Homework 2

Graded

Student

ARNAV KRISHNAKUMAR MARDA

Total Points

70 / 70 pts

Question 1

Perceptron **15** / 15 pts

1.1 (a) **5** / 5 pts

✔ **– 0 pts** Correct

– 2.5 pts partially correct

– 5 pts Incorrect or did not attempt

1.2 (b) **5** / 5 pts

✔ **– 0 pts** Correct

– 3 pts Incorrect w

– 2 pts Incorrect prediction

– 5 pts Incorrect or did not attempt

– 1 pt Minor mistake on computation of prediction

1.3 (c) **5** / 5 pts

✔ **– 0 pts** Correct

– 2.5 pts Incomplete/explanation partly makes sense

– 5 pts Incorrect or did not attempt

Question 2

Neural Network **20** / 20 pts

2.1 (a) **4** / 4 pts

✔ **– 0 pts** Correct

– 2 pts Incorrect choice/explanations on output layer activation function.

– 2 pts Incorrect or missing explanations on hidden layer activation function.

– 4 pts Incorrect or did not attempt

– 2 pts no explanation

– 2 pts Missing output layer part

2.2 (b) **3** / 3 pts

✔ **– 0 pts** Correct

– 1 pt Correct formulas with minor computation error

– 2 pts Some incorrect formulas

– 3 pts Incorrect or did not attempt

2.3 (c) **4** / 4 pts

✔ **– 0 pts** Correct

– 1 pt Correct formula for loss with minor computation error, or wrong log base.

– 2 pts Incorrect loss

– 1 pt Correct formula for derivative with minor computation error.

– 2 pts Incorrect derivative

– 4 pts did not attempt

– 1 pt Partially incorrect formula for derivative

2.4 (d) **6** / 6 pts

✔ **– 0 pts** Correct

– 1 pt Correct formula from chain rule, with one incorrect partial derivative or minor computation error.

– 2 pts Correct formula from chain rule, with two incorrect partial derivatives.

– 3 pts Correct formula from chain rule, with three incorrect partial derivative or minor computation error.

– 4 pts Correct formula from chain rule but did not present the final result/no computation.

– 6 pts Incorrect or did not attempt

– 5 pts incorrect formula or missing formula

– 2.5 pts partially correct formula

– 2 pts Missing final result

2.5 **(e)** **3** / 3 pts

✔ **– 0 pts** Correct

– 1 pt Missing/extra bias term

– 3 pts Incorrect or did not attempt

– 1.5 pts incorrect

– 3 pts unreadable

Question 3

Multi-class Classification **10** / 10 pts

3.1 **(a)** **5** / 5 pts

✔ **– 0 pts** Correct

– 2 pts Missing bias term

– 5 pts Incorrect or did not attempt

– 2 pts Other minor mistake

3.2 **(b)** **5** / 5 pts

✔ **– 0 pts** Correct

– 2.5 pts Incorrect OvR

– 2.5 pts Incorrect multinomial

– 5 pts Incorrect or did not attempt

– 1 pt Partially correct

Question 4

Decision Boundary **10** / 10 pts

4.1 **Neural Network (1 hidden layer with 10 ReLU)** **5** / 5 pts

✔ **– 0 pts** Correct

– 1.5 pts No Explanation

– 2.5 pts Incorrect or missing answer

4.2 **Neural Network (1 hidden layer with 10 tanh units)** **5** / 5 pts

✔ **– 0 pts** Correct

– 1.5 pts No explanation

– 2.5 pts Incorrect or missing answer

Question 5

Coding

15 / 15 pts

- ✔ **+ 5 pts** Train-test splitting
- ✔ **+ 2.5 pts** Computation of training accuracy
- ✔ **+ 2.5 pts** Computation of test accuracy
- ✔ **+ 5 pts** Explain based on LIME

+ 0 pts Missing answer

Questions assigned to the following page: 1.1, 2.1, 2.2, 1.2, 1.3, and 2.3

CS M148 HW 2

Arnav Marda

March 2, 2024

Note: This document does not contain the questions. Only the answers.

1. **Perceptron**

   (a) $w = y_1x_1 + y_3x_3 + y_4x_4$

   (b) $w = [1, 3, 2]$. The model will not make a mistake again and it will predict $+1$.

   (c) The activation function between the models is different. The Perceptron model uses a sign function while the logistic regression model uses a sigmoid function.

2. **Neural Networks**

   (a) Hidden Layers: We can use any activation functions. We could use ReLU, ELU, Leaky ReLU and variants of these models. Activation functions in the hidden layers are meant to make our model sparse and address the gradient vanish or exploding issues.
   Output Layers: sigmoid, softmax, or tanh activations to generate probabilities of the input being in each class makes them suitable options. We need specific activation functions to map the output of our neural network to the desired format.

   (b)

$$\begin{aligned} z_1 &= W_{11}X_1 + W_{12}X_2 + W_{10} = 0.6 \\ h_1 &= \text{ReLU}(z_1) = 0.6 \\ z_2 &= W_{21}X_1 + W_{22}X_2 + W_{20} = -0.9 \\ h_2 &= \text{Sigmoid}(z_2) = 0.2891 \\ \hat{y} &= W_{31}h_1 + W_{32}h_2 + W_{30} \approx 0.34 \end{aligned}$$

   (c)

$$\begin{aligned} \mathcal{L} &= -y\ln(\hat{y}) - (1-y)\ln(1-\hat{y}) = 1.0788 \\ \frac{\partial \mathcal{L}}{\partial \hat{y}} &= \frac{\partial}{\partial \hat{y}}(-y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})) = -\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}} = -2.9412 \end{aligned}$$

Questions assigned to the following page: 3.1, 3.2, 4.1, 2.4, 4.2, and 2.5

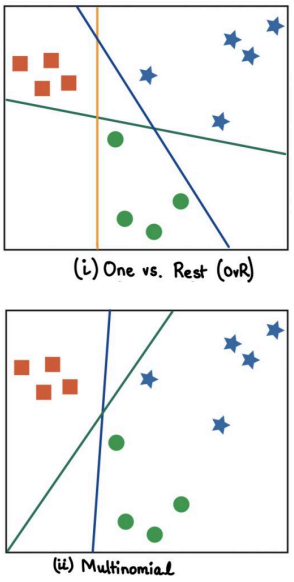
(d)

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W_{12}} &= \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W_{12}} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial_1} \cdot \frac{\partial h_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial W_{12}} \\ &= \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot W_{31} \cdot 1 \cdot X_2 \\ &= -1.7647\end{aligned}$$

(e) $3 \times 2 + 3 = 9$ parameters

3. **Multi-Class Classification**

(a) $21 \times 5 = 105$ parameters



(b)

4. **Decision Boundary**

(1) (c) since the decision boundary for a decision tree should be a straight line.

(2) (d) since random forest looks at the average of deep decision trees leading to lines that are almost straight.

(3) (b) since ReLU activation leads to a piecewise linear function.

(4) (a) since tanh activation function produces a curved decision boundary.m

Question assigned to the following page: 5

CSM148_W24_HW2_Q5

March 2, 2024

0.1 24W-COM SCI-M148 Homework 2 Coding Question

Name: Arnav Marda

UID: 405772661

0.1.1 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.

1 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.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage.color import gray2rgb, rgb2gray, label2rgb # since the code wants␣
↪color images
```

[2]:
```
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)
```

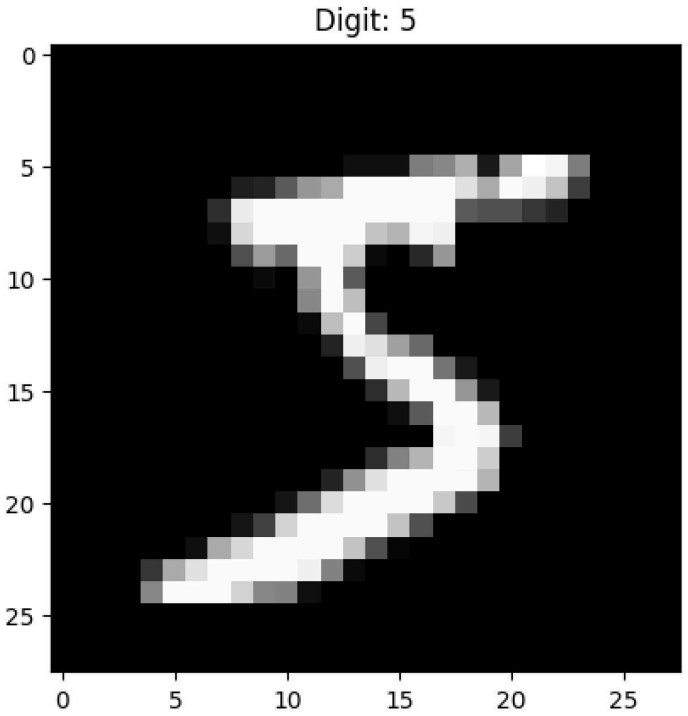
```
/opt/homebrew/lib/python3.11/site-packages/sklearn/datasets/_openml.py:1002:
FutureWarning: The default value of `parser` will change from `'liac-arff'` to
`'auto'` in 1.4. You can set `parser='auto'` to silence this warning. Therefore,
an `ImportError` will be raised from 1.4 if the dataset is dense and pandas is
not installed. Note that the pandas parser may return different data types. See
```

Question assigned to the following page: 5

```
the Notes Section in fetch_openml's API doc for details.
  warn(
```

[3]:
```
%matplotlib inline
fig, ax1 = plt.subplots(1,1)
ax1.imshow(X_vec[0], interpolation = 'none')
ax1.set_title('Digit: {}'.format(y_vec[0]))
```

[3]: Text(0.5, 1.0, 'Digit: 5')



2 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.

[4]:
```
from sklearn.pipeline import Pipeline
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import Normalizer

class PipeStep(object):
    """
```

Question assigned to the following page: 5

```
    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:

[5]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_vec, y_vec, train_size=0.
↪55, random_state=0)# WRITE YOUR CODE HERE
```

[6]:
```
# WRITE YOUR CODE HERE
simple_nn_pipeline.fit(X_train, y_train)
train_score = simple_nn_pipeline.score(X_train, y_train)
test_score = simple_nn_pipeline.score(X_test, y_test)
```

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

[7]:
```
print('Training set score: ' + str(train_score))
print('Test set score: ' + str(test_score))
```

```
Training set score: 1.0
Test set score: 0.9739047619047619
```

[8]:
```
%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)
```

Question assigned to the following page: 5

```
                                     275.7/275.7
kB 6.7 MB/s eta 0:00:00a 0:00:01
  Preparing metadata (setup.py) … done
Requirement already satisfied: matplotlib in
/opt/homebrew/lib/python3.11/site-packages (from lime) (3.7.2)
Requirement already satisfied: numpy in /opt/homebrew/lib/python3.11/site-
packages (from lime) (1.23.5)
Requirement already satisfied: scipy in /opt/homebrew/lib/python3.11/site-
packages (from lime) (1.11.2)
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) (1.3.0)
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) (3.1)
Requirement already satisfied: pillow>=9.0.1 in
/opt/homebrew/lib/python3.11/site-packages (from scikit-image>=0.12->lime)
(9.5.0)
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
/opt/homebrew/lib/python3.11/site-packages (from scikit-image>=0.12->lime)
(23.1)
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) (1.1.0)
Requirement already satisfied: cycler>=0.10 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib->lime) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib->lime) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib->lime) (1.4.4)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in
/opt/homebrew/lib/python3.11/site-packages (from matplotlib->lime) (3.0.9)
```

Question assigned to the following page: 5

```
Requirement already satisfied: python-dateutil>=2.7 in
/opt/homebrew/lib/python3.11/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) … done
  Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.whl
size=283838
sha256=8e25bec3da24f51b3a0bfbdb699dc1b092a56234fb287d1b7c48cfce5dcbc5b9
  Stored in directory: /Users/arnavmarda/Library/Caches/pip/wheels/85/fa/a3/9c2d
44c9f3cd77cf4e533b58900b2bf4487f2a17e8ec212a3d
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.
```

[9]:
```
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)
```

[10]:
```
%%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)
```

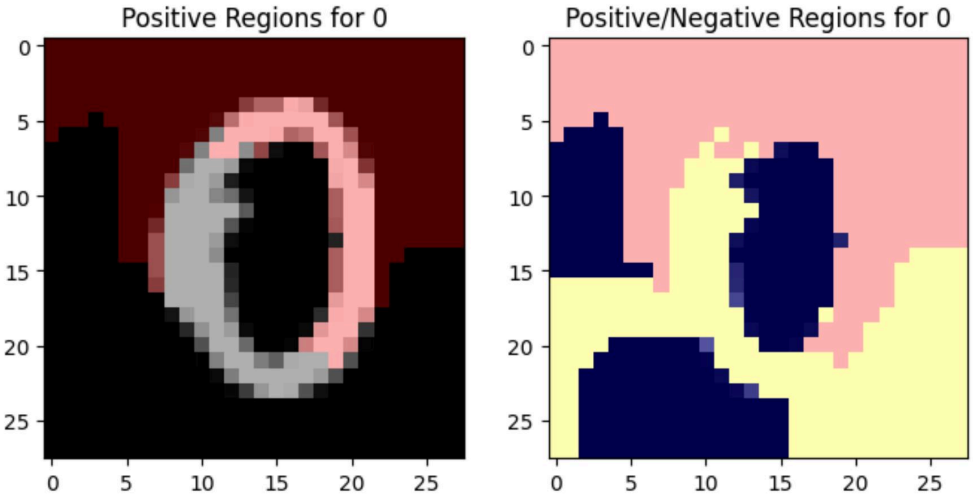
```
  0%|          | 0/10000 [00:00<?, ?it/s]

CPU times: user 12.1 s, sys: 562 ms, total: 12.7 s
Wall time: 1.93 s
```

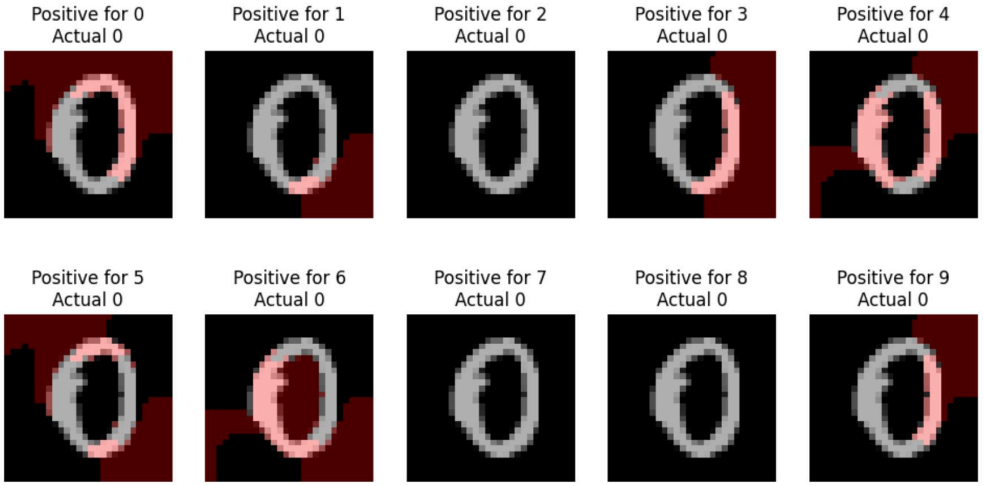
[11]:
```
temp, mask = explanation.get_image_and_mask(y_test.values[0],␣
 ↪positive_only=True, num_features=10, hide_rest=False, min_weight = 0.01)
fig, (ax1, ax2) = plt.subplots(1,2, figsize = (8, 4))
ax1.imshow(label2rgb(mask,temp, bg_label = 0), interpolation = 'nearest')
ax1.set_title('Positive Regions 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]))
```

Question assigned to the following page: 5

[11]: Text(0.5, 1.0, 'Positive/Negative Regions for 0')



[12]:
```
# 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')
```



Question assigned to the following page: 5

3 Gaining Insight

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

[13]:
```
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 #26878 where the label was 3 and the pipeline predicted 8
```

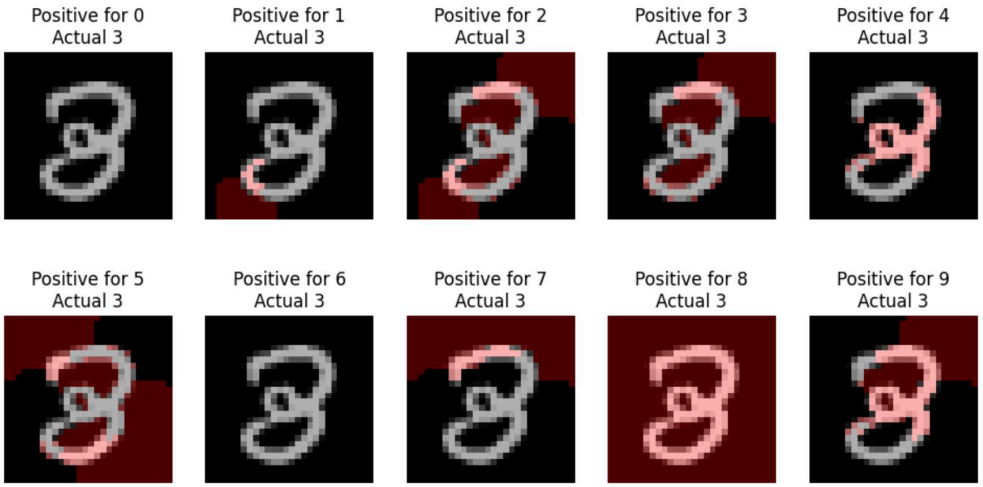
[14]:
```
%%time
explanation = explainer.explain_instance(X_test[wrong_idx],
                                         classifier_fn = simple_nn_pipeline.
 ↪predict_proba,
                                         top_labels=10, hide_color=0,␣
 ↪num_samples=10000, segmentation_fn=segmenter)
```

```
  0%|          | 0/10000 [00:00<?, ?it/s]

CPU times: user 10.9 s, sys: 572 ms, total: 11.5 s
Wall time: 1.71 s
```

[15]:
```
# 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')
```

Question assigned to the following page: 5



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

By analyzing the subplots, we can see that the model focused on specific features that are characteristic of the digit “8”: - Top-right diagonal stroke: The subplots for digits “0”, “1”, “7”, “8”, and “9” show positive contributions in the top-right diagonal region, which is a defining feature of these digits. - Lower left corner: Subplots for digits “6”, “8”, and “9” have positive contributions in the lower left corner, another feature commonly seen in these digits. - Disconnected strokes: The model might have misinterpreted the connected strokes of the digit “3” as two separate strokes, one resembling the top part of “8” and the other resembling the bottom part.

In conclusion, the LIME explanation suggests that the model misclassified the digit “3” as “8” due to:

- Overemphasizing features shared between “3” and “8”: The top-right diagonal stroke and potentially the disconnected strokes in the “3” might have been misinterpreted as features of “8”.
- Not capturing the global context of the digit: The model might have been overly focused on local features instead of considering the entire image, leading it to miss the overall structure of the digit “3”.