CS M148

Homework 2

KRISH PATEL

Q1) **Perceptron:**

a) Weights are only updated when incorrect predictions are made. Thus only 3 updates.

$$w = 0 + y_1 x_1 + 0 + y_3 x_3 + y_4 x_4$$
$$w = y_1 x_1 + y_3 x_3 + y_4 x_4$$

b) $d = 3$, thus $x_i = [1, x_1, x_2]^T$

$$w = 1[1, 1, 0]^T - 1[1, 1, -3]^T + 1[1, 3, -1]^T$$

$$w = [1, 3, 2]^T$$

Now, the prediction would be $w^T x$

which would be $[1, 3, 2]^T [1, 1, 0]$

$= 1.1 + 3.1 + 2.0 = 1 + 4 + 0$

$= 5$

Sign(5) = 1, thus, it makes a correct prediction on x_1

c) Logistic regression, outputs probabilities, i.e., the value of y strictly lies between 0, and 1, which is interpreted as the probability of the positive class (usually threshold is 0.5). However, for this perceptron, the values are -1 and +1, depending on the sign of the output (A step function) Only makes updates, when the predicted output doesn't match ~~actual output~~ true.

(Q2) a) For the hidden layers, the activations choices were
- ReLU (Different variations include: ReLU, PReLU, ELU, Leaky ReLU)
  - This is simple to implement, especially relu, where gradient is 1 if output of neuron gives a positive output, else 0 if negative.
- Tanh and Sigmoid are also viable choices, they squash the output to [-1, 1] and [0, 1], and are also straightforward to calculate the gradient.

For the binary output layer, the activation function that can be used is a sigmoid.
This is because it outputs probabilies between 0 and 1, and can interpret the output as the likelihood of belonging to one of the classes.

b) $z_1 = w_{11}x_1 + w_{12}x_2 + w_{10}$
$= 0.9 \times 2 + 0.4 \times -3 + 0$
$= 0.6$
Neuron 1 uses RELU, thus $f(z_1) = \underline{0.6}$

$z_2 = w_{21}x_1 + w_{22}x_2 + w_{20}$
$= -1.5 \times 2 - 0.7 \times -3 + 0$
$= -0.9$
$f(z_2) = \frac{1}{1+e^{-(-0.9)}} = \frac{1}{1+e^{0.9}} \approx 0.289$

$\hat{y} = -0.6 \times 0.2 + 1.6 \times 0.289 + 0 \approx 0.3424$
$\uparrow$
w_{30}
≈ 0.34

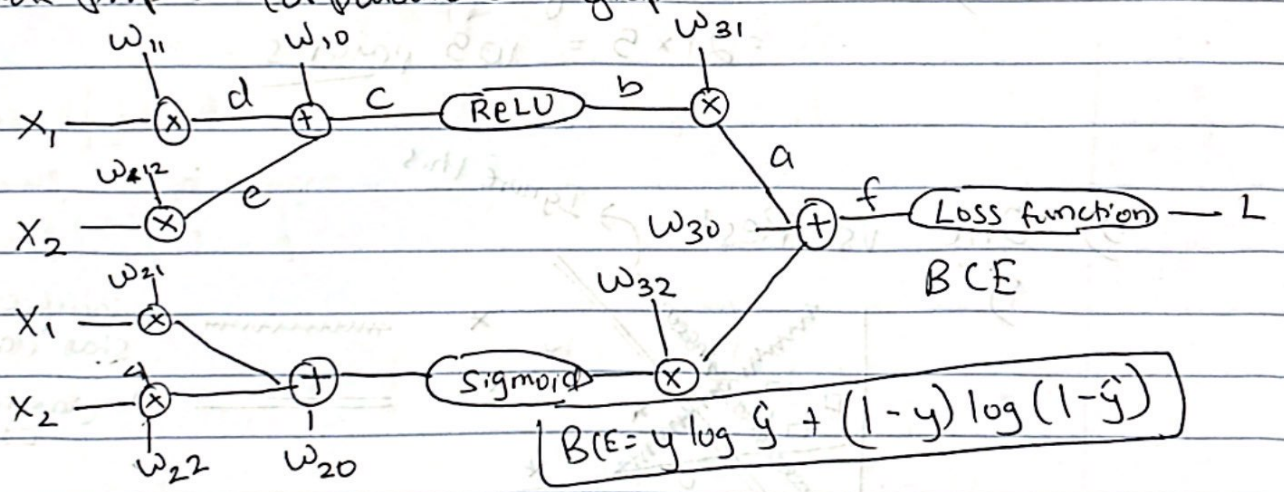
c) Binary cross entropy Loss.

$$L(y,\hat{y}) = \frac{-1}{N}\sum_{i=1}^{N}\left(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right)$$

$N=1$ loss $= 1\log(0.34) + (0)\log(1-0.34)$

$= -(-1.071775)\ldots \simeq 1.07$

d) Back prop → computational graph



$\frac{\partial L}{\partial f}$ = Using formula of BCE : $\frac{\partial L}{\partial \hat{y}} = \frac{-y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$

B through add gate : $\frac{\partial L}{\partial a} = \frac{\partial L}{\partial f}$ $\frac{\partial L}{\partial b} = \frac{\partial a}{\partial b}\cdot\frac{\partial L}{\partial a} =$

~~through~~ $\frac{\partial a}{\partial b} = w_{31}$ derivative through relu = 1 (As positive)

Thus $\frac{\partial L}{\partial c} = 1 \times w_{31} \cdot \frac{\partial L}{\partial a}$. $\frac{\partial L}{\partial e} = \frac{\partial L}{\partial c}$ (Add gate)

$$\frac{\partial L}{\partial w_{12}} = \frac{\partial e}{\partial w_{12}} \cdot \frac{\partial L}{\partial e} = x_2 \cdot w_{31} \cdot \left(\frac{-y}{\hat{y}} + \frac{1-y}{1-\hat{y}}\right)$$

Plugging in values : $1.7523\ldots \simeq 1.75$

e) The number of parameters in (b) is

6 (weight terms) + 3 (bias terms) = 9 total params

$W_1 \rightarrow$ 3 params, $W_2 \rightarrow$ 3 params

$W_3 \rightarrow$ 3 params.

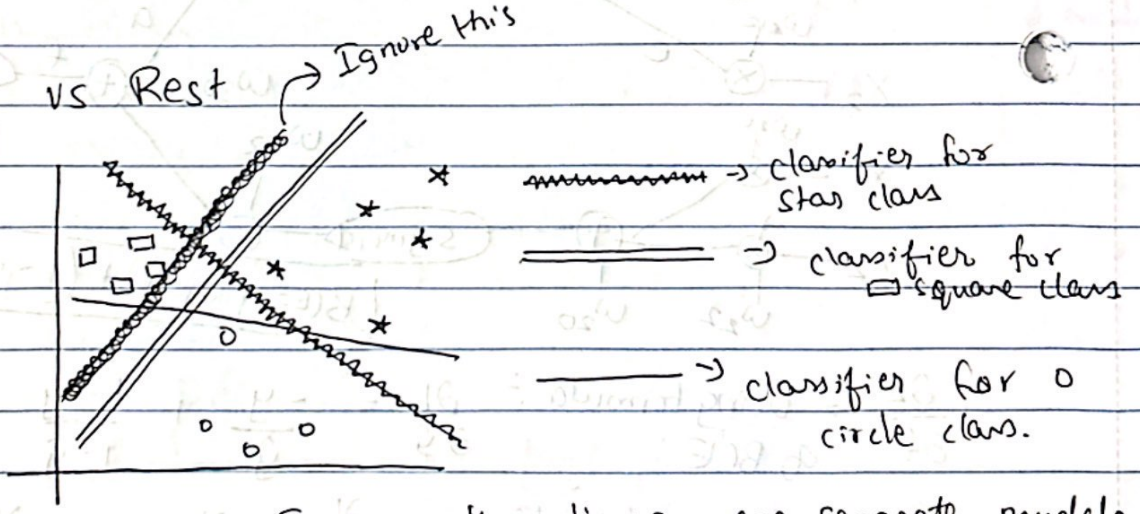
Q3) Multi-class Classification

a) One Vs Rest → Train K binary classifiers, where each model has two classes → probability that it is class k, and probability it is not class K.

Total number of params for 1 classifier:
20 (for 20 features (size of w)) and 1 for bias term.
Thus, total number of params for K=5.
= 21 × 5 = 105 params.

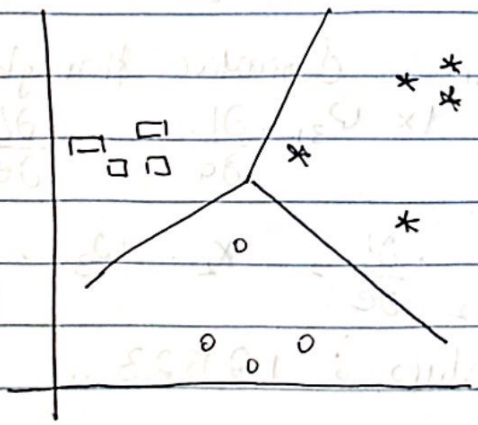
b) One vs Rest

i)



Each of these lines are separate models.

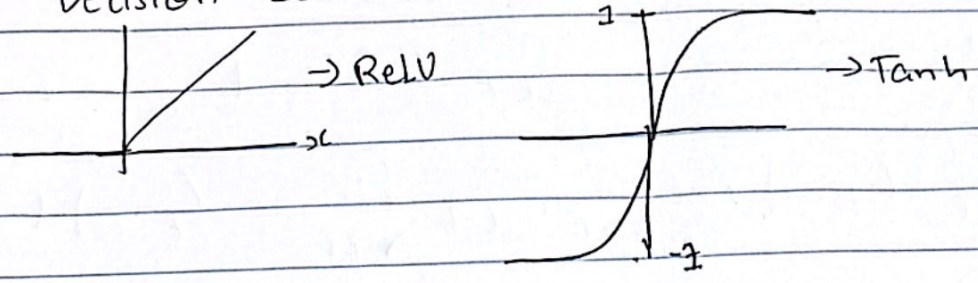
ii)



Multinomial linear boundary

Decision Boundaries

(Q4)



From the above diagrams, we can see that ReLU has a linear activation (~~form~~ when $x>0$) while tanh is more smoother.) (considering that a neural network pieces together different neurons based on activation functions, Tanh would give smoother transitions from one decision boundary of a neuron to another, while relu would look like linear boundaries pieced together. thus, ~~dtg~~ diagram (a) uses a tanh activation function, while (b) uses ReLU activation function.
((a) would look like 10 smooth functions pieced together.)
((b) would look like 10 ~~8~~ linear functions pieced together)

24W-COM SCI-M148 Homework 2 Coding Question

Name: Krish Patel

UID: 605 796 227

Submission Guidelines

1. Please fill in your name and UID above.
2. Please submit a **PDF printout** of your Jupyter Notebook to **Gradescope**. If you have any trouble accessing Gradescope, please let a TA know ASAP.
3. As the PDF can get long, please tag the respective sections to ensure the readers know where to look.

Overview

This coding question is about training and explaining what neural networks are doing with LIME (short for Local Interpretable Model-agnostic Explanations).

We use a small image dataset called MNIST. It is a dataset of handwritten digits that is commonly used for training image classification models.

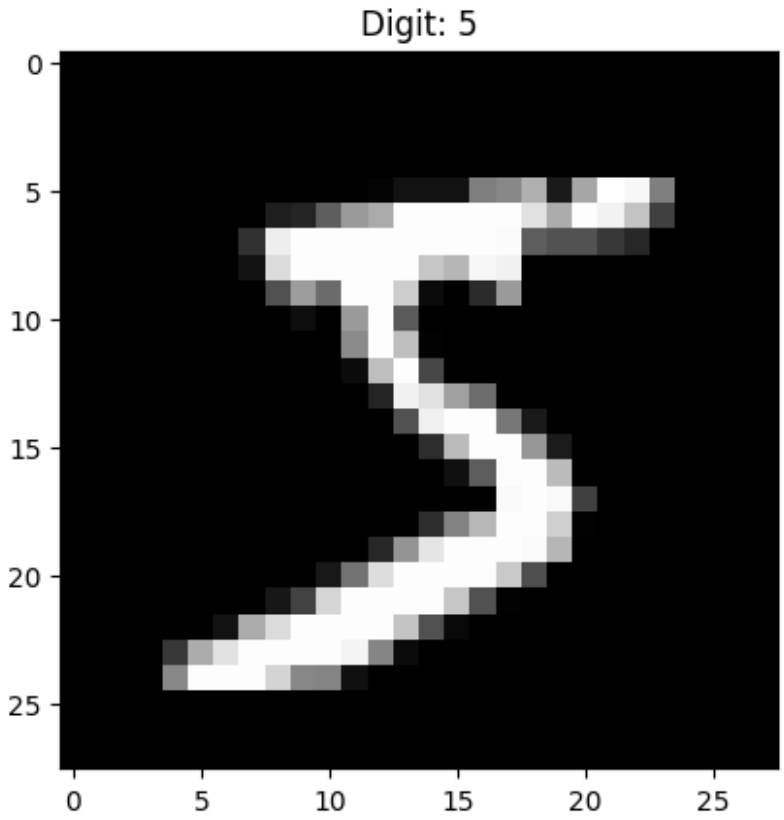
```
import numpy as np
import matplotlib.pyplot as plt
from skimage.color import gray2rgb, rgb2gray, label2rgb # since the
code wants color images

from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784')
# make each image color so lime_image works correctly
X_vec = np.stack([gray2rgb(iimg) for iimg in
mnist.data.values.reshape((-1, 28, 28))],0).astype(np.uint8)
y_vec = mnist.target.astype(np.uint8)
```

```
/opt/homebrew/lib/python3.11/site-packages/sklearn/datasets/
_openml.py:1022: FutureWarning: The default value of `parser` will
change from `'liac-arff'` to `'auto'` in 1.4. You can set
`parser='auto'` to silence this warning. Therefore, an `ImportError`
will be raised from 1.4 if the dataset is dense and pandas is not
installed. Note that the pandas parser may return different data
types. See the Notes Section in fetch_openml's API doc for details.
  warn(
```

```
%matplotlib inline
fig, ax1 = plt.subplots(1,1)
ax1.imshow(X_vec[0], interpolation = 'none')
ax1.set_title('Digit: {}'.format(y_vec[0]))
```

```
Text(0.5, 1.0, 'Digit: 5')
```



Setup a Pipeline

Here we make a pipeline for processing the images where basically we flatten the image back to 1d vectors and then use a neural network with one hidden layer.

```
from sklearn.pipeline import Pipeline
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import Normalizer

class PipeStep(object):
    """
    Wrapper for turning functions into pipeline transforms (no-
fitting)
    """
    def __init__(self, step_func):
        self._step_func=step_func
    def fit(self,*args):
        return self
    def transform(self,X):
        return self._step_func(X)
```

```
makegray_step = PipeStep(lambda img_list: [rgb2gray(img) for img in
img_list])
flatten_step = PipeStep(lambda img_list: [img.ravel() for img in
img_list])

simple_nn_pipeline = Pipeline([
    ('Make Gray', makegray_step),
    ('Flatten Image', flatten_step),
    ('NN', MLPClassifier()) # This is a neural network with 1 hidden
layer
    ])
```

Now, let's do the train-test split to have 55% data in the train set with the random state set to 0:

```
print("Shape of X_vec: {}".format(X_vec.shape))
percent = 0.55

X_train, X_test, y_train, y_test = X_vec[:int(percent*len(X_vec))],
X_vec[int(percent*len(X_vec)):], y_vec[:int(percent*len(X_vec))],
y_vec[int(percent*len(X_vec)):]
```

```
Shape of X_vec: (70000, 28, 28, 3)
```

```
simple_nn_pipeline.fit(X_train, y_train)
```

```
Pipeline(steps=[('Make Gray', <__main__.PipeStep object at
0x14ba2f050>),
                ('Flatten Image', <__main__.PipeStep object at
0x30fad7dd0>),
                ('NN', MLPClassifier())])
```

Now, let's get the training and test scores.

```
print('Training set score: ' + str( simple_nn_pipeline.score(X_train,
y_train)))
print('Test set score: ' + str( simple_nn_pipeline.score(X_test,
y_test)))
```

```
Training set score: 1.0
Test set score: 0.9731111111111111
```

```
%load_ext autoreload
%autoreload 2
import os,sys
try:
    import lime
except:
```

```
    %pip install lime
    import lime

Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275.7/275.7 kB 3.3 MB/s eta
0:00:00a 0:00:01
etadata (setup.py) ... ent already satisfied: matplotlib in
/opt/homebrew/lib/python3.11/site-packages (from lime) (3.8.0)
Requirement already satisfied: numpy in
/opt/homebrew/lib/python3.11/site-packages (from lime) (1.26.1)
Requirement already satisfied: scipy in
/opt/homebrew/lib/python3.11/site-packages (from lime) (1.11.3)
Requirement already satisfied: tqdm in
/opt/homebrew/lib/python3.11/site-packages (from lime) (4.66.1)
Requirement already satisfied: scikit-learn>=0.18 in
/opt/homebrew/lib/python3.11/site-packages (from lime) (1.3.2)
Requirement already satisfied: scikit-image>=0.12 in
/opt/homebrew/lib/python3.11/site-packages (from lime) (0.22.0)
Requirement already satisfied: networkx>=2.8 in
/opt/homebrew/lib/python3.11/site-packages (from scikit-image>=0.12-
>lime) (3.2.1)
Requirement already satisfied: pillow>=9.0.1 in
/opt/homebrew/lib/python3.11/site-packages (from scikit-image>=0.12-
>lime) (10.1.0)
Requirement already satisfied: imageio>=2.27 in
/opt/homebrew/lib/python3.11/site-packages (from scikit-image>=0.12-
>lime) (2.34.0)
Requirement already satisfied: tifffile>=2022.8.12 in
/opt/homebrew/lib/python3.11/site-packages (from scikit-image>=0.12-
>lime) (2024.2.12)
Requirement already satisfied: packaging>=21 in
/Users/krishpatel/Library/Python/3.11/lib/python/site-packages (from
scikit-image>=0.12->lime) (23.2)
Requirement already satisfied: lazy_loader>=0.3 in
/opt/homebrew/lib/python3.11/site-packages (from scikit-image>=0.12-
>lime) (0.3)
Requirement already satisfied: joblib>=1.1.1 in
/opt/homebrew/lib/python3.11/site-packages (from scikit-learn>=0.18-
>lime) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/homebrew/lib/python3.11/site-packages (from scikit-learn>=0.18-
>lime) (3.2.0)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib->lime)
(1.1.1)
Requirement already satisfied: cycler>=0.10 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib->lime)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
```

```
/opt/homebrew/lib/python3.11/site-packages (from matplotlib->lime)
(4.43.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib->lime)
(1.4.5)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib->lime)
(3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/krishpatel/Library/Python/3.11/lib/python/site-packages (from
matplotlib->lime) (2.8.2)
Requirement already satisfied: six>=1.5 in
/opt/homebrew/lib/python3.11/site-packages (from python-dateutil>=2.7-
>matplotlib->lime) (1.16.0)
Building wheels for collected packages: lime
  Building wheel for lime (setup.py) ... e: filename=lime-0.2.0.1-py3-
none-any.whl size=283834
sha256=365a2617c15910d2e1d352c7417eddaac734208561d9ebfbd6c65485996cefc
5
  Stored in directory:
/Users/krishpatel/Library/Caches/pip/wheels/85/fa/a3/9c2d44c9f3cd77cf4
e533b58900b2bf4487f2a17e8ec212a3d
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1

[notice] A new release of pip is available: 23.3.2 -> 24.0
[notice] To update, run: python3.11 -m pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

```
from lime import lime_image
from lime.wrappers.scikit_image import SegmentationAlgorithm
explainer = lime_image.LimeImageExplainer(verbose = False)
segmenter = SegmentationAlgorithm('quickshift', kernel_size=1,
max_dist=200, ratio=0.2)
```

```
/opt/homebrew/lib/python3.11/site-packages/tqdm/auto.py:21:
TqdmWarning: IProgress not found. Please update jupyter and
ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

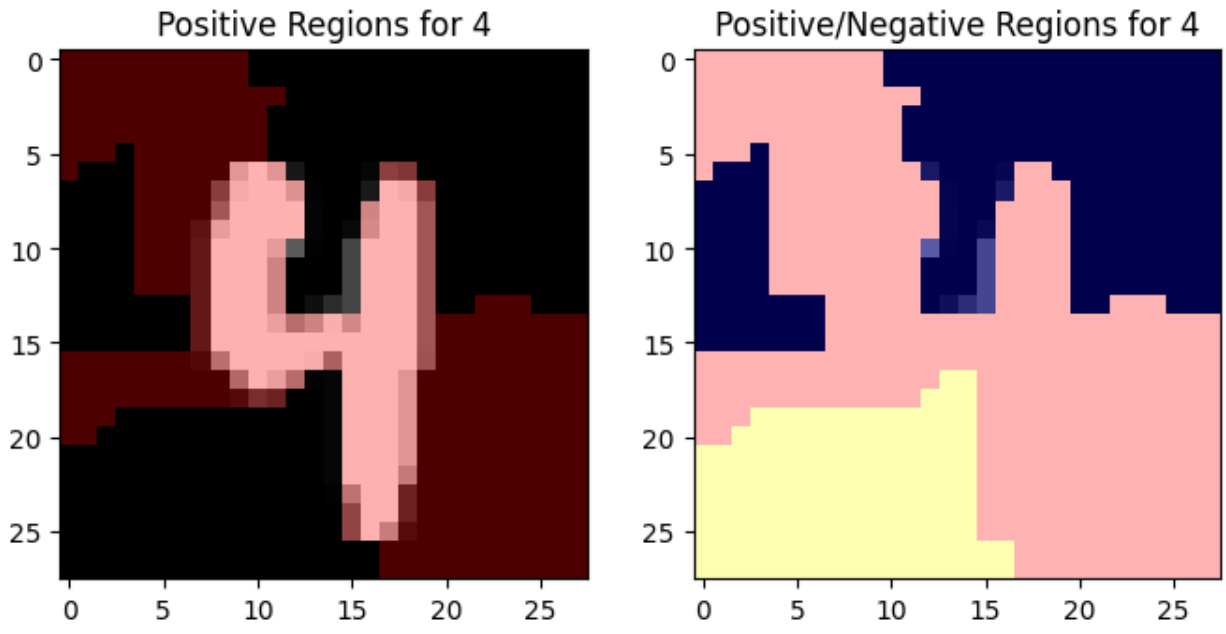
```
%%time
explanation = explainer.explain_instance(X_test[0],
                                         classifier_fn =
simple_nn_pipeline.predict_proba,
                                         top_labels=10, hide_color=0,
num_samples=10000, segmentation_fn=segmenter)
```

```
100%|██████████| 10000/10000 [00:01<00:00, 5487.39it/s]
```

```
CPU times: user 11.7 s, sys: 800 ms, total: 12.5 s
Wall time: 1.93 s
```

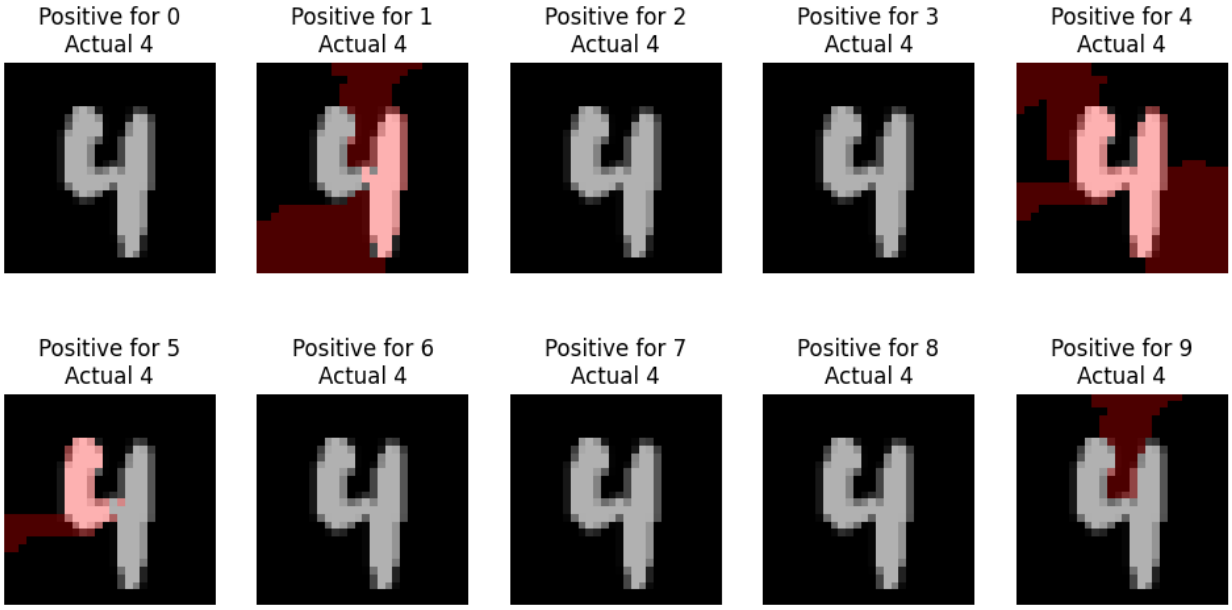
```
temp, mask = explanation.get_image_and_mask(y_test.values[0],
positive_only=True, num_features=10, hide_rest=False, min_weight =
0.01)
fig, (ax1, ax2) = plt.subplots(1,2, figsize = (8, 4))
ax1.imshow(label2rgb(mask,temp, bg_label = 0), interpolation =
'nearest')
ax1.set_title('Positive Regions for {}'.format(y_test.values[0]))
temp, mask = explanation.get_image_and_mask(y_test.values[0],
positive_only=False, num_features=10, hide_rest=False, min_weight =
0.01)
ax2.imshow(label2rgb(3-mask,temp, bg_label = 0), interpolation =
'nearest')
ax2.set_title('Positive/Negative Regions for
{}'.format(y_test.values[0]))
```

```
Text(0.5, 1.0, 'Positive/Negative Regions for 4')
```




```
# now show them for each class
fig, m_axs = plt.subplots(2,5, figsize = (12,6))
for i, c_ax in enumerate(m_axs.flatten()):
    temp, mask = explanation.get_image_and_mask(i, positive_only=True,
num_features=1000, hide_rest=False, min_weight = 0.01 )
    c_ax.imshow(label2rgb(mask,X_test[0], bg_label = 0), interpolation
= 'nearest')
    c_ax.set_title('Positive for {}\nActual {}'.format(i,
```

```
y_test.values[0]))
    c_ax.axis('off')
```



Gaining Insight

Can we find an explanation for a classification the algorithm got wrong

```
pipe_pred_test = simple_nn_pipeline.predict(X_test)
np.random.seed(0)
wrong_idx = np.random.choice(np.where(pipe_pred_test!=y_test)[0])
print('Using #{} where the label was {} and the pipeline predicted
{}'.format(wrong_idx, y_test.values[wrong_idx],
pipe_pred_test[wrong_idx]))
```

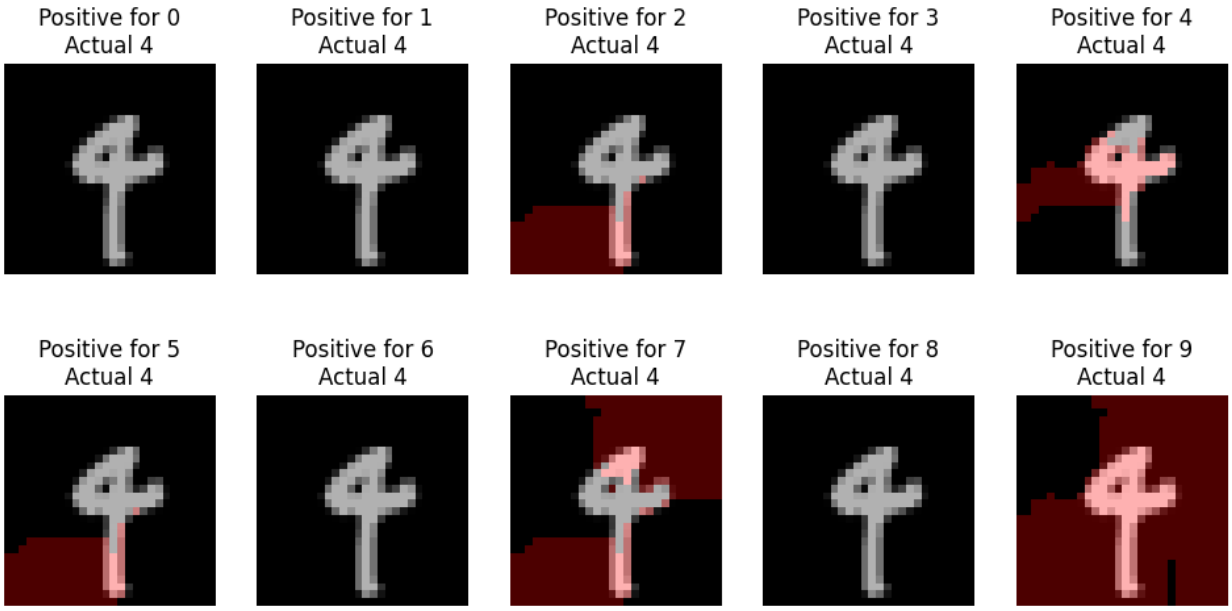
```
Using #23553 where the label was 4 and the pipeline predicted 9
```

```
%%time
explanation = explainer.explain_instance(X_test[wrong_idx],
                                         classifier_fn =
simple_nn_pipeline.predict_proba,
                                         top_labels=10, hide_color=0,
num_samples=10000, segmentation_fn=segmenter)
```

```
100%|██████████| 10000/10000 [00:01<00:00, 5392.52it/s]

CPU times: user 12.2 s, sys: 782 ms, total: 13 s
Wall time: 1.92 s
```

```
# now show them for each class
fig, m_axs = plt.subplots(2,5, figsize = (12,6))
for i, c_ax in enumerate(m_axs.flatten()):
    temp, mask = explanation.get_image_and_mask(i, positive_only=True,
num_features=10, hide_rest=False, min_weight = 0.01 )
    c_ax.imshow(label2rgb(mask,temp, bg_label = 0), interpolation =
'nearest')
    c_ax.set_title('Positive for {}\nActual {}'.format(i,
y_test.values[wrong_idx]))
    c_ax.axis('off')
```



Explain why the model misclassified this example based on the output of LIME:

According to the picture above, we can see that the model classified the handwritten digit 4 as a 9. From the LIME used earlier for the correctly predicted 4, we can see that the red portion is used to classify if the digit is a 9. For this specific example, which features a closed 4, the portion that contributes to the classification of 9 is highlighted, and thus, the classifier labels it as a 9 instead. The most likely explanation of the misclassification, thus, is the closed part of the 4 which is what the model uses to classify a 9 from the LIME analysis for the open 4 above.