Perceptron

1) Explain the Perceptron algorithm for the binary classification case, providing its pseudo code.

```
train(data, iterations, y, weights):
    for j in range(0, iterations):
        for d in data:
            a = d . weights + b
            if y*a <= 0:
                weights += d * y
                b += y
        j += 1
    return weights
test(weights, data)
    a = weights . data
    return sign(a)
```

Where data is the training/test dataset, iterations is the number of times you wish to train on a given dataset, weights are the weight associated with each feature, b is the bias term and y is whether the classification is a positive or a negative.

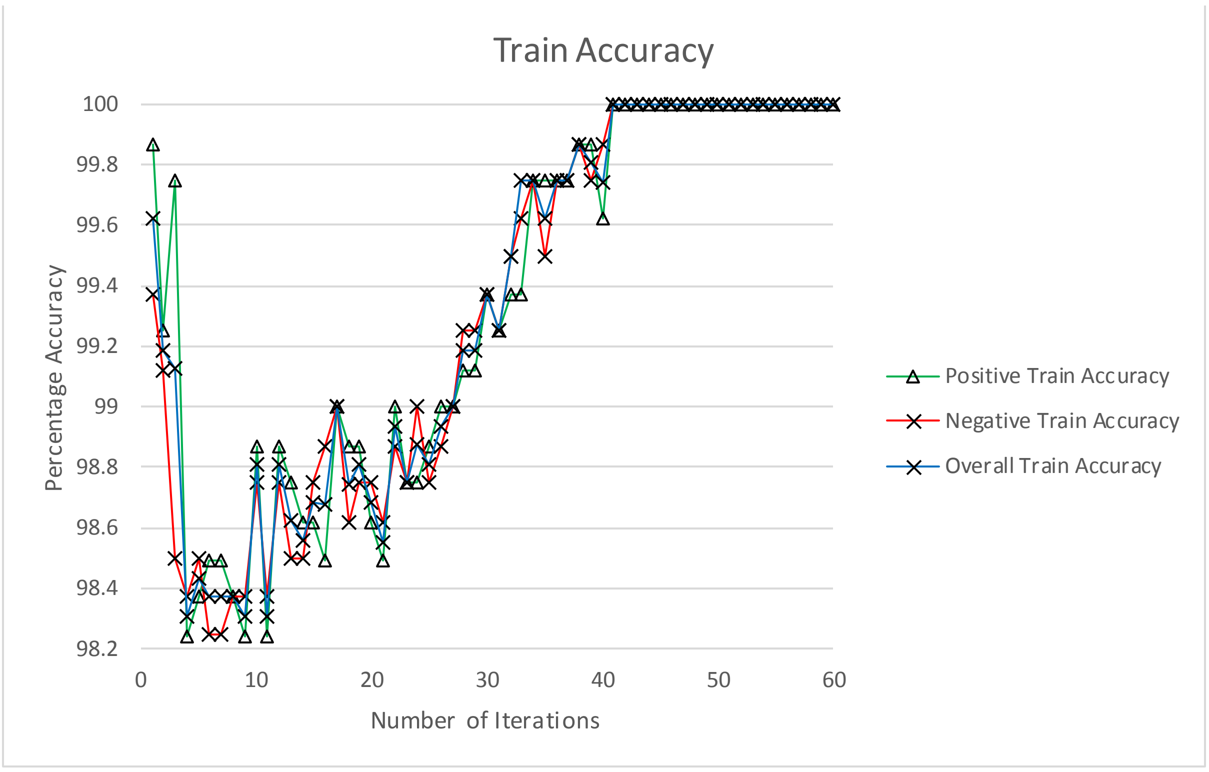
The data is passed to the training algorithm, it takes the form of an array with indices equal to the number of features present in the feature space. If a word from the feature-space occurs in the dataset then the index of that feature has a value of one. If a feature in the feature space does not occur in the dataset then the value at that index remains at zero.

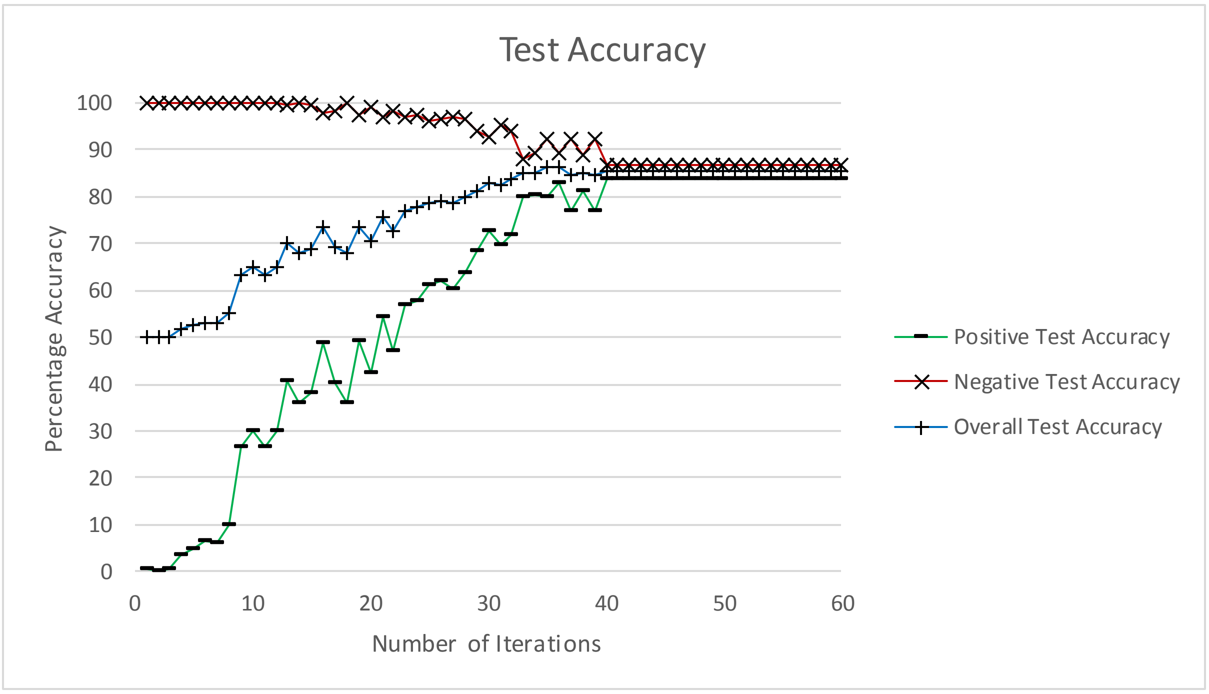
Each word that belongs to the feature space has a weight associated with it. These are all initialised to zero. The training then takes the scalar product of the weights and the dataset and calculates a, the activation. If the activation multiplied by y (which may take on the value of 1 or -1, depending on the classification of the training data) is less than or equal to zero then the perceptron has misclassified that dataset. If the data has been misclassified then the weight vector is then updated. The features of the misclassified dataset, d is multiplied by y then added to the weight vector so that the features that occurred in d are now more heavily considered in classification.

The bias term b is a constant offset that is applied independent of the content of the dataset being classified. It offsets the activation value to the region where classification can take place. When a misclassification occurs the bias term is updated by the value of y.

When an unknown dataset is passed to the test function the scalar product of both it and the weight vector are calculated. The sign of the activation determines its classification, if the activation is negative then the classification is a negative, if not then it is positive.

4) Plot the train error rate and test error rate against the number of iterations. According to your plot, what would be the ideal number of iterations to terminate the training?





According to the plot the train accuracy converges at 100% at around 40 iterations. This varies with each run of the perceptron so I would say that the ideal number of iterations is ~50 to be certain that the training has converged. No over-fitting occurs with more training iterations, the weight vector will never change once training accuracy has converged at 100% in this implementation.