INFO371 Problem Set 4

Your name:

Deadline: Wed, Feb 21, 10:30am

Introduction

The deadline appears to be short, but the problems are easier this time (at least I hope ;-)). Please submit a) your code (notebooks, rmd, whatever) *and* b) the results in a final output form (html or pdf). You are free to choose either R or python for solving this problem set. In case of python, I recommend to use *scikit-learn.linear_model*. In case of R, you have to find a package that includes logistic regression with regularization. You may consider *LiblineaR* or *glmnet*. There are probably more but I am not familiar with those. There are many packages that implement kNN in R, I have good experience with *FNN* library.

You are welcome to answer some of the questions on paper but please include the result as an image in your final file. Note that you can easily include images in both notebooks and .rmd—besides of the code, both are just markdown documents.

Working together is fun and useful but you have to submit your own work. Discussing the solutions and problems with your classmates is all right but do not copy-paste their solution! Please list all your collaborators below:

1.
2. ...

Wisconsin Breast Cancer Dataset

You will work with Wisconsin Breast Cancer Dataset (WBCD), available at UCI Machine Learning Repository. I recommend to dowload the files *wdbc.csv.bz2* and *wdbc_doc.txt* from canvas (under *files/data*). The first one is the csv with variable names, the second one a brief description of the data.

The data includes diagnosis of the tumor with "M" meaning cancer (malignant) and "B" no cancer (benign), and 10 features, describing physical properties of cell nuclei from biopsy samples. Each feature is represented three times, once for mean, once for standard error, and once for the worst values. Your task is to predict diagnosis based on this data.

1 Explore the data

As the first step, explore the data.

1. Load the data. You may drop *id* or just ignore it in the rest of your analysis.
2. Create a summary table where you show means, ranges, and number of missings for each variable. In addition, add correlation between the diagnosis and the corresponding feature. You may add more statistics you consider useful to this table.

3. Graphical exploration. Make a number of scatterplots where you explore the relationship between features and the diagnosis.

   Note: you may attempt to display all 435 possible combinations as a scatterplot matrix, but that will most likely be just unreadable. Choose instead a few cases with higher correlation. Avoid overwhelming your reader with tens of similar figures.

2 Decision Boundary

The first task is to plot the decision boundary by kNN and by logistic regression. You will also play a little with feature engineering.

If you are uncertain about what is decision boundary, I recommend to consult Daume's textbook (see canvas-files-readings) chapter 3, in particular page 34. There are two broad strategies to plot it. In any case, you have first to estimate (train) your model. Thereafter you have to predict the classes (cancer/no cancer here) on a regular dense grid that covers the parameter space. Afterwards you can either plot your predicted values with a certain color code, or alternatively, say, set predicted $\mathsf{M} = 1$ and predicted $\mathsf{B} = 0$, and make a contour plot for a single contour at level 0.5. You may also combine these both methods.

We ignore training/testing/overfitting issues for now.

2.1 kNN Case

First, let's explore the decision boundary using kNN.

Pick two features. I recommend to use a few that show relative strong correlation with diagnosis. Feel free to use a combination you already plotted above.

1. Next, predict the diagnosis on a grid (say, 100x100) that covers the range of the explanatory variables. Use kNN with $\mathsf{k} = 3..7$ (pick just one value). This gives you 100x100 predicted diagnoses.

2. Plot the actual data, and the decision boundary on the same plot. Ensure that actual observations and predictions are clearly distinguishable, and that one can easily understand the color code.

3. Describe your observations. How good is kNN in picking up the actual shape? Does it also pick up noise?

   Note: unless you do cross-validation, you cannot know if the model picks up noise (overfit). Here I just ask your best judgement, not any formal analysis.

2.2 Logistic Regression

Now repeat the process above by logistic regression. Pick the same features as in the case of kNN.

1. Fit a logistic regression model with these two features.

2. Predict the diagnosis on a similar grid...

3. ...and create a similar plot.

4. Describe your observations. How does the result for kNN compare to that for Logistic Regression?

2.3 Feature Engineering

So far you were using just two of the existing features in the data. However, now let's create some more. Use these two features to compute some new ones. Let's denote your original features by $\mathtt{x}$ and $\mathtt{y}$. Examples of new ones you may create include: $\mathtt{x}^2, \mathtt{y}^2, \mathtt{x} \cdot \mathtt{y}, \mathbb{1}(\mathtt{x} > 5), \mathbb{1}(\mathtt{y} < 1) \cdot \mathtt{x}^2, \log \mathtt{x} \ldots$ You can use all sorts of mathematical operations as long as a) you only use $\mathtt{x}$ and $\mathtt{y}$, not other features, and b) the engineered ones remain linearly independent.

1. Fit a logistic regression model. However, this time pick both $\mathtt{x}$, $\mathtt{y}$, and some of your engineered features.
2. Create the decision boundary plot.
3. Comment on the shape of the boundary. What do you think about being able to capture better the actual boundary, and about overfitting?
4. Repeat the exercise a few times where you pick/engineer different new features, and try to get as reasonable boundary as you can.

   As above, I am asking "reasonable" boundary in the sense of your best judgement. No actual cross validation is necessary.

Note: I have mixed experience with `scikit-learn.linear_model.LogisticRegression`. It occasionally appears the default convergence tolerance is far too big, and the default *liblinear* solver too imprecise. Setting tolerance to 10^{-12} and solver to *lbfgs* improved the results for me, but did not make it flawless.

3 Use the full data

Finally, we'll get serious. We use full data set in the logistic regression, include regularization, and cross-validate our results.

5. Split your data into training/testing data (say 80/20%).
6. Fit logistic regression model on training data using all features. You may add engineered features if you wish.
7. Calculate accuracy, precision and recall.
8. Repeat the process a number of times ($\geqslant 10$), each time choosing a new random testing/training split.

   Note: instead of doing this manually, you may also use a built-in cross-validation function.
9. Report the average accuracy, precision, and recall over this process.
10. Consult Daume book ch7 for regularization. You may consider either ridge or lasso regularization, or include both at the same time (it's called *elastic net*).

    Repeat the process for different regularization parameters. Report the results (accuracy, precision, recall) as a function for regularization parameters. This may be in the form of a table or a graph.

    Report the best regularization parameters, and the best results.