter data. You must run 3 different classifiers and feature selection on that data. You may use whatever platform you wish. In the interest of submitting sources, the autograder does have 1 point set aside to check if there is any content in the "ex3" folder. This could be a result buffer from weka, sklearn source, or other file. In your report, be sure to list the classifiers, their f-measures, and the top 3 selected features. Discuss the results compared to what you got with just the Rubine features, in the context of the linked article.
- Exercise 4 provides a set of gesture features. As in exercise 3, run 3 classifiers and feature selection, reporting the f-measures and top 3 features in your report. Likewise, the autograder uses 1 point to check that some content / sources are present in "ex4". Discuss these results in the context of machine learning and heuristics per the video.
- For exercise 5, you should generate the best sklearn classifier you can for children's shape drawing data. You don't have the original data or the validation set (which will be run by the autograder), so the goal is to build the best classifier based on the data you do have and return it in the autograder's format. Mention your process and results on the provided data in your report. Use the "pickle" module to save your classifier to the "ex5" folder; it must end with the ".sav" extension.

A blank copy of the submission format is available for download. It is a folder called "submission" which contains folders ex1, ex2, ex3, ex4, and ex5. Zip and submit the submission folder after filling the contents for each exercise in its respective folder. Teams may want to recreate this folder structure in Google Drive for easier collaboration.

Add your report directly to the "submission" folder as a PDF, DOCX, RTF, or other common format.

Submit to Gradescope.