**Perceptron Algorithm Model**

• Initial value of weights

While exploring ways to initialize suitable weight values, I checked accuracy of model on various weight values:

1. All weight values to 0

2. Random weight values between (-1,1)

3. A fixed value between 0 and 1

And on checking accuracy, I got following result for a combination of iterations and learning rate:

For Iteration=20 and LR = 0.08,

1. Random values: accuracy fluctuates in range (0. 91841 and 0.96025)

2. 0: accuracy is 0.9351

3. A Fixed value, 0.1 and 0.2 gave the accuracy value of 0.96025

So, I set the initial weight value to 0.1

Note: for unseen word from test dataset the accuracy is set to the default weight = 0.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Iterations** | **learning rate** | **Accuracy with Stop Words** | **Accuracy without Stop Words** |
| 3 | 0.08 | 0.7489539748953975 | 0.7405857740585774 |
| 9 | 0.08 | 0.8138075313807531 | 0.7907949790794979 |
| 15 | 0.08 | 0.9121338912133892 | 0.9246861924686193 |
| 20 | 0.08 | **0.9602510460251046** | **0.9456066945606695** |
| 25 | 0.08 | 0.9602510460251046 | 0.9456066945606695 |
| 30 | 0.08 | 0.9602510460251046 | 0.9456066945606695 |
| 3 | 0.1 | 0.7531380753138075 | 0.7489539748953975 |
| 9 | 0.1 | 0.7782426778242678 | 0.8870292887029289 |
| 15 | 0.1 | 0.7343096234309623 | 0.9016736401673641 |
| 20 | 0.1 | 0.9476987447698745 | 0.9351464435146444 |
| 25 | 0.1 | 0.8828451882845189 | 0.9351464435146444 |
| 30 | 0.1 | 0.9456066945606695 | 0.9351464435146444 |
| 3 | 0.12 | 0.7301255230125523 | 0.7280334728033473 |
| 9 | 0.12 | 0.7280334728033473 | 0.7698744769874477 |
| 15 | 0.12 | 0.9246861924686193 | 0.8723849372384938 |
| 20 | 0.12 | 0.9539748953974896 | 0.9393305439330544 |
| 25 | 0.12 | 0.9539748953974896 | 0.9393305439330544 |
| 30 | 0.12 | 0.9539748953974896 | 0.9393305439330544 |
| 3 | 0.14 | 0.7301255230125523 | 0.7280334728033473 |
| 9 | 0.14 | 0.7782426778242678 | 0.7615062761506276 |
| 15 | 0.14 | 0.891213389121339 | 0.8765690376569037 |
| 20 | 0.14 | 0.9393305439330544 | 0.9372384937238494 |
| 25 | 0.14 | 0.9393305439330544 | 0.9372384937238494 |
| 30 | 0.14 | 0.9393305439330544 | 0.9372384937238494 |
| 3 | 0.16 | 0.7301255230125523 | 0.7280334728033473 |
| 9 | 0.16 | 0.7343096234309623 | 0.7489539748953975 |
| 15 | 0.16 | 0.7510460251046025 | 0.9205020920502092 |
| 20 | 0.16 | 0.9539748953974896 | 0.891213389121339 |
| 25 | 0.16 | 0.9539748953974896 | 0.9435146443514645 |
| 30 | 0.16 | 0.9539748953974896 | 0.9435146443514645 |
| 3 | 0.2 | 0.7301255230125523 | 0.7280334728033473 |
| 9 | 0.2 | 0.7280334728033473 | 0.803347280334728 |
| 15 | 0.2 | 0.895397489539749 | 0.9246861924686193 |
| 20 | 0.2 | 0.8305439330543933 | 0.9456066945606695 |
| 25 | 0.2 | 0.9330543933054394 | 0.9351464435146444 |
| 30 | 0.2 | 0.9330543933054394 | 0.9351464435146444 |

Analysing the above table, for a certain value of learning rate (LR), accuracy increases with number of iterations and then when the weights are stable, it remains constant even when the iterations increase. In table above, accuracy remains stable after Iterations = 15 as weight values become stable by then. When stop words are removed, weight values become stable early before Iterations = 9 and accuracy value remain same after.

For a certain iteration count, accuracy value reaches its maximum values for a certain learning rate, and then fluctuates. In table above, for Iterations=20 (indicated by bold), accuracy reaches its maximum value (0.96) at learning rate of 0.08 and after that it fluctuates.

When stop words are removed, accuracy decreases as it happened in Naïve Bayes. This is because of the following reasons:

* Stop word is might be the word that has high frequency and which gives meaning to context.
* Removing stop words might change meaning of the context, some words may appear "too close" to each other. For e.g spam might look like ham

**Comparison of accuracies predicted by Perceptron Algorithm and Naïve Bayes:**

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy (with stop words)** | **Accuracy (without stop words)** |
| **Naïve Bayes** | 0.997134670487106 | 0.9914040114613181 |
| **Perceptron Algorithm** | 0.9602510460251046 | 0.9456066945606695 |

Looking at the accuracy values of both models, it looks like Naïve Bayes model performed better. Reasons are:

1. A stronger model is the one which handles decision boundaries well and makes suitable limiting assumptions.

2. Even though Perceptron Algorithm is a linear model, and it is harder for Naïve Bayes to handle decision boundary, it performs better because it handles new features very well compared to Perceptron rule.

3. With the help of **K-Laplace Smoothing**, it handles unseen features very well as compared to Perceptron rule when it comes to prediction on Test dataset. Its assumptions for new features are better, hence Naïve Bayes gives better accuracy than Perceptron Algorithm.