ECS607/766 Data Mining 2017/18

Assignment 2: Classification

Introduction

The aim of this lab is to get experience with classification problems, and the concepts of optimization, regularization and feature relevance. This lab will use both matlab and weka. Start by downloading the zip file of lab materials from qmplus. Questions in *red* are assessed toward your final grade.

1. Warm up

The first exercise walks through doing classification in Weka.

1. Load up the credit dataset in Weka. This dataset is a credit scoring dataset, about if a borrower repays their loan or not. Choose Classify in the main explorer window. You can try the two classifiers we learned about so far. (Note that "Max Entropy" and "Logistic Regression" classifiers are the same thing. Despite the confusing name, logistic regression is the linear classifier we learned about last week, and nothing at all to do with regression. We used the less ambiguous of the two names MaxEnt, but weka calls the same thing Logistic Regression).
    - Logistic Regression classifier is under Functions=>Logistic.
    - KNN classifier is under Lazy => IBk
2. Run LogisticRegression/Maximum Entropy classifier. (Make sure Test Options are set to Cross-validation, Folds=10 as is the default. This means we are estimating the test performance through 10 splits of train on 9/10ths, test on 1/10th)
    a. The output window shows you the resulting accuracy ("correctly classified instances") as well which features turned out to be important cues for good or bad credit.
    b. Note the "confusion matrix" in the output window. This says whether the mistakes made were predicting a bad loan to be good, or vice-versa.
3. Run the KNN classifier. The default setting is K=1.
    - Click on the text line right of the choose box for KNN options. Here you can set KNNs K. By setting K=4 you should be able to gain a few percent of accuracy compared to default K=1.
    - There are 1000 instances in this dataset. Try setting K=999 and re-running KNN.
    - What are the new correctly classified instances, and the new confusion matrix? [1 mark]
    - What happened here, and what is the significance?

In the following exercises, we will examine some matlab code for KNN and MaxEnt to get deeper understanding of the concepts and strengths/limitations.

2. MaxEnt/Logistic Regression Classification

1. Invoke matlab. Edit the file lab3_1.m.

- Recall that you can run matlab code one cell at a time by ctrl-enter. You can also set breakpoints by clicking to the right of the line number and step through. You can see a variable in a running program by typing the name, finding it in the workspace browser, or mouse-over the variable name while debugging.

2. Run Cell 1. We will try to classify the 2D dataset with two classes shown.
   - Is it linearly separable?
3. Cell 2 visualizes the predictions of a given MaxEnt classifier
   - The classifier is specified by the three variable array **w**: since it's a 2D classification we need three parameters.
   - The background color shows the class and confidence that would be predicted for a test instance at every point in the space.
   - Compare the code in **util_lrclass_2c** to the logistic regression prediction equation in your notes or textbook.
   - The current setting of the classifier is backward (predicting red for the blue circles, and blue for the red crosses). So the reported accuracy is low.
   - Run cell 2, trying a few different weights. The first weight dimension moves the boundary around, the second two rotate it. Negative weights flip the direction. The scale of the second two weights adjust the sharpness (try dividing, multiplying by 10).
     Can you find a weight that makes a reasonable classifer?
4. Cell 3 has some for loops to search for good weights. This can be used to implement an exhaustive search algorithm to find a good classifier.
   - The line lw0 = -1:0.05:1 means try -1 to 1 in increments of 0.05.
   - Routine **`util_lrlik_reg_2c`** reports the negative log likelihood of the training data (~inverse probability of everything correctly classified). From the class notes, this is the evaluation criterion for LR/MaxEnt. So a good classifier minimizes this.
   - Try to set ranges for the next two parameters that enable the loop to find a reasonable classifier. The visualization routine will show the classifiers being searched. But you may want to use VIZ_INT to adjust how frequently the display is updated to speed up the search.
   - How many times did your loop execute? Report your classifier weights that get you > 75% train accuracy* [1 mark]
5. Cell 4. The previous algorithm is inefficient. Let's look for a better way to optimize the classifier.
   - Cell4a: The first return value of **`util_lrlik_reg_2`** `i`s the current likelihood (~probability of classifying everything correctly). The second return value "dllh" is the **gradient of the likelihood**. The current accuracy is bad (45%), so we should move the weights in this direction to improve the probability of classification.
   - Cell 4b: Update the weights to move in the direction of the gradient (Hint: try w = w-0.1*dllh)
     i. What is the new likelihood (llh)? Did it improve? (Recall that we are measuring negative log likelihood, so improve means gets lower)
     ii. *What are your new weights and classification accuracy? [1 mark]
     iii. How would you use this concept to optimize your classifier?

iv. **Bonus question:** If you know matlab, try to write a simple gradient optimizer that uses a loop to follow the gradient downhill until a good classifier is obtained. What problems do you encounter?

6. Cell 5: Let's automate the idea of following the gradient to improve the classifier.
    - **fminunc** is a matlab routine which takes a pointer to a function (such as `util_lrlik_reg_2)` which takes data and a model parameter, and returns an objective and objective gradient. It will follow the gradient downhill until it gets to the bottom and stops changing.
    - The first return value of fminunc is the optimized weights. You will see the classification using these weights now works.
    - The fourth return value of fminunc is a structure with some metadata. out.funcCount reports how many times fminunc had to to evaluate the classifier objective.
    - *How many times was it evaluated? How does this relate to the # of loop executions in Q2.4? [1 mark]

3. MaxEnt versus KNN

1. Edit lab3_2.m. Run first cell to load some data.
2. Run the second cell.
    - MaxEnt can't classify this data. Why?
3. Third cell: Shows how to classify with KNN.
    a. Note that the parameter K is available for you to set.
    b. Run the cell. If you couldn't see it before, now you can see that the KNN classifier's prediction (background color) should show that the two classes are arranged in a spiral.
    c. * What K gets you best train accuracy? What K gets you best test accuracy? [2 marks]
    d. Why?
    e. Observe the change in decision boundary complexity with K. E.g.,At K=40 the boundary is a simple wave rather than the spiral at K=4.
    f. The fifth parameter off knnclassify is the type of distance to use. Conventional Euclidean distance is the default. Try some others (type **doc knnclassify** in the command window), for example 'cityblock'. How does the decision surface change?
4. Bonus question: Go to the fourth cell.
    a. This is back to the easier data as shown in Fig 2.
    b. Now we investigate what happens if there are some irrelevant columns (features/attributes) in our database. (Added by third and fourth lines in the cell). You can't visualize all ten dimensions at once, but a Fig 3 shows the original two dimensions (x+y axis)with one of eight new irrelevant columns on Z-axis. Try to use the rotation function to rotate the view so you can see the two circles.
    c. All the other perspectives where the data looks confusing are a challenge the classifier has to deal with when there are irrelevant columns!

d. You have seen how to use knnclassify and MaxEnt to train and test. Use these two methods to classify the new data. (Hint: The data is now as 10 attributes, so the weight vector for MaxEnt should have 11 dimensions).
e. What are the train/test performances for this new data? Which classifier is better?
f. Why?