BMI260 2019 Problem Set 3

May 30, 2020

1 BIOMEDIN 260/RAD260: Problem Set 3 - Mammogram Project

1.1 Spring 2020

1.2 Name 1:

Ruben Krueger

1.3 Name 2:

Marc Huo

1.4 Introduction

Breast cancer has the highest incidence and second highest mortality rate for women in the US.

Your task is to utilize machine learning to study mammograms in any way you want (e.g. classification, segmentation) as long as you justify why it is useful to do whatever it is you want to do. Turning in a deep dream assignment using mammograms might be amusing, for example, but not so useful to patients. That being said, choose something that interests you. As the adage goes, "do what you love, and you'll never have to work another day in your life, at least in BMI 260."

Treat this as a mini-project. We highly encourage working with 1 other person, possibly someone in your main project team.

In addition to the mammograms themselves, the dataset includes "ground-truth" segmentations and `mass_case_description_train_set.csv`, which contains metadata information about mass shapes, mass margins, assessment numbers, pathology diagnoses, and subtlety in the data. Take some time to research what all of these different fields mean and how you might utilize them in your work. You dont need to use all of what is provided to you.

Some ideas:

1. Use the ROI's or segmentations to extract features, and then train a classifier based on those features using the algorithms presented to you in the machine learning lectures (doesn't need to use deep learning).
2. Use convolutional neural networks. Feel free to use any of the code we went over in class or use your own (custom code, sklearn, keras, Tensorflow etc.). If you dont want to place helper functions and classes into this notebook, place them in a `.py` file in the same folder called `helperfunctions.py` and import them into this notebook.

1.5 Data

The data is here:

https://wiki.cancerimagingarchive.net/display/Public/CBIS-DDSM

1.6 Grading and Submission

This assignment has 3 components: code, figures (outputs/analyses of your code), and a write-up detailing your mini-project. You will be graded on these categories.

If you're OK with Python or R, please place all three parts into this notebook/.Rmd file that we have provided where indicated. We have written template sections for you to follow for simplicity/completeness. When you're done, save as a `.pdf` (please knit to `.pdf` if you are using `.Rmd`, or knit to `.html` and use a browser's "Print" function to convert to `.pdf`).

If you don't like Python OR R, we will allow you to use a different language, but please turn your assignment in with: 1) a folder with all your code, 2) a folder with all your figures, and 3) a `.tex`/`.doc`/`.pdf` file with a write-up.

1.7 PROJECT TITLE HERE

1. Describe what you are doing and why it matters to patients using at least one citation.

We are training a model to predict if a lesion is benign, malignant, or benign without callback based on the features in `features_matrix.csv` (e.g., max intensity, energy). These features are from the ground-truth segmentations. This computer-aided detection/diagnosis can reduce interpretation error by a clinician.

2. Describe the relevant statistics of the data. How were the images taken? How were they labeled? What is the class balance and majority classifier accuracy? How will you divide the data into testing, training and validation sets?

[12]:
```python
import matplotlib.pyplot as plt
from skimage import io, color
from PIL import Image
import cv2
import numpy as np
import pandas as pd
#from pgmpy.models import BayesianModel
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn import linear_model

data_path = "/Users/rubenkrueger/downloads/cs 235 img/"

case_description = pd.read_csv(data_path +'mass_case_description_train_set.csv')
features_matrix = pd.read_csv(data_path +'features_matrix.csv')
features_matrix.rename(columns={'Unnamed: 0': 'Image'}, inplace=True)
```

```
pathology = [np.nan] * len(features_matrix)

imgs = case_description['patient_id'].map(str) + '_' + case_description['side'].
 ↪map(str) + '_' + case_description['view'].map(str)

for i in range(len(imgs)):
    index = np.where(features_matrix["Image"] == imgs[i])
    if len(index[0]) == 0:
        continue
    index = index[0][0]

    pathology[index] = case_description.iloc[i].pathology


features_matrix['Pathology'] = pathology

features_matrix = features_matrix.dropna()
X = features_matrix.drop(["Image", "Centroid_x", "Centroid_y", "Pathology"],␣
 ↪axis=1)

le = preprocessing.LabelEncoder()
le.fit(features_matrix['Pathology'])

y = le.transform(features_matrix['Pathology'])

# list(le.inverse_transform([2, 2, 1]))



# df.drop(['B', 'C'], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
 ↪random_state=42)
```

There are 1599 data rows in `features_matrix.csv` and 1318 data rows in `mass_case_description_train_set.csv`. The data was divided in testing, training, and validation using scikit-learn's `train_test_split()`. The class labels are malignant: 472, benign: 424, benign without callback: 65.

3. Describe your data pipeline (how is the data scrubbed, normalized, stored, and fed to the model for training?).

The data was not normalized nor scrubbed, as this was not deemed necessary for our models.

4. Explain how the model you chose works alongside the code for it. Add at least one technical citation to give credit where credit is due.

[18]:
```
def train_and_test_decision_tree(X_train, X_test, y_train, y_test):
    clf = tree.DecisionTreeClassifier()
    clf = clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    pred = clf.predict(X_test)
    print("Accuracy:",metrics.accuracy_score(y_test, pred))
    print(confusion_matrix(y_test, pred))
    print(classification_report(y_test, pred))
    #print("Accuracy:", score)

    return metrics.accuracy_score(y_test, pred)



# def train_test_linear_regression(X_train, X_test, y_train, y_test):

#     lr = linear_model.LinearRegression()
#     lr.fit(X_train, y_train)


#     score = lr.score(X_test, y_test)

#     print("Accuracy:", score)

#     return score



from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix
def train_test_nearest_neighbors(X_train, X_test, y_train, y_test):
    #nbrs = NearestNeighbors(n_neighbors=2, algorithm='auto').fit(X)
    knn = KNeighborsClassifier(n_neighbors = 34)
    knn.fit(X_train, y_train)
    pred = knn.predict(X_test)
    print("Accuracy:",metrics.accuracy_score(y_test, pred))
    print(confusion_matrix(y_test, pred))
    print(classification_report(y_test, pred))

    return metrics.accuracy_score(y_test, pred)

    def nearest_neighbors_k():
        #iteration to test for best K value (adapted from Shoaib)
        error_rate = []
        for i in range(1,40):
            knn = KNeighborsClassifier(n_neighbors = i)
            knn.fit(X_train, y_train)
            pred_i = knn.predict(X_test)
            error_rate.append(np.mean(pred_i != y_test))

        plt.figure(figsize=(15,6))
        plt.
↪plot(range(1,40),error_rate,color='blue',linestyle='dashed',marker='o',␣
↪markerfacecolor='red', markersize='10')
        plt.xlabel('no. of K')
        plt.ylabel('Error Rate')


from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
def train_test_naive_bayes(X_train, X_test, y_train, y_test):
    gnb = GaussianNB()
    gnb.fit(X_train, y_train)
    pred = gnb.predict(X_test)
    print("Accuracy:",metrics.accuracy_score(y_test, pred))
    print(confusion_matrix(y_test, pred))
    print(classification_report(y_test, pred))

    return metrics.accuracy_score(y_test, pred)


from sklearn import svm
def train_test_svm(X_train, X_test, y_train, y_test):
    clf = svm.SVC(kernel='linear')
    clf.fit(X_train, y_train)
    pred = clf.predict(X_test)
    print("Accuracy:",metrics.accuracy_score(y_test, pred))
    print(confusion_matrix(y_test, pred))
    print(classification_report(y_test, pred))

    return metrics.accuracy_score(y_test, pred)



from sklearn.ensemble import RandomForestClassifier
def train_test_rfc(X_train, X_test, y_train, y_test):
    classifier = RandomForestClassifier(max_depth=2, random_state=0)
    classifier.fit(X_train, y_train)
    pred = classifier.predict(X_test)
    print("Accuracy:",metrics.accuracy_score(y_test, pred))
    print(confusion_matrix(y_test, pred))
    print(classification_report(y_test, pred))
    return metrics.accuracy_score(y_test, pred)

results = {}
results['DT'] = train_and_test_decision_tree(X_train, X_test, y_train, y_test)

results['KNN'] = train_test_nearest_neighbors(X_train, X_test, y_train, y_test)
# results['LR' ] = train_test_linear_regression(X_train, X_test, y_train,␣
 ↪y_test)
results['NB'] = train_test_naive_bayes(X_train, X_test, y_train, y_test)
results['RFC'] = train_test_rfc(X_train, X_test, y_train, y_test)
```

```
Accuracy: 0.46540880503144655
[[68 11 56]
 [ 9  2 11]
 [71 12 78]]
              precision    recall  f1-score   support

           0       0.46      0.50      0.48       135
           1       0.08      0.09      0.09        22
           2       0.54      0.48      0.51       161

    accuracy                           0.47       318
   macro avg       0.36      0.36      0.36       318
weighted avg       0.47      0.47      0.47       318

Accuracy: 0.5408805031446541
[[78  0 57]
 [13  0  9]
 [67  0 94]]
              precision    recall  f1-score   support

           0       0.49      0.58      0.53       135
           1       0.00      0.00      0.00        22
           2       0.59      0.58      0.59       161

    accuracy                           0.54       318
   macro avg       0.36      0.39      0.37       318
weighted avg       0.51      0.54      0.52       318

Accuracy: 0.4937106918238994
[[102   3  30]
 [ 17   0   5]
 [ 98   8  55]]
              precision    recall  f1-score   support

           0       0.47      0.76      0.58       135
           1       0.00      0.00      0.00        22
           2       0.61      0.34      0.44       161

    accuracy                           0.49       318
   macro avg       0.36      0.37      0.34       318
weighted avg       0.51      0.49      0.47       318

Accuracy: 0.5471698113207547
[[ 72   0  63]
 [ 15   0   7]
 [ 59   0 102]]
              precision    recall  f1-score   support

           0       0.49      0.53      0.51       135
           1       0.00      0.00      0.00        22
           2       0.59      0.63      0.61       161

    accuracy                           0.55       318
   macro avg       0.36      0.39      0.38       318
weighted avg       0.51      0.55      0.53       318


/Users/rubenkrueger/opt/anaconda3/lib/python3.7/site-
packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/rubenkrueger/opt/anaconda3/lib/python3.7/site-
packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

We had many models, but one of the most interesting is KNN. The K-nearest neighbor algorithm is one of the least complex machine learning algorithms. Each data point is classified according the most common label of its K closest neighbors, where distance can be interepreted to be Euclidian distance [3].

5. There are many ways to do training. Take us through how you do it (e.g. "We used early stopping and stopped when validation loss increased twice in a row.").

We used many different models, ranging from linear regression to decision trees to kNNs. For each, we tried to minimize the bias error as much as possible, since these were simple models.

6. Make a figure displaying your results.

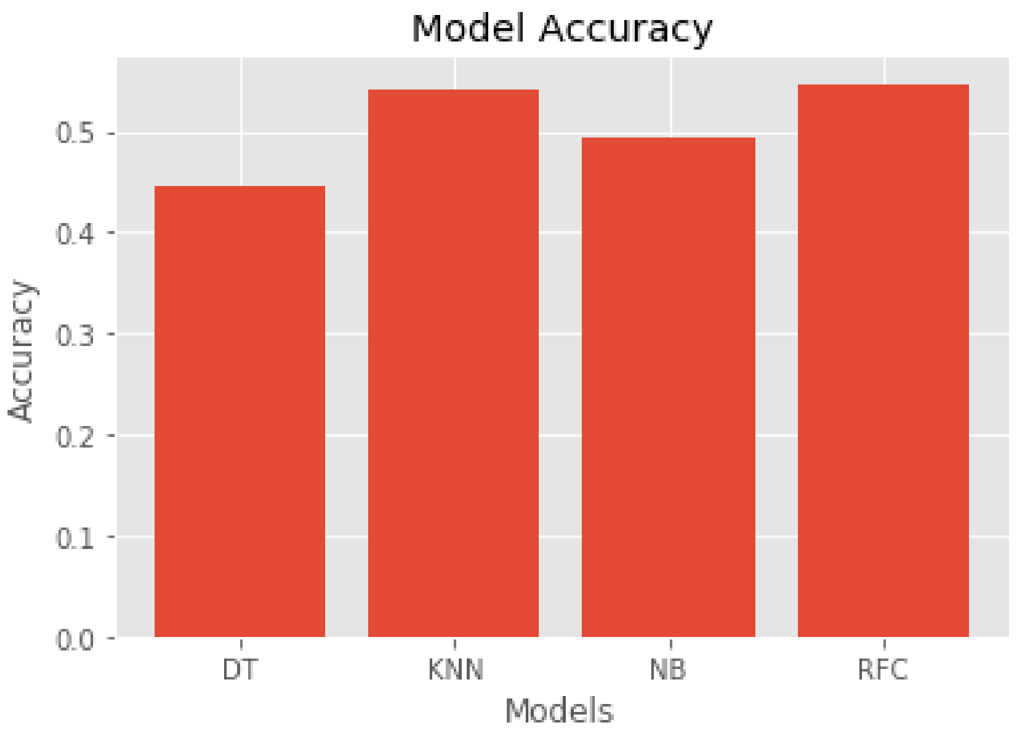
[17]:
```python
import matplotlib.pyplot as plt
plt.style.use('ggplot')



D = results
```

```
plt.bar(range(len(D)), list(D.values()), align='center')
plt.xticks(range(len(D)), list(D.keys()))
plt.title('Model Accuracy')
plt.xlabel('Models')
plt.ylabel('Accuracy')

plt.show()
```



7. Discuss pros and cons of your method and what you might have done differently now that you've tried or would try if you had more time.

Our results suggest that these simple, highly interpetable models may be less effective for this dataset. Thus, if we had more time, we would have applied either deep learning methods or graphical models. Graphical models are have already shown impressive results in medicine; in particular, a Bayesian network shows promise here [2].

You will not be graded on the performance of your model. You'll only be graded on the scientific soundness of your claims, methodology, evaluation (i.e. fair but insightful statistics), and discussion of the strengths and shortcomings of what you tried. Feel free to reuse some of the code you are/will be using for your projects. The write-up doesn't need to be long (~1 page will suffice), but please cite at least one clinical paper and one technical paper (1 each in questions 1 and 4 at least, and more if needed).

2 References

1. Antal, P., G. Fannes, B. De Moor, and Y. Moreau. "Probabilistic Graphical Models for Computational Biomedicine." Methods of Information in Medicine 42, no. 02 (2003): 161–68. https://doi.org/10.1055/s-0038-1634328.
2. Jalalian, Afsaneh, Syamsiah B.t. Mashohor, Hajjah Rozi Mahmud, M. Iqbal B. Saripan, Abdul Rahman B. Ramli, and Babak Karasfi. "Computer-Aided Detection/Diagnosis of Breast Cancer in Mammography and Ultrasound: a Review." Clinical Imaging 37, no. 3 (2013): 420–26. https://doi.org/10.1016/j.clinimag.2012.09.024.
3. Li, Chao, Shuheng Zhang, Huan Zhang, Lifang Pang, Kinman Lam, Chun Hui, and Su Zhang. "Using the K-Nearest Neighbor Algorithm for the Classification of Lymph Node Metastasis in Gastric Cancer." Computational and Mathematical Methods in Medicine 2012 (2012): 1–11. https://doi.org/10.1155/2012/876545.

```
[ ]:
```