Helping the Blind See with Machine Learning

PHYS 453
Keaton, Christine, Roy
MCC Group
05/05/2020

Outline:

- Introduction to Classifiers
- Introduction to Problem and Data set
  - Reduction of scope
- Data Processing
- Classifier
- Results
- Accuracy
- How work was distributed
- What just didn't work
- What got left out
- Future Modifications
- Conclusions

Intro to Classifiers

What is a classifier?

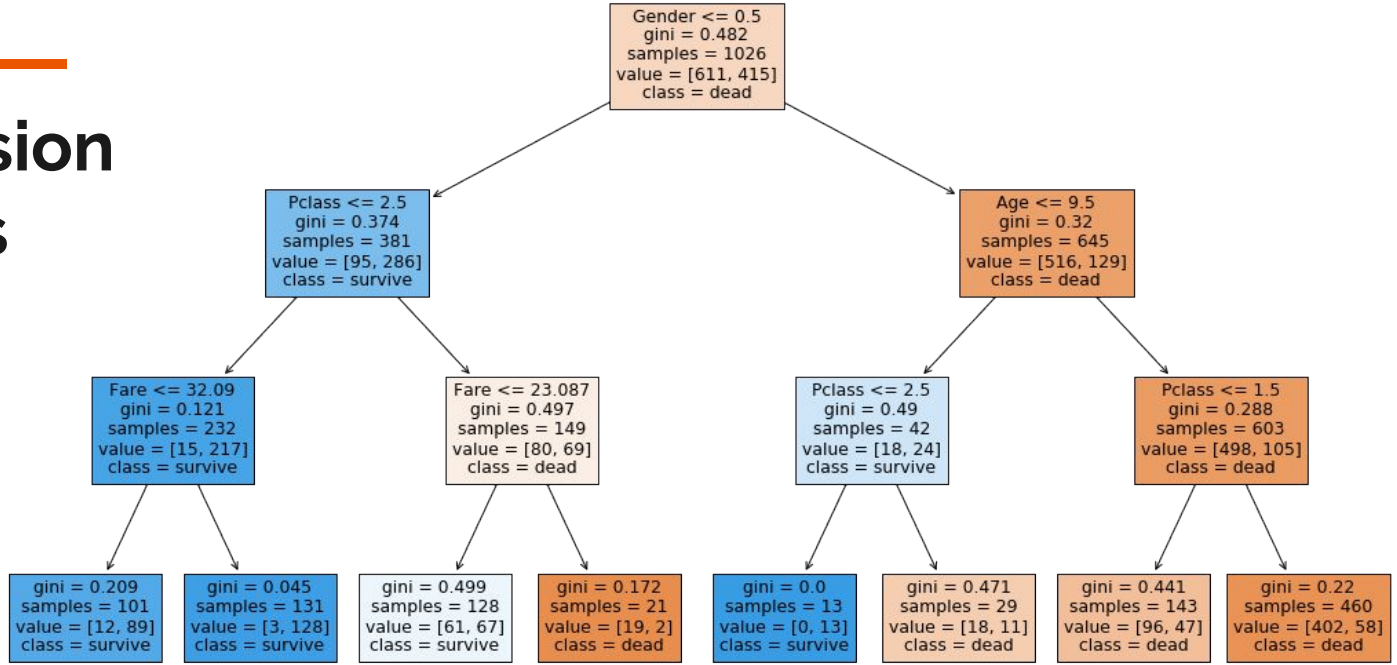
A classifier is a term used in pattern recognition, a subset of machine learning. The classifier takes in training data, trains on the data, and then is able to apply to predict how new data should be classified.

Different types of classifiers that MCC used:

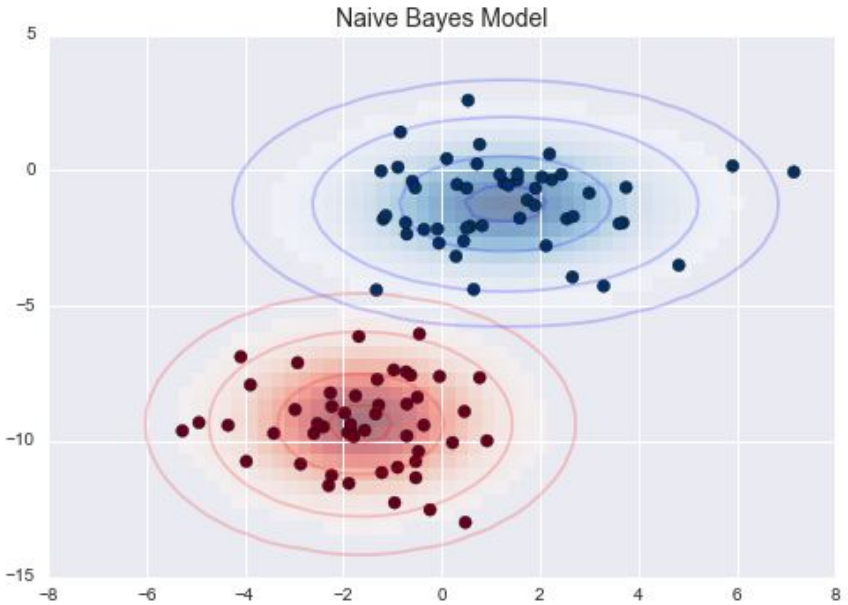
- Nearest neighbors
- Decision trees
- Gaussian Bayes
- Neural network

Nearest Neighbors Classifier

Decision Trees

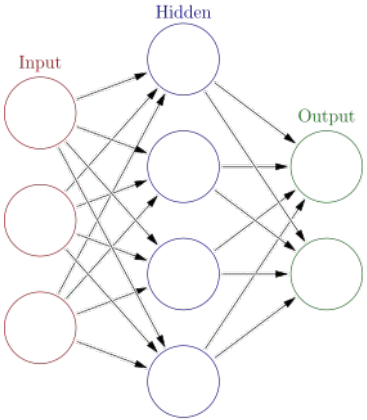


Naive Bayes (GaussianNB)

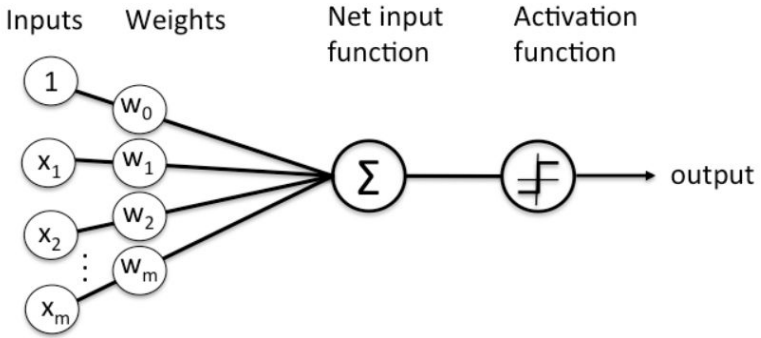


Neural Network

Neural networks depend on weights, activation functions, learning rates, and learning method.



Neural Network Example



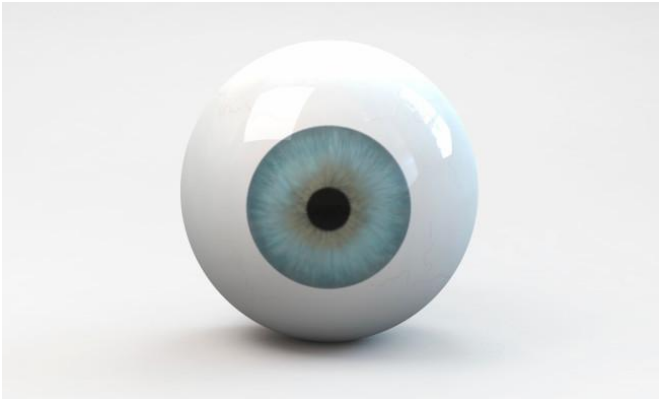
Single node in activation function cartoon

Problem Statement

Research Question: Can we use machine learning to optimize image comprehension for the visually impaired?

Prevalence of Blindness

- 285 million people currently with visual impairments [1]
- 39 million blind [1]



Adjustments of Scope

- Initially, we wanted the user to be able to ask any question and our classifier would answer
- Now, we are just able to classify the color that people labeled the image due to current circumstance
- This is different from the raw color data that the computer can output automatically, we produce results that humans would call the image

Visual question: *Is this inhaler **blue** or yellow?*

Answers:

1. yellow
2. yellow
3. yellow
4. yellow
5. yellow
6. yellow
7. yellow
8. yellow
9. yellow
10. yellow

Source: Search “blue” on VizWiz

VizWiz Challenge

Using Visual Question Answering from VizWiz, we are provided training, validation, and testing json files.

Input:

- User uploads picture

Output:

- Code uses neural networks to predict color


```
"answerable": 0,
"image": "VizWiz_val_00028000.jpg",
"question": "What is this?"
"answer_type": "unanswerable",
"answers": [
    {"answer": "unanswerable", "answer_confidence": "yes"},
    {"answer": "chair", "answer_confidence": "yes"},
    {"answer": "unanswerable", "answer_confidence": "yes"},
    {"answer": "unanswerable", "answer_confidence": "no"},
    {"answer": "unanswerable", "answer_confidence": "yes"},
    {"answer": "text", "answer_confidence": "maybe"},
    {"answer": "unanswerable", "answer_confidence": "yes"},
    {"answer": "bottle", "answer_confidence": "yes"},
    {"answer": "unanswerable", "answer_confidence": "yes"},
    {"answer": "unanswerable", "answer_confidence": "yes"}
]
```

RPI Purpose

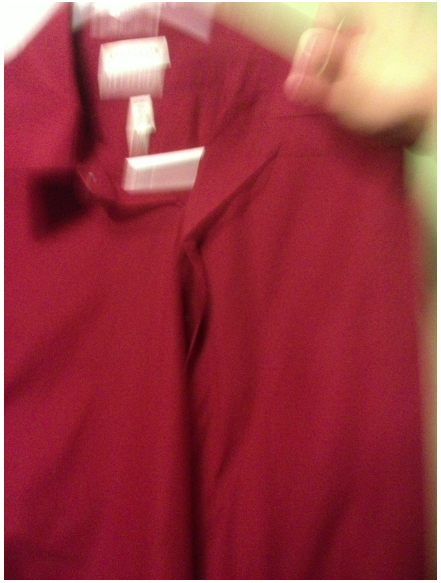
- Meant to be a way to have all code and data on one machine rather than having it on each local machine
- Easier to access data and code


```
pi@raspberrypi: ~
roy@DESKTOP-6R98KNV:~$ ssh pi@150.252.115.10
pi@150.252.115.10's password:
Linux raspberrypi 4.19.97-v7+ #1294 SMP Thu Jan 30 13:15:58 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon May  4 14:29:25 2020 from 150.252.77.36
**************************************************
        W E L C O M E  T O  T H E  P I
               Current home
                    of
               The MCC Group
**************************************************
pi@raspberrypi:~ $
```

Data Processing:
Pulling out color

The features MCC will use for its classifier are the top five RGB values and the percentage of the RGB in picture

```
VizWiz_train_00003664.jpg
['[206.44666667 152.68666667 132.05333333]', '6.69']
['[246.17829457 232.46511628 196.04651163]', '6.88']
['[170.03181818  66.07272727  74.33636364]', '10.52']
['[115.01587302  16.92063492  32.18871252]', '26.65']
['[151.78051392  33.15631692  50.22912206]', '49.26']
```



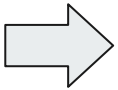
Data Processing:
Parsing through Data: Images and .json

- Required parsing through the .json file first
  - Came formatted as an array of maps (key:value)
- Looked through key values for “question”
  - Sorted through the value (string) for the word “color” or “color?”
- Once found, saved it and checked answers for specific color
  - Saved the image name based on highest frequency of color mentioned in “answer” key
- Iterated through all of data to have corresponding “red” , “blue”, ... images
  - Created separate “color” .txt files
- Looked through the image directories for matching image names
  - If matching, data for the five most dominant colors were extracted
  - Used MiniBatchKMeans for this due to large amount of data
- Once all names in a “color” .txt file were found, it was written out to file “color_info” .txt file
  - This file contained image name, five dominant RGB values and percentages for that RGB value

Data Processing:
Fitting data into array

- Process of execution: Required both shell (for user interface) and .py for actual "work"
  - User was asked to enter "color_info" .txt file
  - Parsed through line by line to get required information
  - Information was then formatted to have [R, G, B] % -> [R*%, G*%, B*%]
    - This serves as a "weight" for the RGB values
  - Data was saved as a (Nx15) multi-dimensional array
    - N: Number of images
    - 15 for the 3 features for our five dominant colors
    - Data was saved as a "pickle" object
      - User could use or not use this object, meant to be a convenience
- Here is the Reader Function

```
VizWiz_train_00003664.jpg
['[206.44666667 152.68666667
132.05333333]', '6.69'] ['[246.17829457
232.46511628 196.04651163]', '6.88'] ….
```



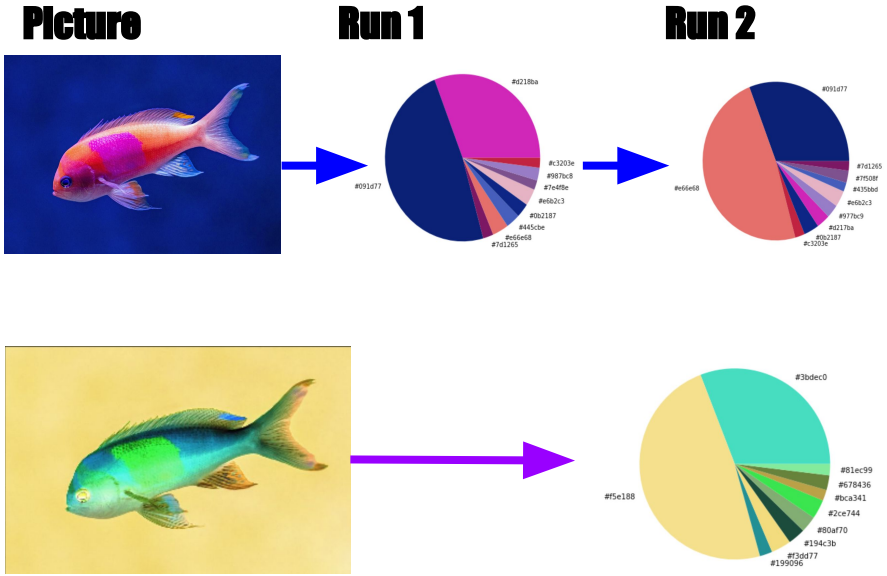
[206.4*6.69,152.68*6.69,132.05*6.69, . . .]

Data Processing:
An example

```
pi@raspberrypi:~/code $ ./RUN_ALL.sh
You are beginning the process!
Are you looking to parse through all or one image? one
You are now looking to extract the color information from a single picture!
Enter the exact color IMAGE NAME you are wanting to use as it appears in '/home/pi/ext_drive': pic.jpg
/home/pi/ext_drive/pic.jpg
READ THE IMAGE SIR
converted image
reshaped image
clustering
Grabbing RGB values and percentages
[154.84070796 158.15929204 135.7079646 ] 4.28%
[109.5915493  103.          85.41549296] 7.88%
[63.44086022 58.32258065 63.70967742] 10.20%
[35.1799308  29.23529412 29.10034602] 14.54%
[115.71282051  62.73247863   4.97606838] 63.10%
Writing out to file now
You have gone through 1 images.
The name of the file as it appears is /home/pi/ext_drive/pic.jpg_testing_colorinfo.txt
Beginning to pickle data...
Enter the file name of color info file you want to transfer data to.
The name should be as it appears in '/home/pi/ext_drive/': pic.jpg_testing_colorinfo.txt
/home/pi/ext_drive/pic.jpg_testing_colorinfo.txt
/home/pi/ext_drive/pic.jpg_testing_colorinfo.txt
pic.jpg_testing_colorinfo
(1, 15)
The pickle files are x_data_pic.jpg_testing_colorinfo y_data_pic.jpg_testing_colorinfo
pi@raspberrypi:~/code $
```

Data Processing: *Obstacles in colors*

- Color wheels worked initially
  - Gave different colors if run twice
- When background remover applied, color wheel became less useful
- MCC had to come up with a script



Data Processing: *Color identification*

- New script provides color names
- Colors can be extracted as RGB values
- Main colors displayed in color bar
- Percentage of RGB value can be printed into array
- MCC now closer to pulling color into classifier

Picture

Color Bar

Output Code

```
Actual colour name: None closest colour name: lightsteelblue
Actual colour name: None closest colour name: dimgray
Actual colour name: None closest colour name: deeppink
Actual colour name: None closest colour name: firebrick
Actual colour name: None closest colour name: black
(array([158.14881036, 201.39976254, 216.86114638]), '12.97%')
(array([ 78.34795759, 122.07240443,  94.57138093]), '13.26%')
(array([246.44202366,  64.04333565, 120.01836796]), '15.33%')
(array([165.69375466,   8.82057311,  53.89559981]), '16.99%')
(array([19.55387421, 36.73438896, 16.1252119 ]), '41.45%')
```

Server
What it looks like

- The RPI is set up to host both a web server and a SSH platform. Ultimately, we would like to have the server have images placed into it , processed, and return a color. Further implementations would incorporate returning size, shape, and brightness.
- One can access the server web page if you are on the ACU network or use a VPN tunnel. This is poised to provide a user friendly interface to use the trained classifier.

Classifier Choice

- We trained using several different classifiers, and several different versions.
- Shown below is the best score we got training on all our colors

Neural Network **(sgd)**: 0.388

Trees: 0.111 (we have 9 colors, so this is basically a random guess)

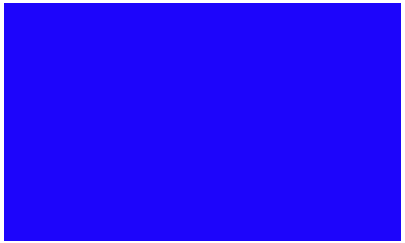
Nearest Neighbor: 0.15

Naive Bayes: 0.15

Example Images

Prediction:
Blue

Prediction:
Purple



when your lvl1 so you
equip anything you can find

Prediction:
White

Prediction:
Green

Score on Testing Set

- Different combinations gave different scores. [Yellow, Blue, Green, Pink, Red, White, Purple, Black, Brown]
- ALL 9 COLORS: 0.388 (score from training set is 0.531)
- FIRST 6 COLORS: 0.522
- Maybe some colors conflict and confuse the classifier, such as purple vs blue seen on the previous slide

How work was distributed

Roy:

- Set up SSH capabilities
- Wrote working color identifier
- Integrated high pass filter to work for finding edges
- Set up parsing file to separate colors
- Set up code to get X_train, y_train
- Ran classifiers
- Worked on Slides

Keaton:

- Found largest area on picture
- Used contours to remove background
- Uploaded data to RPi
- Parsed files into folders by color
- Classified and scored data
- Made Slides
- Enhanced score of Neural Network

Christine

- Worked on color wheel
- Worked on pulling color out of pictures and appending pixels to array
- Worked on uploading data to RPi
- Worked on parsing colors
- Worked on classifier
- Worked on Slides
- Worked on improving classifier score

What just didn't work

- We tried blob detector, but couldn't quite get it to do what we needed
- Trying to answer too many questions; it was much more in depth than anticipated

Future Modifications

- Would run cv2 from laptop camera and guess continuously
- Would use background remover to narrow down features
- Would have multiple classifiers
    - Each having a “confidence” in the answer
    - Use this as a way to predict which classifier works with
- Would have trained classifier on website with a user friendly interface

Conclusions

- This is much harder and more involved than it initially appeared!
- More preprocessing to find the main subject likely needs to be utilized
- Potentially having larger training sets could help, although we already have huge data sets

Our AWESOME GITHUB REPO!!!

Sources

[1] World Health Organization, *Global Data On Visual Impairments,* 2010, https://www.who.int/blindness/GLOBALDATAFINALforweb.pdf

[2] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[3] Jake Vanderplas, *Python Data Science Handbook: Essential Tools for Working with Data*, https://jakevdp.github.io/PythonDataScienceHandbook/

[4] D. Gurari, Y. Zhao, M. Zhang, N. Bhattacharya, *Captioning Images Taken by People Who are Blind,* University of Texas at Austin. arXiv 20 Feb 2020.

[5] Viz-Wiz, V*isual Question Answering*, https://vizwiz.org/tasks-and-datasets/vqa/

[6] Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

Questions?