BIMM 185 Lab Report Week 7

Lihui Lu
5/23/2017

Introduction

Last week we kept on working with the operon mini-project. After building the inference model, thresholds should be set in order to make predictions on unknown data. To test the model threshold, both sensitivity and specificity should be taken into account.

Sensitivity

The sensitivity is the measurement of true positive rate. It measures the fraction of the true positives that can be caught by the program from all the true positives. A model with high sensitivity is able to catch as much as true positives as possible.

$$Sensitivity = \frac{TP}{TP + FN}$$

Specificity

The sensitivity is the measurement of true negative rate. It measures the fraction of the true negatives that can be identified by the program to be negatives. A model with high specificity is about to limit the amount of false positives in the positive set.

$$Specificity = \frac{TN}{TN + FP}$$

Accuracy

The accuracy counts the number of predictions that are correct overall. Thus, it takes into account the fractions of the true positives and true negatives.

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

Picking a threshold

First check the data produced by the model on the E.coli genome. For all adjacent gene pairs in the same directon, the probability of the pair of gene appears the the same operon is calculated. True positives are genes recorded in the 'OperonSet.txt' and the True negatives are the gene pairs that appear at the border of the operons. Move the threshold from the min value of prob for the TP set to the max vale of prob for the TP set with increments of 0.01.

mysql> select status, min(prob) from tus group by status;
+--------------+------------------+
| status | min(prob) |
+--------------+------------------+
TP	0.147796029736
TN	1.83067568463e-77
UNDETERMINED	0
+--------------+------------------+

mysql> select status, avg(prob) from tus group by status;
+--------------+-------------------+
| status | avg(prob) |
+--------------+-------------------+
TP	0.7684180680170897
TN	0.3718289765606816
UNDETERMINED	0.6420713046557022
+--------------+-------------------+

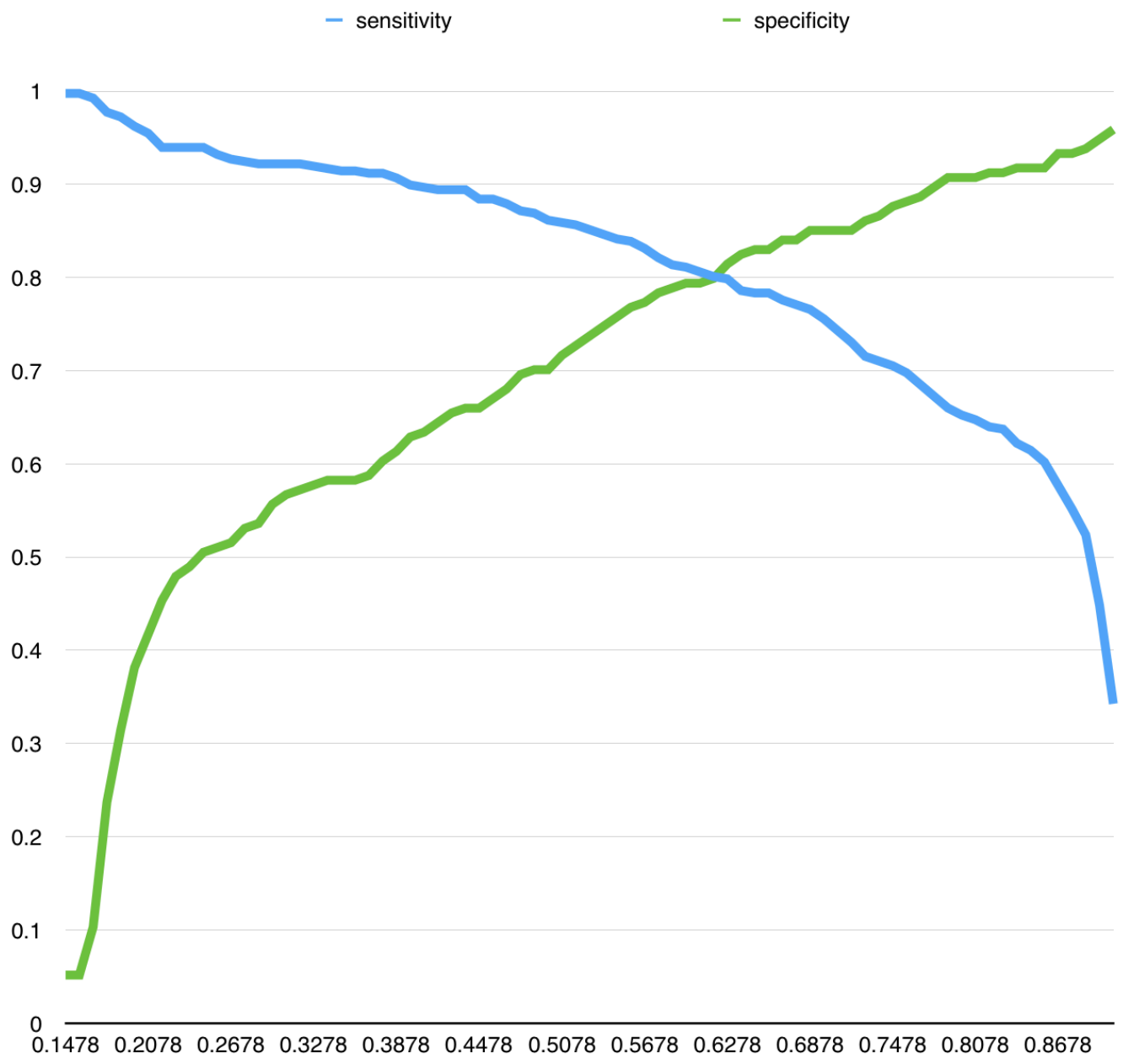
mysql> select max(prob), status from tus group by status;
+---------------+--------------+
| max(prob) | status |
+---------------+--------------+
0.914181591669	TP
0.914204014239	TN
0.955004163996	UNDETERMINED
+---------------+--------------+

Plot the sensitivity and specificity against different threshold, it can be found that sensitivity decreases as the threshold is raised and the specificity increases.

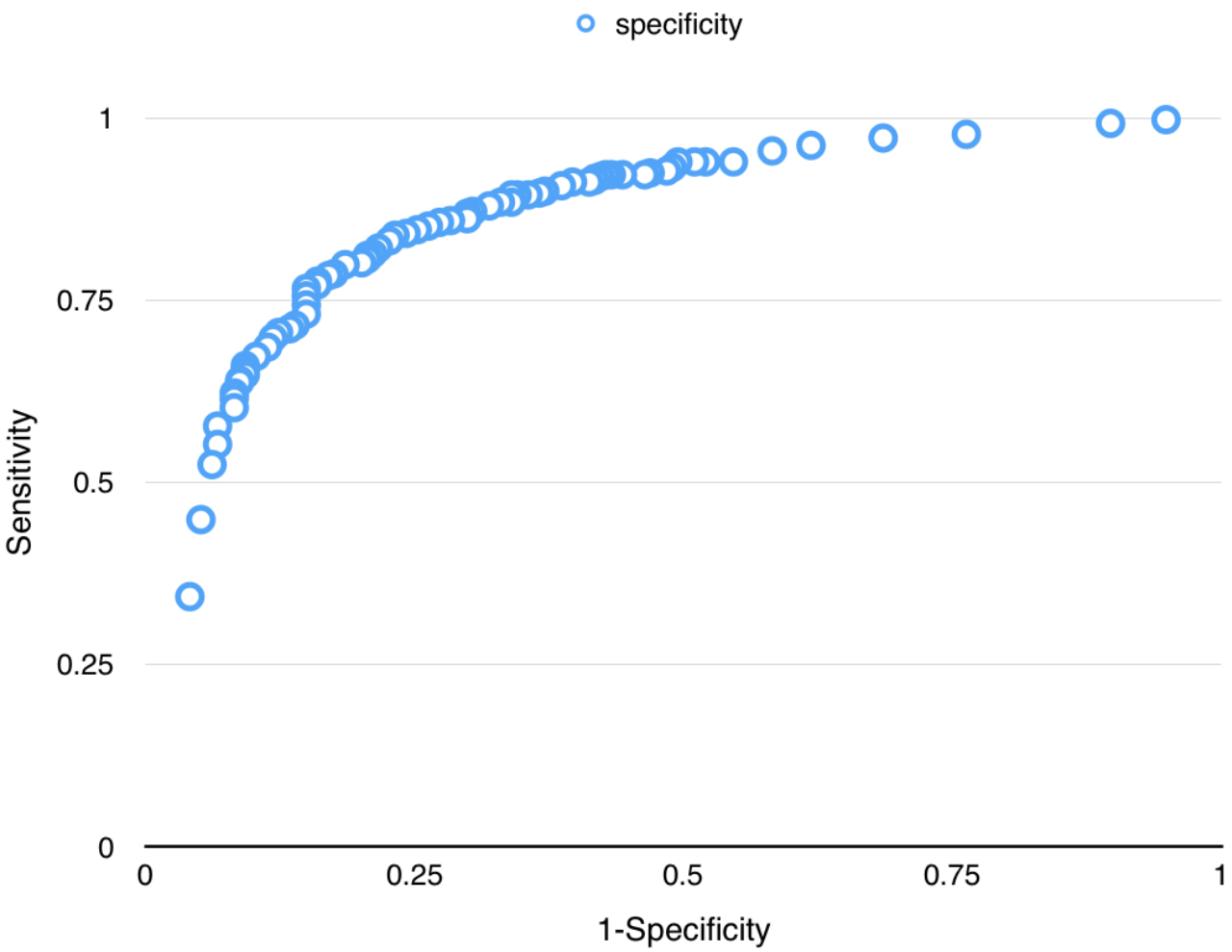
Python source code for threshold walkthrough

```
import sys
import pymysql
import matplotlib.pyplot as plt
import numpy as np

'''
get average probability predicted by the model on the TP set
'''
def query_avg(conn):
    cur = conn.cursor()
    sql_statement = ("select avg(prob) from tus where status = 'TP';")
    cur.execute(sql_statement)
    result = cur.fetchone()
    return result

'''
get minimum probability predicted by the model on the TP set
'''
def query_min(conn):
    cur = conn.cursor()
    sql_statement = ("select min(prob) from tus where status = 'TP';")
    cur.execute(sql_statement)
    result = cur.fetchone()
    return result

'''
get maximum probability predicted by the model on the TP set
'''
def query_max(conn):
    cur = conn.cursor()
    sql_statement = ("select max(prob) from tus where status = 'TP';")
    cur.execute(sql_statement)
    result = cur.fetchone()
    return result


def query_count(conn):
    cur = conn.cursor()
    sql_statement = ("select count(*), status from tus group by status;")
```

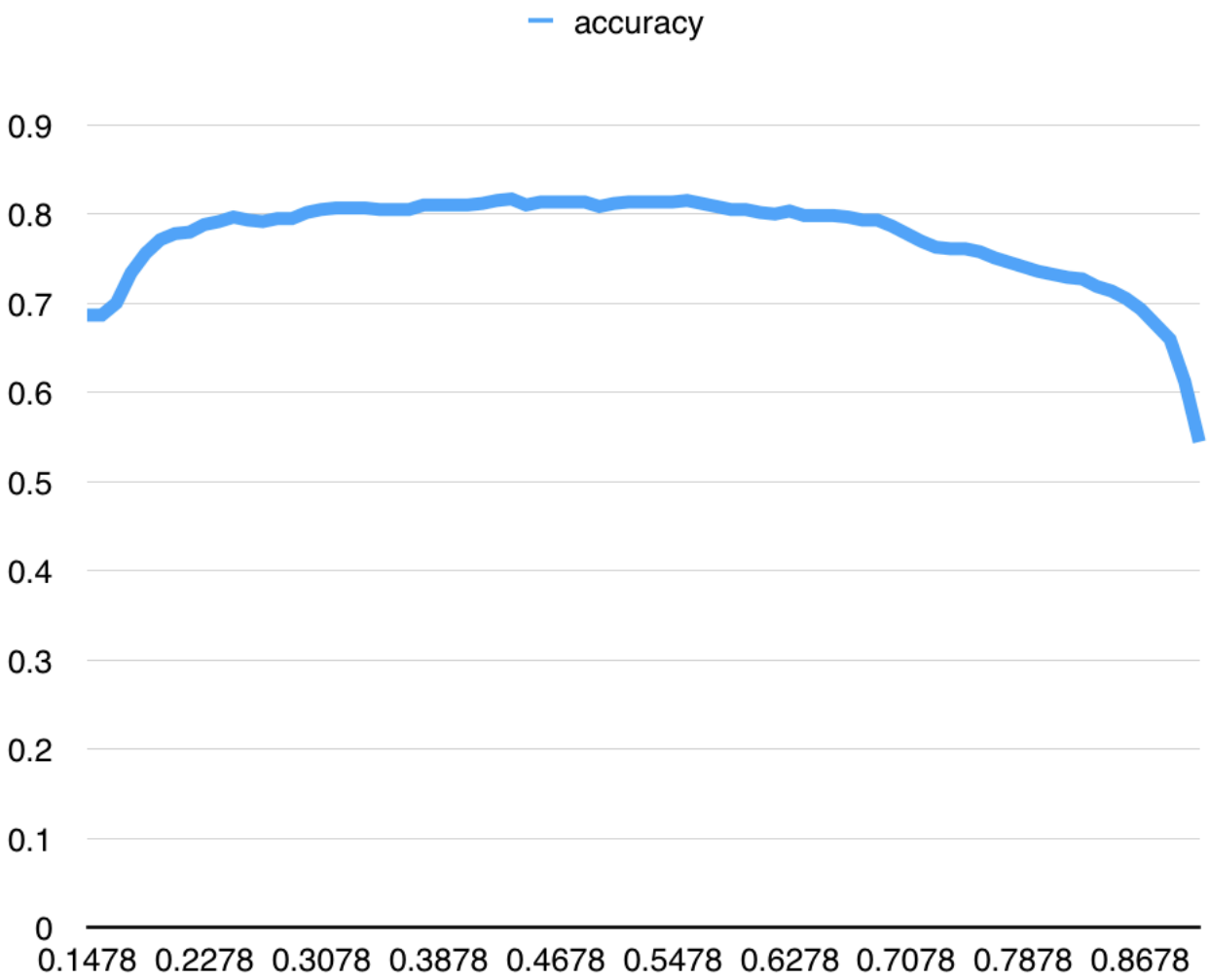
```
    cur.execute(sql_statement)
    result = cur.fetchall()

    return result


def query_prob(conn, threshold):
    cur = conn.cursor()
    sql_statement = ("select count(*), status from tus where prob > {threshold}
 group by status;".format(threshold=threshold))
    cur.execute(sql_statement)
    result = cur.fetchall()
    #print(gene)
    #print(result)
    return result


def main():
    #connect the DB
    myConnection = pymysql.connect(host=hostname, user=username, passwd=passwor
d, db=database, local_infile=True, autocommit=True)

    #get boundary values
    prob_avg = query_avg(myConnection)[0]
    prob_min = query_min(myConnection)[0]
    prob_max = query_max(myConnection)[0]

    #get TP and TN set count
    count_result = query_count(myConnection)
    tp = count_result[0][0]
    tn = count_result[1][0]

    #walk through the thresholds with increment of 0.01
    for i in np.arange(prob_min, prob_max, 0.01):
        #print(i)
        result = query_prob(myConnection, i)
        tp_count = result[0][0]
        fp_count = result[1][0]
        sensitivity = tp_count / tp
        specificity = (tn - fp_count) / tn
        accuracy = (tp_count + tn - fp_count)/ (tp + tn)
        print(i, sensitivity, specificity, 1-specificity, accuracy, sep='\t')
```



The ROC curve is then plotted using sensitivity on the y-axis, and (1-specificity) on the x-axis.



Finally, the accuracy is plotted against the threshold.



From all three plots above, it can be seen that a threshold of 0.6 would gives a sensitivity as well as specificity of 0.8, and an accuracy of 0.8 as well. In the case where both sensitivity and specificity need to be kept high, a threshold of 0.6 could be a good threshold to choose.

Discussion

It is important to pick the suitable threshold to suit one's own needs. It depends on the goal of the prediction model, would it be more important to catch all the true positives, with allowing some false positives or would it be important to predict everything correctly with the risk of losing some false negatives. As the sensitivity and the specificity is in negative relation to each other, one should consider the trade off between the two when picking a good threshold.