Udacity: AI for Healthcare

Contents

1 Applications to 2D Medical Imaging

1.1 Clinical Foundations of 2D Medical Imaging

- **X-ray**: a 2D imaging technique that projects a type of radiation called x-rays down at the body from a single direction to capture a single image.
- **Ultrasound**: a 2D imaging technique that uses high-frequency sound waves to generate images.
- **Computed Tomography (CT)**: a 3D imaging technique that emits x-rays from many different angles around the human body to capture more detail from more different angles.
- **Magnetic Resonance Imaging (MRI)**: a 3D imaging technique that uses strong magnetic fields and radio waves to create images of areas of the body from all different angles.
- **2D imaging**: an imaging technique that pictures are taken from a single angle.
- **3D imaging**: an imaging technique that pictures are taken from different angles to create a volume of images.

X-rays can show us different absorption levels by the following: bone has high absorption (*bright white*), soft tissue has medium absorption (*gray*), and air has low absorption (*dark*). Chest x-rays for detecting **pneumonia** will be filled with infiltrates (which are denser with air) and appear as lighter shading in x-rays of lungs. X-rays are stored as single-channel grayscale images and in DICOM format.

PACS (Picture Archiving and Communication System) are used for storing and accessing images from different areas. **Diagnostic imaging** occurs when Clinician believes something *may* be wrong with a patient and needs an image to verify (i.e. brain tumors). **Screening imaging** occurs when nothing actually wrong with patient, but patient is in risk group for disease and need regular imaging to monitor.

We want to prioritize the most urgent images for radiologists to read based on the probabilities of their being a life threatening finding on these images (between two diseases, we want to read the max first).

```python
worklist = pd.read_csv('probabilities.csv')
worklist['time_to_read'] = np.arange(6, 6*(len(worklist)+1), 6)
worklist['max_prob'] = worklist.iloc[:, 1:3].max(axis=1)
worklist.sort_values(by='max_prob', ascending=False, inplace=True)
worklist['time_to_read_max'] = np.arange(6, 6*(len(worklist)+1), 6)
worklist['time_delta'] = worklist['time_to_read'] - worklist['time_to_read_max']
# Check to see how well algorithm had brain bleeds read 30 mins faster
worklist[(worklist.time_delta > 30) & (worklist.Image_Type == 'head_ct')]
# Check to see how well algorithm had aortic dissections read 15 mins faster
worklist[(worklist.time_delta > 15) & (worklist.Image_Type == 'chest_xray')]
```

- We find the initial time that each image will be read based on a 6 minute read time.
- We then order the images based on the max probability between the image types and sort them.
- We then find the new read times and find the change in times between that and the original.
- We can then check to see how our times improved. 14 head CTs that were read more than 30 minutes faster than their original order. 28 chest x-rays that were read more than 15 minutes faster than their original order.

Sensitivity measures the proportion of accurately-identified *positive* cases, also called *recall*, and is found by $\frac{TP}{TP+FN}$. **High Sensitivity Tests** are good for ruling out diseases (if 100% accuracy then it can find all patients with the disease) and is reliable when the result is negative (rarely misdiagnose people who have disease). **Specificity** measures the proportion of accurately-identified *negative* cases, and is found by $\frac{TN}{TN+FP}$. **Highly Specific Tests** are good for ruling in disease (if 100% accuracy then it can find all patients without the disease) and is reliable when the result is positive (rarely misdiagnose people who don't have disease).

To evaluate a segmentation/localization algorithm, we look at the overlap between our algorithm segment (X) and radiologist segment (Y), and finding the overlap between the two. This can be found by the **Dice Coefficient**, which is $Dice(X,Y) = \frac{2*|X \cap Y|}{|X|+|Y|}$

1.2 2D Medical Imaging Exploratory Data Analysis

DICOM (Digital Imaging and Communications in Medicine) is the standard for the communication and management of medical imaging information and related data. This allows for *interoperability*, which means images from one hospital can be read from any other hospital. A **series** refers to a single 2D image, and a **study** is comprised of all image series for the patient. The file contains:

- Header - contains all attributes about an image except the pixel data. This includes information on the patient (name, ID, etc.), the study (date, time, etc.), and the series (number, type, etc.).
- Image - contains the pixel data.

Radiologist Report is not part of the DICOM, but is stored in the PACS and EMR, which tell us details such as location, area of interest, findings, and impression (summary).

Using the **pydicom** package, we can read these files and extract the image pixel data from within. We may also want to explore the intensity profiles of these images by plotting histograms.

```
bbox = pd.read_csv(’bounding_boxes.csv’) # bounding box locations
dicom1 = pydicom.dcmread(’dicom_00013659_019.dcm’) # read single dicom series
dicom1_pdata = dicom1.pixel_array # get pixel data
plt.imshow(dicom1_pdata, cmap=’gray’) # see image
plt.hist(dicom1_pdata.ravel(), bins = 256); # intensity

new_img = dicom1_pdata.copy() # Normalize Image
new_img = (new_img - np.mean(dicom1_pdata))/np.std(dicom1_pdata)
```

When preparing **DICOM Headers** to be used in training, we want fast access by pre-extracting all data from DICOM headers into a dataframe. The key attributes are:

- Patient ID - be sure not to include same patient in both train and test set.
- Patient’s Sex - be sure to have equal male/female proportions in train and test set.
- Patient’s Age - be sure to have equal age distributions in train and test set.
- Study & Series Instance UID - make sure we don’t overtrain on same study or series.

```
mydicoms = glob.glob("*.dcm") # gather all dicom files names
all_data = []

for dicom in mydicoms: # read dicom and save relevant information
  dicom_tmp = pydicom.dcmread(dicom)
  fields = [dicom_tmp.PatientID, dicom_tmp.PatientAge, dicom_tmp.PatientSex,
            dicom_tmp.Modality, dicom_tmp.StudyDescription, dicom_tmp.Rows,
            dicom_tmp.Columns]
  all_data.append(fields)

mydata = pd.DataFrame(all_data, # create new dataframe with header info
                      columns = [’PatientID’,’PatientAge’,’PatientSex’,
                                 ’Modality’,’Findings’,’Rows’,’Columns’])
```

- **Image artifact**: An object or distortion in an image that reduces its quality
- **Foreign body**: An object in a medical image that is not biological material from the patient, such as a pacemaker or wire.
- **Intensity profile**: the distribution of all pixels’ intensity values that comprise an image

1.3 Classification Models of 2D Medical Images

Otsu's Method is often used for background extraction and classification. It takes the intensity distribution of an image and searches it to find the intensity threshold that minimizes the variance in each of the two classes. Once it discovers that threshold, it considers every pixel on one side of that image to be one class and on the other side to be another class. We can then take the modes (peaks) for both classes, and identify the new image class by seeing which mode it is closest to.

Below, we implement this algorithm and use it to extract the image intensity distributions for two types of breasts: fatty and dense. These are standard classifications of breast tissue that radiologists make for all mammography studies. We then use image intensity to classify images into one of the two classes. Using a threshold, we are able to look at pixel intensity values only for the tissue (exclude background).

```python
# Use a threshold to separate the image background to create new binarized images
thresh = 50
fatty_imgs = glob.glob("fatty/*")
dense_imgs = glob.glob("dense/*")
fatty_intensities = []
dense_intensities = []

for i in fatty_imgs:
  img = plt.imread(i)
  img_mask = (img > thresh)
  fatty_intensities.extend(img[img_mask].tolist())

for i in dense_imgs:
  img = plt.imread(i)
  img_mask = (img > thresh)
  dense_intensities.extend(img[img_mask].tolist())

dense_mode = scipy.stats.mode(dense_intensities)[0][0] # 176
fatty_mode = scipy.stats.mode(fatty_intensities)[0][0] # 140

for i in dense_imgs:
  img = plt.imread(i)
  img_mask = (img > thresh)
  img_mode = scipy.stats.mode(img[img_mask])[0][0]
  fatty_delta = img_mode - fatty_mode
  dense_delta = img_mode - dense_mode
  print('Fatty') if (np.abs(fatty_delta) < np.abs(dense_delta)) else print("Dense")
```

We often want out data for training and testing to follow the same distributions, as well as minimize the class imbalances found within training data. We can do this be random under sampling to balance out the classes. In the following example, we will want even class balance for training set, but we want only 20% of the testing data to be positive and 80% to be negative.

```python
d = pd.read_csv('findings_data_5000.csv')
train_df, valid_df = train_test_split(d, test_size=0.2, stratify = d['Pneumothorax'])
train_df['Pneumothorax'].sum()/len(train_df) # 0.044 (unequal class balance)
valid_df['Pneumothorax'].sum()/len(valid_df) # 0.044 (unequal class balance)

p_ind = train_df[train_df.Pneumothorax==1].index.tolist()
np_ind = train_df[train_df.Pneumothorax==0].index.tolist()
np_sample = sample(np_ind, len(p_ind))
train_df = train_df.loc[p_ind + np_sample]
train_df['Pneumothorax'].sum()/len(train_df) # 0.5 (equal class balance)

p_ind = valid_df[valid_df.Pneumothorax==1].index.tolist()
np_ind = valid_df[valid_df.Pneumothorax==0].index.tolist()
np_sample = sample(np_ind, 4*len(p_ind))
valid_df = valid_df.loc[p_ind + np_sample]
valid_df['Pneumothorax'].sum()/len(valid_df) # 0.2 (20/80 ratio)
```

The **gold standard** is the method that detects the disease with the highest sensitivity and accuracy. Any new method can then be compared to this. For example, the gold standard for pneumonia detection is a biopsy (but takes a long time) so they rely on the radiologists suggestion to start treatment and then verify with the biopsy later. Often times, the gold standard is unattainable for an algorithm developer. So, you still need to establish the **ground truth** to compare your algorithm. Typically sources for these ground truth are:

- Biopsy-based labeling. Limitations: difficult and expensive to obtain (outside of PACS).
- NLP extraction from radiology reports. Limitations: may not be accurate.
- Expert (radiologist) labeling. Limitations: expensive and a lot of time to come up with protocols.
- Labeling by another state-of-the-art algorithm. Limitations: may not be accurate.

The **silver standard** involves hiring several radiologists to each make their own diagnosis of an image. The final diagnosis is then determined by a voting system across all of the radiologists' labels for each image. Note, sometimes radiologists' experience levels are taken into account and votes are weighted by years of experience.

```python
labels = pd.read_csv('labels.csv')
labels = labels.replace('benign',1).replace('cancer',0)

gt1 = labels['biopsy'] # biopsy labels
gt2 = labels.iloc[:, 0:3].mode(axis=1) # voting system (equal weight)

weighted_labels = 0.33*labels['rad1'] + 0.67*labels['rad2'] + labels['rad3']
gt3 = [1 if x > 1 else 0 for x in weighted_labels] # voting system (weighted)

biopsy_to_votes = (gt1 == gt2)
len(biopsy_to_votes[biopsy_to_votes==False]) # 9 mislabled

biopsy_to_votes = (gt1 == gt3)
len(biopsy_to_votes[biopsy_to_votes==False]) # 15 mislabled
```

1.4 Translating Algorithm into a Clinical Setting with the FDA

There are 3 **Risk Categories** for the FDA for new medical devices:

- **Class I** - not intended for use in supporting/sustaining life or preventing impairment to human health (also can't present potential risk of illness/injury). This is the most common (47% of devices), pose the lowest risk, and require the fewest regulatory requirements (exempt or 510(k)).
- **Class II** - devices which require substantial equivalence to a predicate to prove safety/effectiveness of the device. Most common route that AI programs go through, are considered medium risk, and have more regulatory requirements (exempt or 510(k)).
- **Class III** - devices that sustain/support life, are implanted, or present potential unreasonable risk of illness/injury. Only 10% of devices fall into this category, are considered high risk, and have the highest regulatory requirements (510(k) or PMA).

The **Intended Use** ties directly to your risk category. For example, a 2D medical imagining algorithm to detect breast cancer from a mammogram would be Class III, since a missed diagnosis could be fatal. If predicate device exists we follow the 510(k), but since there is none, we would have to follow the PMA route. If instead our algorithm was used to *assist the radiologist* in the detection of breast abnormalities on mammograms, this is considered a *Computer-Assisted Diagnosis (CADx)* and the FDA assumes the radiologists will depend heavily on this, it will still be Class III.

The **Indications for Use** describe conditions for use of the device and make more specific suggestions about how your algorithm could be used. Indications for use statement describes precise situations and reasons where and why you would use this device. For our mammogram algorithm this could be describing it is used for screening mammography studies, the age range, and with no prior history of breast cancer.

When the FDA talks about **algorithm limitations**, they want to know more about scenarios where your algorithm is not safe and effective to use. In other words, they want to know where our algorithm will fail. This could be things such as poor performance for patients with prior disease history.

After your algorithm is cleared by the FDA and released, the FDA has a system called **Medical Device Reporting** to continuously monitor. Any time one of your end-users discovers a malfunction in your software, they report this back to you, the manufacturer, and you are required to report it back to the FDA. Depending on the severity of the malfunction, and whether or not it is life-threatening, the FDA will either completely recall your device or require you to update its labeling and explicitly state new limitations that have been encountered.

If your algorithm needs to work in an emergency workflow, you need to consider **computational limitations** and inform the FDA that the algorithm does not achieve fast performance in the absence of certain types of computational infrastructure. This would let your end consumers know if the device is right for them.

```python
def sens_spec(tp, fp, tn, fn):
  sens = tp/(tp+fn)
  spec = tn/(tn+fp)
  return sens, spec

tn, fp, fn, tp = confusion_matrix(data.iloc[:, 0], data.iloc[:, -1],
                                  labels=[0,1]).ravel()
sens_spec(tp, fp, tn, fn) # (0.8166666666666667, 0.8235294117647058)

for i in data.columns[1:-1].tolist():
  tn, fp, fn, tp = confusion_matrix(data[data[i]==1].Pneumonia.values,
                                    data[data[i]==1].algorithm_output.values,
                                    labels=[0,1]).ravel()
  sens_spec(tp, fp, tn, fn)
  print(f'{i}\nSensitivity: {sens}\nSpecificity: {spec}\n')
```

- The results indicate that the presence of infiltrations in a chest x-ray is a limitation of this algorithm, and that it performs very poorly on the accurate detection of pneumonia in the presence of infiltration. The presence of nodules and pneumothorax have a slight impact on the algorithm's sensitivity and may reduce the ability to detect pneumonia, while the presence of effusion has a slight impact on specificity and may increase the number of false positive pneumonia classifications.

Precision is optimized when we care most for confidence in positive results, but does not take FN into account (so they are beneficial when you want to *confirm* a diagnosis). **Recall** is optimized for most confident when the test is negative, but does not take FP into account (beneficial for making sure someone *does not* have a disease).

The **Threshold** is the cut-off probability value from which we determine what class a predicted value belongs to, and we change this value depending on which performance metrics we want to maximize. We can asses the trade-off across all threshold values by using a **Precision-Recall Curve**. We can also use the **F1 Score** for binary classification to find the mean of precision and recall (balances the two metrics and is used as a final evaluation of model).

You'll need to perform a standalone clinical assessment of your tool that uses an **FDA validation set** from a real-world *clinical setting* to prove to the FDA that your algorithm works. You will run this FDA validation set through your algorithm just ONCE. You need to identify a clinical partner to gather the FDA validation set. First, you need to describe who you want the data from. Second, you need to specify what types of images you're looking for. You need to gather the **ground truth** that can be used to compare the model output tested on the FDA validation set. The choice of your ground truth method ties back to your intended use statement (biopsy for diagnosis, radiologist read for CADx tool). For your validation plan, you need evidence to support your reasoning. As a result, you need a **performance standard** (pull from literature reports). For example, if the literature have F1 scores you could say your model needs an F1 equivalent to the mean of all those scores. You may also need to include assessment time for the algorithm (speed for emergency use).

2 Applications to 3D Medical Imaging

2.1 Clinical Fundamentals