COMPSCI 4ML3

Introduction to Machine Learning

Winter 2021

Assignment Three

Tahseen Ahmed

ahmedt26

McMaster University

Revised: March 20th, 2021

Hello, and welcome to my submission for assignment three. Below you will find all of my solutions to the requested questions from the assignment instructions, as well as answers and justifications to programming components tasks.

1. **Programming Component: Task One**

   For the first part of this task, we use RBF SVM using various γ values $(0.600, 0.650, 0.700)$ and compare their performance against the test data. I personally add $\gamma = 0.999$ and try the RBF auto and scale options to see what they do. Below is the given output from my code:

```
========================================
RBF Kernel, Gamma 10.0  - accuracy:  0.42942743009320905
========================================
RBF Kernel, Gamma 0.700  - accuracy:  0.8621837549933422
RBF Kernel, Gamma 0.650  - accuracy:  0.8608521970705726
RBF Kernel, Gamma 0.600  - accuracy:  0.859520639147803
========================================
RBF Kernel, Gamma Scaled - accuracy:  0.36950732356857524
RBF Kernel, Gamma Auto   - accuracy:  0.26498002663115844
========================================
```

   The way we can interpret γ is that it adjusts how far the influence of a training point extends to and how sharp the decision boundary is. A very low γ acts like linear classifier almost, and a very high γ has sharp decision boundaries, and and the boundary can go around individual points, making islands with sharp curves.

   The γs we have been given to test all are rough islands around different points. It looks as if $\gamma = 0.70$ as it more accurately encapsulates the 'shape' each category of document resides in. This could be because the points of each class would be located 'nearby' one another, so much sharper boundaries help with correctly identifying different document classes.

   When we switch the TfidTransformer function to true, we improve most of our results:

```
========================================
TfidTransformer(use_idf = TRUE) for these cases
========================================
RBF Kernel, Gamma 10.0  - accuracy:  0.2756324900133156
========================================
RBF Kernel, Gamma 0.700  - accuracy:  0.9001331557922769
RBF Kernel, Gamma 0.650  - accuracy:  0.9014647137150466
RBF Kernel, Gamma 0.600  - accuracy:  0.9021304926764314
========================================
RBF Kernel, Gamma Scaled - accuracy:  0.26498002663115844
RBF Kernel, Gamma Auto   - accuracy:  0.26498002663115844
========================================
```

   We put less emphasis frequent and common words, and focus on the words that *actually* differentiate between the different categories of documents. This results in much higher accuracy for the models, except for the scaled and auto and the $\gamma = 10$. I think for the high gamma case it allows for even sharper boundaries, and so it starts overfitting much faster since there are less words to predict with.

2. **Programming Component: Task Two**

We plot a confusion matrix of our classifier. Note the order of labels outputted below are equivalent to the columns (left-to-right) and rows (top-to-bottom) of the matrix.

```
Labels:
['alt.atheism', 'comp.graphics', 'sci.med', 'soc.religion.christian']
Accuracy :  0.9001331557922769
array([[253,  13,  16,  37],
       [  3, 374,   6,   6],
       [  2,  44, 347,   3],
       [  3,  12,   5, 378]], dtype=int64)
```

The computer graphics document type was misclassified as medical science 44 times, the most out of any misclassifications.

3. **Programming Component: Task Three**

We adjust the ngram range values to see how it affects the accuracy of our model. We first get the dimensions of each ngram fitting, which is the second number in the tuple given (shape of matrix):

```
Single words dimension:       (2257, 35788)
Two-words dimension:          (2257, 264196)
Single and 2-word dimension:  (2257, 299984)
```

The default for CountVectorizer() is single words, hence the dimension is the same. Two-word combinations are plentiful, hence the large number. It is then obvious that 1 and 2-word combination range are just the sum of the two.

```
Below: RBF Kernel, Gamma 0.700, TfidTrasnformer use_idf = true, different ngram ranges:
=======================================
ngram range (1,1)  - accuracy:  0.9001331557922769
ngram range (2,2)  - accuracy:  0.8229027962716379
ngram range (1,2)  - accuracy:  0.8861517976031957
=======================================
```

The first accuracy is the same as the default CountVectorizer() because the default takes in single words. The second only accounts for two-word combinations. This appears to be not as accurate as single words. The one-word and two-word range results in a slightly better accuracy. If you average the exclusive single and two-word model accuracies, you get the (1,2) range accuracy. This output implies that single words are better for determining the class of document in this dataset.

When we try to analyze individual characters, we get considerably worse results. The dimensions for these matrices are quite massive:

```
Single Characters:  (2257, 35788)
1-2 Characters:     (2257, 299984)
1-3 Characters:     (2257, 759896)
1-4 Characters:     (2257, 1288592)
```

The model accuarcy is worse when using n-characters:

```
Below: RBF Kernel, Gamma 0.700, TfidTrasnformer use_idf = true, analyzing characters diffe
=======================================
ngram range (1,1)  - accuracy:  0.5346205059920106
ngram range (1,2)  - accuracy:  0.6837549933422103
ngram range (1,3)  - accuracy:  0.8035952063914781
ngram range (1,4)  - accuracy:  0.8408788282290279
=======================================
```

One or two characters are not effective in determining the class of document. I hypothesize that the greater ranges account for complete words in the documents, that's why their accuracies are higher than that of the ranges which hold one or two characters. If we were to increase the upper limit to maybe around 8-12 we would get similar accuracies as the two-word ranges.

In conclusion, it looks like using single words as tokens for RBF SVM with a gamma of around $\gamma = 0.700^{+}$ results in an accurate model.

For this question, we have been given following properties of the Gaussian Discriminant Analysis method for binary classification:

$$\textsf{(1)}:\ P(x \mid y=0, \mu_1, \mu_2, \beta) = \frac{1}{a} e^{-\|x-\mu_1\|_2^2}$$

$$\textsf{(2)}:\ P(x \mid y=1, \mu_1, \mu_2, \beta) = \frac{1}{a} e^{-\|x-\mu_2\|_2^2}$$

$$\textsf{(3)}:\ P(y=0 \mid \mu_1, \mu_2, \beta) = 1-\beta$$

We can deduce that:

$$\textsf{(4)}:\ P(y=1 \mid \mu_1, \mu_2, \beta) = \beta$$

From Lecture 12, Slide 4, we know that if we have μ_1, μ_2 and β then:

$$\textsf{(5)}:\ P(x, y) = P(x \mid y) \cdot P(y)$$

1. **Question 2A**

   We denote $\theta = \{\mu_1, \mu_2, \beta\}$ and $Z = \{(x^i, y^i)\}_{i=1}^N$. We start our proof below:

$$\begin{aligned}
\textsf{argmax}_\theta P(Z \mid \theta) &= \textsf{argmax}_\theta \prod_{i=1}^{N} P(x^i, y^i \mid \theta) \\
&= \textsf{argmax}_\theta \prod_i \frac{P(x^i, y^i, \theta)}{P(\theta)} \\
&= \textsf{argmax}_\theta \prod_i \frac{P(x^i \mid y^i, \theta) \cdot P(y^i \mid \theta) \cdot P(\theta)}{P(\theta)} \\
&= \textsf{argmax}_\theta \prod_i P(x^i \mid y^i, \theta) \cdot P(y^i \mid \theta) \cdot P(\theta) \\
&= \textsf{argmax}_\theta \left[\left(\prod_{i:y^i=0} \frac{1}{a} \cdot e^{-\|x-\mu_1\|_2^2} \cdot (1-\beta) \right) \left(\prod_{i:y^i=1} \frac{1}{a} \cdot e^{-\|x-\mu_2\|_2^2} \cdot (\beta) \right) \right] \\
&\textsf{argmax}_\theta (\frac{1}{a})^N \cdot ((1-\beta)^{n_0} \cdot \beta^{n_1}) \cdot (e^{-\sum_{i:y^i=0} \left\|x^i-\mu_1\right\|_2^2}) \cdot (e^{-\sum_{i:y^i=1} \left\|x^i-\mu_2\right\|_2^2})
\end{aligned}$$

   We don't really need the constant $\frac{1}{a}^N$ when maximizing, so it simplifies to:

$$\textsf{argmax}_\theta (((1-\beta)^{n_0} \cdot \beta^{n_1}) \cdot (e^{-\sum_{i:y^i=0} \left\|x^i-\mu_1\right\|_2^2}) \cdot (e^{-\sum_{i:y^i=1} \left\|x^i-\mu_2\right\|_2^2})$$

   We have thus estimated the parameters for μ_1, μ_2 and β.

2. **Question 2B**

Equations for reference:

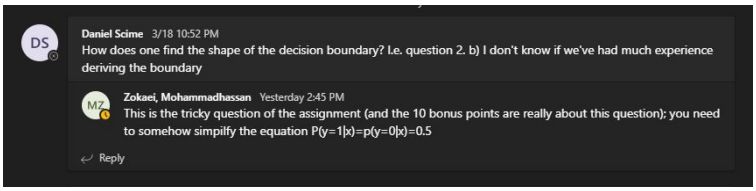
$$\textsf{(1)}:\ P(x \mid y=0, \mu_1, \mu_2, \beta) = \frac{1}{a} e^{-\|x-\mu_1\|_2^2}$$

$$\textsf{(2)}:\ P(x \mid y=1, \mu_1, \mu_2, \beta) = \frac{1}{a} e^{-\|x-\mu_2\|_2^2}$$

$$\textsf{(3)}:\ P(y=0 \mid \mu_1, \mu_2, \beta) = 1-\beta$$

$$\textsf{(4)}:\ P(y=1 \mid \mu_1, \mu_2, \beta) = \beta$$

$$\textsf{(5)}:\ P(x,y) = P(x \mid y) \cdot P(y)$$



We use this hint given by Dr. Ashtiani from the MS Teams discussions.

We know that μ_1, μ_2 and β are given, known constants.

If both classes share the same covariance matrix, then the boundary is linear, otherwise it's quadratic.

Proof. We begin to prove that $P(y=1 \mid x) = P(y=0 \mid x) = 0.50$.

My initial intuition for the above equation is because there are two classes, with the same distribution (not correlated), the data points take up half the dataset respectively. As the number of generated points increase to infinity for both classes, there is a 50/50 chance one of them will be from one class or the other. The probability $P(y=1 \mid x) + P(y=0 \mid x)$,covers all possible points, hence the value summing to 1.

We will attempt to prove this mathematically. We start with $P(y=1 \mid x) = 0.50$. The proof for $P(y=0 \mid x) = 0.50$ is nearly identical.

$P(y=1 \mid x)$	Starting definition
$\dfrac{P(x \mid y=1) \cdot P(y=1)}{P(x)}$	Bayes' Theorem
$\dfrac{P(x, y=1)}{P(x)}$	Equation 5, where $y=1$
$\dfrac{P(y=1, x)}{P(x)}$	$P(x,y) = P(y,x)$ since X and Y are uncorrelated (IID Assumption)

I haven't figured this proof out unfortunately, but I believe my intuition holds. □