HUMBOLDT-UNIVERSITÄT ZU BERLIN

Application of Support Vector Machine Algorithm in E-Mail Spam Filtering

Julia Bluszcz, Daria Fitisova, Alexander Hamann, Alexey Trifonov

Advisor: Patrick Jähnichen

Abstract

The problem of spam detection has received broad interest for many years. One of the most common supervised learning methods applied for binary text classification problems like spam filtering is the Naïve Bayes (NB) algorithm. Treating NB as performance baseline, we propose a spam filter based on a Support Vector Machine (SVM) classifier. Taking into account SVMs ability to separate features in high-dimensional space, we show that the SVM is superior to the Naïve Bayes approach regarding the spam filtering problem.

Introduction

The problem of email classification, although broadly recognized for many years, still attracts the interest of researchers nowadays. Every webmail service provides its users with a built-in spam filter, which is crucial for a good user experience. **Our goal is to develop our own classifier which categorizes English-language emails as spam or ham**.

As emails are essentially text documents, the binary classification problem of separating incoming email into wanted messages (*ham*) and unsolicited messages (*spam*) should be treated as a Natural Language Processing problem. One of the most common methods applied within the described field is Naïve Bayes [1].

We have the hypothesis that using a SVM, working effectively in a multidimensional space, leads to a more accurate classifier. To evaluate different algorithms, the publicly available *SpamAssassin* training corpus is used. It provides a sizeable set of correctly classified emails. Using *scikit-learn*, a machine learning toolkit for *Python*, we implemented a SVM based text processing pipeline that classifies unseen emails.

Algorithms

At the core of the project is the application of the Support Vector Machine algorithm. A Naïve Bayes classifier is used as benchmark.

Support Vector Machine

The SVM, a supervised learning technique, is a conjunction of linear learning machine and kernel function. It creates a hyperplane which divides two classes of variables by maximizing the margin between the hyperplane and the closest data points by assigning a set of weights w to the feature vector. In the soft-margin SVM approach, which is applied in this project, some observations are allowed to overlap to wrong classes [2]. Formally, it can be presented in the form of finding an optimal solution to the quadratic problem [3]:

$$max_{\alpha\in\mathbb{R},w\in\mathbb{R}^d,\xi_i\geq 0}\gamma - C\sum_{i=1}^{n}\xi_i \quad \text{s.t.} \quad \forall_i\colon y_i(w^T x_i + b) \geq \|w\|\gamma - \xi_i \tag{1}$$

The primal problem can be next transformed into computationally less intense dual form:

$$max_{\alpha\in\mathbb{R}^n,0\leq\alpha\leq c}\sum_{i=1}^{n}\alpha_i - \frac{1}{2}(\alpha\circ y)^T X^T X(\alpha\circ y) \tag{2}$$

What makes the SVM attractive in email classification problems is its robustness and ability to handle large feature spaces. Since the algorithm is not trying to minimize the error rate, but rather separate the patterns in high dimensional space, the result is that SVMs are quite insensitive to the relative size of each class. To mention some disadvantages, SVM classifiers could be calculation intensive while training the model.

Naïve Bayes

Naïve Bayes is a simple, but powerful classifier based on a probabilistic model derived from the Bayes theorem. Basically, it determines the probability that an instance belongs to a class based on each of the feature value probabilities. Abstractly, NB is a conditional probability model: given a problem instance to be classified to any class $\{C_j\}_{1<j<n}$, represented by a vector of features (independent variables) $X \in \{x_1, \ldots, x_n\}$ it can be formally shown as:

$$C_{MAP} = argmax_{c\in C}P(x_1, \ldots, x_n|c)P(c) \tag{3}$$

$$C_{NB} = argmax_{c\in C}P(c_j)\prod_{x\in X}P(x|c) \tag{4}$$

The crucial benefit of the NB algorithm in technical concern is a reduction of the number of parameters needed for modeling. Namely, due to the independence assumption, we need $2n$ parameters to model $P(X|Y)$ instead of original $2(2^n - 1)$ [4]. This quality guarantees simplicity and quickness of the method.

Spam Filter Development and Evaluation

We divided the spam classification problem into four subproblems. The first step, preprocessing, is crucial as raw emails need to be cleaned up before they can be turned into feature vectors. Focussing on the email body, we remove existing HTML tags and normalize items like currency symbols, email addresses, or URLs, among others.

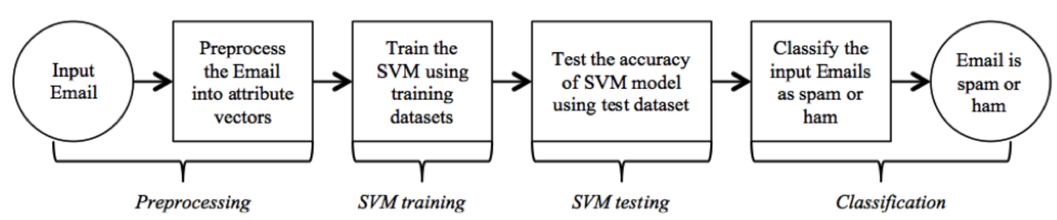

Fig. 1: Execution steps

Subsequently, the body text is transformed into a *bag-of-words* representation, which disregards grammar and word order and simply views a text as a multiset of its constituting terms. *Tf-idf*, a statistic that reflects the importance of a term in a document and the whole corpus simultaneously, is used as numerical representation in the feature vector. In addition, the length of the text and the number of capital letters are used as features.

Roughly 2/3 of the 6.046 emails in the SpamAssassin dataset were ham. To account for this class imbalance, we trained the parameters of the SVM using stratified 5-fold cross validation. A grid search helped in finding the optimal hyperparameters within a given set of combinations. To assess the classifier's out-of-sample performance, we kept 20% of the data set as hold-out and trained on the remaining 80%.

Our results show that a properly trained SVM reaches a significantly higher classification accuracy than the Naïve Bayes approach. To qualify the results, a *ZeroR* (Zero Rules) classifier, which simply always predicts the majority category of the training set, acts as baseline.

In our experiments, the Linear SVM was consistently better in terms of accuracy than SVMs with Gaussian or polynomial kernels. The results in tables 1 and 2 are thus for a Linear SVM with $C = 1$, which proved to be a sensible default. It can be concluded that the vast majority of mails was classified correctly.

Method	Accuracy
ZeroR	68.71%
Naïve Bayes	79.50%
Linear SVM	98.59%

Table 1: Classifier comparison

The *precision* of a classification task denotes the percentage of instances classified e.g. as *ham* as indeed being *ham*, but says nothing about the number of *ham* that was not labelled as such. *Recall* denotes the percentage of instances of e.g. *ham* being classified as such, but says nothing about how many other instances (e.g. *spam*) were incorrectly also classified as *ham*. *Support* stands for the number of instances of each class we have in the testing set.

Class	Precision	Recall	Support
Ham	98.54%	99.38%	817
Spam	98.70%	96.95%	393
Avg/Tot	98.58%	98.58%	1210

Table 2: Classification report for Linear SVM with C=1

Figure 2 illustrates that the more samples the applied SVM algorithm can train on, the better its accuracy is for the hold-out set, and that one needs at least a few hundred samples to get good results. The model is neither underfitting the training set since the training score is very close to the maximum (1.00), nor is it overfitting, since training score and test score are quite close with sufficient training set size.

Fig. 2: Learning curves for SVM with linear kernel and C=1

Conclusion

With two methods applied to the binary text classification problem that is spam filtering, the SVM proved to be far more effective than the Naïve Bayes algorithm in stopping unsolicited emails. Showing an almost flawless accuracy of 98.6%, the empirical experiment clearly proves the advantages of the Support Vector Machine, which are its speed, robustness, and efficiency. We conclude that SVMs are more than suitable to build state of the art, powerful email filtering systems with.

References

[1] R. Garetta, G. Moncecchi. *Learning scikit-learn: Machine Learning in Python*, 2013.

[2] H. Friedman, R. Tibshirani, T. Hastie. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2001.

[3] G. Chechik, G. Heitz *Max-margin Classification of Data with Absent Futures*. In Journal of Machine Learning Research 9, 2008.

[4] T. Mitchell *Generative and Discriminative Classifiers: Naïve Bayes and Logistic Regression*. In T. Mitchell, M. Hill. Machine Learning, 2015.