46-927, Fall 2017: Homework #4

Due 5:20PM, Monday, February 12, 2018.

Homework should be submitted electronically on canvas before the deadline. Please submit your homework as a single file (pdf or html). We recommend using a jupyter notebook to prepare your homework.

Note: For this homework, try to do all problems in python.

Please post any questions on the Piazza discussion board.

1. *Best classification when your losses are asymmetric.* Consider a two-class ($\{0, 1\}$) classification problem. We saw in class that for 0-1 loss, the optimal classification rule is

$$f(x) = \begin{cases} 1 & \text{if } P\left(Y = 1 \mid X = x\right) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

if we assume that we know everything about the distribution of (X, Y).

Suppose instead that our loss is

	True $Y = 1$	True $Y = 0$
Guess $Y = 1$	0	L_{01}
Guess $Y = 0$	L_{10}	0

(a) Again assuming that we know everything about the distribution, find the best possible classification function $f(x)$ for minimizing this new, more general loss. The calculation will be very similar to the one in class.

(b) If $L_{10} > L_{01}$ (so that false negatives are worse than false positives), how does this classification rule differ from the 0-1 loss rule we derived in class? Why does this make sense? (You can just answer with two short sentences.)

(c) Suppose that you are constructing your classifier using logistic regression. Before, we had a classification rule of

$$\widehat{f}(x) = \begin{cases} 1 & \text{if } x^T \widehat{\beta} > 0 \\ 0 & \text{otherwise} \end{cases}$$

What should this rule become under the new loss? What does this do to the linear decision boundary when $L_{10} > L_{01}$?

To think about: How should your classifiers change if your class frequencies are different between your training and test data?

2. *Prediction with marketing data* For this problem, we will be working with marketing data from a bank, found in `marketing.csv` on blackboard. We observe the results from a calling campaign that attempted to sell a new product to clients. Based on the covariates of the client, we want to build a classifier to predict whether the client will invest in the product.

   We observe the following variables about the client:

   - `age`
   - `job`: Job classification
   - `marital`: Marital status
   - `education`: Level of education
   - `default`: Currently in default?
   - `balance`: Average yearly account balance
   - `housing`: Currently has a housing loan?
   - `loan`: Currently has a personal loan?

   You also observe a variable `y` which is either `no` or `yes`, indicating whether the sale was successful. This is your outcome variable.

   You can load the data into `pythohn` with the `read_csv` command. Familiarize yourself with the variables and data types in this data set.

   Hint: You will notice that we have several categorical variables. In sklearn's logistic regression, these are not handled automatically like they are in `R`. Instead, you need to convert them to dummy variables (also called one-hot encoding. You can do this by calling the `pd.get_dummies` function on your data frame, which returns a dummy-encoded version of the data frame.

   Similarly, you can recode your `y` variable as $\{0, 1\}$ with

   ```
   y = np.where(dat['y'] == 'yes',1,0)
   ```

   or just use a column returned from the dummy call.

   We will also split the data into a training set and a test set. For uniformity and ease of grading, use the following code to generate your training and test data:

```
train_test_split(X, y, test_size=0.33, random_state=1)
```

This splits off 33% of the data to be used as a validation set.

(a) Using the training data, estimate the fraction of successes (your *base rate*). This will be useful to keep in mind.

(b) Using your training data, fit a logistic regression using all of the variables to predict the outcome. Use `LogisticRegression` from `sklearn`, with `C=100000` to downweight the ridge penalty. What is your misclassification rate (fraction of wrong guesses in any direction) on the test set, using the usual logistic regression classification rule?

> **Hint:** You can get three different kinds of "predictions" from logistic regression. Suppose that `fit` is the object returned by a call to `LogisticRegresssion().fit()`. Then:
>
> - `fit.predict` gives class labels in $\{0, 1\}$
> - `fit.predict_proba` gives probabilities on the p(x) scale. Note that this returns probabilities of both class 0 and class 1, so
>
>   `fit.predict_proba(X_test)[:,1]`
>
>   corresponds to the vector of $P(Y = 1|X = x_i)$.
> - `fit.predict_log_proba` similarly gives the log probabilities. This is sometimes more convenient.

(c) If you used the silly classifier that guesses `'no'` for all observations, what would your misclassification rate be on the test data? How does this compare to the misclassification rate of your logistic classifier?

This might be rather discouraging for our classifier, if our true goal is minimizing 0-1 loss.

(d) Suppose that, despite this hopelessness, you still feel the need to call some clients. Using your logistic model, find the 1000 clients in the test set that are most likely to score `'yes'`. (That is, the 1000 observations from the test set with the highest predicted probability of $Y = 1$.) For this set of 1000 clients, what fraction of them actually say yes? What fraction would you expect if you picked a random set of 1000 clients?

Suddenly our model feels a little more useful!

Hint: You may find the `np.argsort` and/or `np.flip` functions useful.

(e) Even though our simple classifier doesn't win in terms of 0-1 loss, we see that it helps to sort our points according to their likelihood of being `'yes'`. This can be a much more reasonable goal than misclassification error in heavily imbalanced samples. You can think of this as an approach to selecting an "enriched" set.

As you change the size of your selected set (500, 1000, 2000, etc), you change both:

- The fraction of true positives in the data that you manage to capture (called *sensitivity*)
- The fraction of true negatives you manage to avoid (called *specificity*)

When the set gets larger, sensitivity will necessarily increase, but specificity will decrease. The better your classifier, the more efficiently it makes this trade off.

An easy way to assess the quality of your classifier in this sense is to plot sensitivity versus specificity. A better classifier will tend to have a higher curve (better sensitivity for a given specificity). These curves (called ROC curves) have other nice properties that we will discuss in class; This problem is just intended as an introduction.

The `roc_curve` function from `sklearn.metrics` will compute TPR (the same as sensitivity) and FPR (the same as 1-specificity). It returns a tuple, of which the first two elements are FPR and TPR. Example usage:

```
fpr, tpr, _ = roc_curve(true_y_values, predicted_probabilities)
```

(The _ discards the third returned value, which we don't care about yet)

Using the `roc_curve` function, plot the TPR (y axis) against the FPR (x axis). This is your ROC curve.

If you classified by guessing randomly, on average you would just get the diagonal $y = x$ line; if your classifier is better than random guessing it should give a curve above this line. Include a diagonal $y = x$ line in your plot for comparison.

We will talk more about these measures and plots in our next class.

3. *Marketing data, continued.* Now we will try to improve your marketing classifier. Start where you left off in the last problem, using the same data sets.

(a) Two of our variables, balance and age, are continuous. Furthermore, we doubt that the log odds of successfully advertising do not actually grow linearly in

either of these variables. Fit a logistic GAM to the data, using smoothing splines to model the effect of balance and age. Plot the partial dependence plots for these two variables.

(b) Evaluate the predictions from your logistic GAM in terms of misclassification. Also calculate the fraction of successes in the top ranked 1000 test observations. How does your performance on each of these metrics compare to logistic regression?

(c) Make a new ROC curve plot. This plot should have three curves: (i) the straight $y = x$ line; (ii) The ROC curve for logistic regression from the previous problem; (iii) The ROC curve for your new logistic GAM.