# Sentiment Analysis with Machine Learning Techniques: Perceptron and Decision Tree

Beatrice Paoli

## 1 Introduction

In this paper we are going to test two Machine Learning algorithms for classifying movie reviews by sentiment (e.g. positive or negative). The two classifiers used are the Perceptron Classifier and the Decision Classifier, both trained in different ways, using different types of features, to test their performance. The dataset used for the tests is the polarity_dataset v2.0 from Bo Pang and Lillian Lee (`http://www.cs.cornell.edu/people/pabo/movie-review-data/`) containing 1000 positive and 1000 negative movie reviews and the tests are based on their work *"Thumbs up? Sentiment Classification using Machine Learning Techniques"* (`http://www.cs.cornell.edu/home/llee/papers/sentiment.pdf`).

### 1.1 Perceptron

Perceptron is an algorithm for supervised learning of binary classifiers. Given a dataset divided in samples and $n$ features $x(i)$ and their labels (positive or negative) $y^{(i)}$, the algorithm tries to learn a linear decision function (an hyperplane) $f(x) = \sum_{j=0}^{n} w_j x_j = w^T x$, if $f(x) \geq 0$ the sample is assigned to the positive class, otherwise to the negative class.
The algorithm iterates on all the train samples until the function predicts correctly all of their labels, otherwise the parameters of the function are updated as follows: $w = w + y^{(i)} x^{(i)}$ and $b = b + y^{(i)} R^2$ where $R = max||x^{(i)}||$. If this hyperplane exists the algorithm will find it and terminate and the data is said to be linearly separable.

### 1.2 Decision Tree

The Decision Tree algorithm is another algorithm for discrete classyfing of data (in our case with only two clasees). The algorithm tries to build a decision tree: a function that takes as input a vector of attribute values and returns a single class.
The decision is reached by performing a sequence of tests: each internal node in the tree corresponds to a test of the value of an attributes, the branches represent the possible values of the attribute and the leaves specify the class to be returned by the function. The algorithm implements a greedy strategy to find the smallest tree, with the least numbers of nodes, that minimizes the error committed on the train data.

## 2 Experiments and Results

The implementations used for Perceptron and Decision Tree where the ones from the SciKit-Learn library.
To experiment on the two algorithms, the text data was represented with the *bag of words* model: given a set of features that can appear in a document, a review is represented by a vector with the number of times each feature appears in the text. The tests are then done using 3-Folds Cross Validation Tests for each algorithm and for each type of feature tested.

| **Features** | **freq. or pres.** | **Perceptron** | **Decision Tree** |
|---|---|---|---|
| unigrams | freq. | 83 (+/- 3) | 63 (+/- 3) |
| unigrams | pres. | 83 (+/- 2) | 64 (+/- 2) |
| unigrams + bigrams | pres. | 84 (+/- 3) | 58 (+/- 2) |
| bigrams | pres. | 81 (+/- 5) | 55 (+/- 6) |
| top 2633 unigrams | pres. | 83 (+/- 2) | 62 (+/- 5) |
| unigrams + POS | pres. | 84 (+/- 2) | 61 (+/- 1) |
| adjectives | pres. | 79 (+/- 0) | 62 (+/- 4) |
| unigrams + position | pres. | 82 (+/- 3) | 61 (+/- 6) |

Table 1: Average 3-fold cross-validation accuracies and their error in percent.

**Unigrams and Bigrams** The first experiments were done using unigrams and/or bigrams as features. The use of only unigrams gave some of the highest accuracies for both algorithms, while the exclusive use of bigrams gave some of the worst results, even if they're supposed to be better at representing the context behind the text.

The combined use of both unigrams and bigrams improved the performance for the Perceptron classifier but worse for Decision Tree. And for last, limiting the maximum number of features didn't bring a significant change.

**Feature Frequence vs Presence** A distinction made for all the types of tests was wether to consider the feature frequencies in the documents or the feature presence. A comparison between the two was made for the unigrams and the results show no particular difference between the two. The other tests accounted only for feature presence (the same choice done in Bo Pang and Lillian Lee's work).

**Parts Of Speech** Another experiment was done by appending POS (Parts of Speech Tags) to each word, to distinguish their different usages and influence on the sentiment of the review (e.g. the difference between "I love this movie" and "This is a love story"). This distinction seemed to improve the Perceptron performance but lowered the Decision Tree performance. Another test was done by only including the adjectives of the reviews, since they're supposed to be the most informative for understanding the sentiment. This approach doesn't seem to be working as this led to the lowest performance of Perceptron.

**Words position** One last experiment was done by considering the words position in the text, as usually when writing a review, the beginning might have an overall sentiment statement and the ending might have the final author's view. To roughly account for this, each word was tagged with their position in the first quarter, last quarter or middle part of the review.

# 3 Conclusions

In general the performance of Decision Tree is significantly lower than Perceptron. This could be a problem of overfitting of the algorithm, making a tree too accurate for the train data that then fails for the test data (the algorithm used doesn't do any pruning of the tree to improve the accuracy). However, even by tweaking the parameters for the algorithm, such as the maximum number of nodes or the minimum size of the leaves, the outcome didn't improve. Overall, the Perceptron classifier performed way better than the Decision Tree classifier.