PIC 16, Winter 2018 – Preparation 10W

Assigned 3/9/2018. To be completed by class 3/14/2018.

Intended Learning Outcomes

By the end of this preparatory assignment, students should be able to:

- explain the meaning of and apply in conversation essential terms of the machine learning / scikit-learn vocabulary, including samples, features, targets, supervised learning, classification, regression, unsupervised learning, clustering, training set, testing set, estimator, and predict;
- load example data from the included datasets (which are useful for trying new scikit-learn features);
- given the class of an estimator, any required parameters, training data, and test data, instantiate an estimator, `fit` the estimator to the training data, and use the trained estimator to `predict`;
- `pickle` an estimator so that it can be revived without having to train it again; and
- reshape data to the `n_samples`, `n_features` shape required by scikit-learn.

Tasks

- ☐ Scikit-learn's documentation is quite good. I think it provides a great intro to both the subject of machine learning and actually using scikit-learn to use machine learning algorithms. Please read the following carefully. Sometimes I (accidentally) turn off my brain and read the words of complex passages without thinking about them. Actively avoid this. Instead, click links as needed and look up terms you are having trouble understanding. It really does make sense, and I am confident that everyone can understand what the tutorial is saying, regardless of mathematical background. If you run into trouble with something, please ask me for clarification!
- ☐ After reading "Machine Learning: the Problem Setting", consider the following. Some students want to use measurements of an individual's walking gait (motion) as a sort of biometric signature, like a fingerprint or retina scan. One of the goals is to create an app that can identify which of a set of people – the owner, or one of his/her close friends (or ultimately, none of them) – has the owner's cell phone in his/her pocket based on acceleration measurements taken by the cell phone. (Note that taking acceleration measurements is a common capability of modern smart-phones.)
  - Is this a supervised learning problem or an unsupervised learning problem?
  - If supervised learning, is it classification or regression?
  - If unsupervised learning, is it clustering or density estimation?
  - What parts of the description gave you the answers to the previous questions?
- ☐ Read Loading an example dataset. The data is just a NumPy array. Try reshaping the first row of `digits.data` to the shape of `digits.images[0]` and use the appropriate NumPy function to confirm that all corresponding elements are close (or the same). Visualize (graphically) a few of the 8×8 arrays in `digits.image` using the appropriate function of `matplotlib`, i.e., you should produce a (preferably grayscale) *image* of a number.
- ☐ Please follow along with Learning and Predicting (entering the code into the console yourself, preferably). Note the line:
  `clf.predict(digits.data[-1:])`
  and how it differs from
  `clf.predict(digits.data[-1])`, and
  `clf.predict(digits.data[-1,:])`
  Why don't the latter two work? Read the error message and see what's different about the arguments passed to `clf.predict` to find out. (It's a NumPy subtlety I wasn't aware of.)

- The digits data set is quite large. The example has you training using $M = 1795$ samples and testing on only one. When learning numbers, did you need to see 100 examples of each before you could recognize the digits 0-9? Modify the program to train using the first $M = L - N$ samples (where L is the total number of samples) and test using the last N samples. Create a plot of the % of test digits accurately classified as a function of the number of digits M used as training data. (It's OK to use a loop or list comprehension here. Also, the computation could take a while, so you can increment by 10 or so.)
- While accuracy tends to improve with more training data, mote that the plot is not strictly monotonic – increasing the amount of training data does not *always* improve the classification accuracy. Another important thing to note is that the particular samples used to train on can also affect the accuracy. For M = 1000, train on a random subset of the samples (see `numpy.random.choice`), test on the remaining samples, and calculate the accuracy. Repeat this process many times and generate a histogram of the accuracy.
- Certain classifiers are affected by the order in which they are trained. Is your classifier's accuracy affected by the order in which the training samples are provided? (See `numpy.random.permutation`).
- After reading "Learning and Predicting", consider the following additional information about the case study introduced previously: to test the feasibility of creating such an app, the students have collected acceleration measurements of the hip of each of 10 individuals during 100 steps of walking (from a cell phone in the individual's pocket). They process the raw data: for each step of an individual, they record the duration of the step and the maximum acceleration. They fit an estimator with these measurements for 80 steps of the individual, providing the estimator with the corresponding individual's name for each step-worth of data. They have the estimator predict the name of the individual carrying the phone for each of the remaining 20 steps-worth of data, and then compare the estimator's prediction with the truth to determine the accuracy of the estimator. In this case:
    - What are the features and what are the targets?
    - If this data were in the same format as the `iris` and `digits` data sets, what would `n_samples` be, and what would `n_features` be (numerically)?
    - Which is the training data, and which is the test data? (the 80 steps-worth, or the remaining 20 steps-worth?)
- Read all of Statistical learning: the setting and the estimator object in scikit-learn. It might help you answer some of the questions above.
- Read Supervised learning: predicting an output variable from high-dimensional observations up to "The curse of dimensionality". The curse of dimensionality is an important concept, but you don't need to read the explanation here if you find it unclear. The concept is simply that as the number of features increases, the number of samples required to train the estimator grows exponentially.
- Read the section about Support Vector Machines (SVMs) from the same page. The only thing you really need to understand is that an SVM tries to perform classification by determining hyperplanes (the multidimensional analog to a 2D plane) that separate samples from the different classes. We'll talk about it in class.
- Read Model Persistence. Training your machine can take a long time. If you want to use it again, you can save it rather than re-training it every time.
- Skim Conventions.