CS273a Project Description

Introduction to Machine Learning: Fall 2016

Due: Wednesday December 7, 2016

For your course project, we will deepen our exploration of data mining and prediction with real-world data. There are two main options:

(a) **Class Kaggle Data:** Use the data set in our class Kaggle competition, `UC Irvine CS273a 2016` competition, at
`https://inclass.kaggle.com/c/uc-irvine-cs273a-2016`
I encourage this option; it is well-defined and will have plenty of support from other students working on related issues.

(b) **External Kaggle:** Participate in a different Kaggle competition of your choice; examples currently available include the Nature Conservancy Fisheries Monitoring, Santander Product Recommendation, Outbrain Click Prediction, and Allstate Claims Severity competitions. (**Note:** please do not select "toy" or "getting started" competitions, only sponsored ones.) However, note that some competitions have significant data processing components (extracting features from images, etc.) so make sure you can make some progress easily before you commit to one of these.

If you have a personal project you would like to do instead, please speak to me about it after class; this is also valid but should not overlap with projects for other classes.

Step 1: Form teams.

Please form teams of 2-3 students who share your interests and with whom you will directly collaborate. Piazza can help with locating other students in need of joining at team.

There is no barrier to collaboration on this project, so feel free to talk to other teams about what they are doing, how they are doing it, and how well it works (and you can see this last point on the leaderboard as well). A significant component of good performance can boil down to feature design and choices, so you may want to talk to other teams about their choices and how it affects performance. You are also free to use online code, or other sources. However, please make sure that your own entries are developed by your team members – the purpose of this project is to give you practical experience, so it will not be helpful if you just follow instructions from a friend.

Step 2: Download the data.

For our in-class data, you should have already done this from a previous homework; for example,

```
Xtr = np.loadtxt("X_train.txt")
Ytr = np.loadtxt("Y_train.txt")
# also load features of the test data (to be predicted)
Xte = np.loadtxt("X_test.txt")
```

Step 3: Choose a technique (or more) to explore

Select as a focus of your attempts some aspect of prediction that we have not already done in depth. There are a nearly infinite number of possible avenues of exploration; a good guideline is to identify

a research paper that proposes a method you think could be helpful or effective, and explore its use. Some examples of techniques you might focus on include:

- Semi-supervised methods: investigate how your knowledge of the test features can be used to improve prediction. As examples, see e.g., label propagation (`http://www.cs.cmu.edu/~zhuxj/pub/CMU-CALD-02-107.pdf`, or using EM (withing e.g. naïve Bayes or a Gaussian mixture model, e.g., `http://www.kamalnigam.com/papers/emcat-mlj99.pdf`).
- Kernel learning, or "learning" the measure of dissimilarity used in, for example, nearest neighbors or SVMs, to improve their performance. See for example Weinberger and Saul 2008, `http://www.cse.wustl.edu/~kilian/papers/jmlr08_lmnn.pdf`.
- Neural networks and deep learning; see e.g. `http://deeplearning.net/tutorial/`. PyLearn2 is a toolkit for deep learning which is quite sophisticated but not trivial to use.
- Support vector machines. (For example, you could investigate the effect of different kernel choices, regularization, etc.) The implementation `libsvm` is pretty good.
- Go in-depth with ensembles. Use lots of learners, stacking, and information from your leaderboard performance to try to improve your prediction quality.
- Feature selection methods, such as stepwise regression (or in this case, classification); e.g. `http://en.wikipedia.org/wiki/Stepwise_regression`. (Note: if you use feature selection, you should use a predictor that is sufficiently complex to *need* feature selection!)
- Techniques for creating new features, including "kitchen sink" features (`http://books.nips.cc/papers/files/nips21/NIPS2008_0885.pdf`), clustering-based features, etc. Once you have many features, of course, you may also have to explore feature selection (see above) or regularization to control complexity.
- More sophisticated decision tree structures, e.g., `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.1587`.
- and many more...

You may use any of the code provided from class; online code packages such as Weka, PMTK, PyLearn2, or others; or write your own.

However, a key point is that you must *explore* your approach(es); so you must do more than simply download a publicly available package and run it with default settings, or trying a few values for regularization or other basic parameter settings. You must at least explore the method fully enough to understand how changes might affect its performance, verify that your findings make sense, and then use your findings to optimize your performance; ideally, you should repeat this process several times, exploring several different ideas. In your report, you should describe why you decided to explore this aspect, what you expected to find, and how your findings matched (or didn't match) your expectations.

Step 3: Build your learners

Use your selected focus, along with the techniques we have developed so far in class, to construct predictive models for your target(s).

Be aware of the positive and negative aspects of the learners we have discussed. For example, nearest neighbor methods can be very powerful, but can also be very slow for large data sets; a similar statement applies to dual-form SVMs. For such learners, dealing with the large data set may be a significant issue – perhaps you could reduce the data in some way without sacrificing performance? On the other hand, linear methods are very fast, but may not have enough model complexity to provide a good fit. For such learners, you may need to try to generate better features, etc.

Step 4: Evaluate; go to Step 3.

Output your predictions and evaluate them on Kaggle:

```
# make predictions & extract class-1 probabilities
Yte = learner.predictSoft( Xte )[:,1];
# Note: be sure Yte is a flat vector, shape (m,)
# otherwise, reshape it using e.g.
#   Yte = Yte.ravel()
# or change the indexing in the code below:

fh = open('predictions.csv','w')    # open file for upload
fh.write('ID,Prob1\n')              # output header line
for i,yi in enumerate(Yte):
  fh.write('{},{}\n'.format(i,yi))  # output each prediction
fh.close()                          # close the file
```

You can check the leaderboard to see your "test" performance.

NOTE: You're limited to a few (≈ 2) submissions per day to avoid over-loading their servers and to enforce good practices. Thus, you should not try to upload every possible model with every possible parameter setting – use validation data, or cross-validation, to assess *which* models are worth uploading, and just use the uploads to verify your performance on those. You can use the `auc` and `roc` functions (in the base classifier class) to estimate your AUC performance or visualize the ROC curve.

Step 5: Write it up

Your team will produce a single write-up document, approximately **6 pages** long, describing the problem you chose to tackle and the methods you used to address it, including which model(s) you tried, how you trained them, how you selected any parameters they might require, and how they performed in on the test data. Consider including tables of performance of different approaches, or plots of performance used to perform model selection (i.e., parameters that control complexity).

Within your document, please try to describe to the best of your ability who was responsible for which aspects (which learners, etc.), and how the team as a whole put the ideas together.

You are free to collaborate with other teams, *including* sharing ideas and even code, but please document where your predictions came from. (This also relaxes the proscription from posting code or asking for code help on Piazza, at least for project purposes.) For example, for any code you use, please say in your report who wrote the code and how it was applied (who determined the parameter settings and how, etc.) Collaboration is particularly true for learning ensembles of predictors: your teams may each supply a set of predictors, and then collaborate to learn an ensemble from the set.

Requirements / Grading

I am looking for several elements to be present in any good project. These are:

(a) Exploration of *at least* one or two techniques on which we did not spend significant time in class. For example, using neural networks, support vector machines, or random forests are great ideas; but if you do this, you should explore in some depth the various options available to you for parameterizing the model, controlling complexity, etc. (This should involve more than simply varying a parameter and showing a plot of results.) Other options might include feature design, or optimizing your models to deal with special aspects of the data (positivity

of predictions; large numbers of zeros in the data; possible outlier data; etc.). Your report should describe what aspects you chose to focus on.

(b) Performance validation. You should practice good form and use validation or cross-validation to assess your models' performance, do model selection, combine models, etc. You should *not* simply upload hundreds of different predictors to the website to see how they do. Think of the website as "test" performance – in practice, you would only be able to measure this once you go live.

(c) Adaptation to under- and over-fitting. Machine learning is not very "one size fits all" – it is impossible to know for sure what model to choose, what features to give it, or how to set the parameters until you see how it does on the data. Therefore, much of machine learning revolves around assessing performance (e.g., is my poor performance due to underfitting, or overfitting?) and deciding how to modify your techniques in response. Your report should describe how, during your process, you decided how to adapt your models and why.