Detecting Spam Words Using Eigenspaces and Support Vector Machines

Jeremy Buchmann, George Bebis

December 16, 2002

Abstract

Unsolicited Bulk Email, commonly called Spam, has become a daily problem for email users. It is not only annoying and sometimes offensive, it also wastes time and bandwidth. As word-based spam filters have become better and more prevalent, spammers have adjusted their tactics by embedding the text of an email into an image or multiple images, and then sending those images as the entire content of the email. This leaves word-based filters without enough information to make a classification. In this paper, we will demonstrate how Eigenspaces and Support Vector Machines can be used to recognize certain keywords in images that would provide a strong indication that the image is spam-like.

1 Introduction

1.1 Motivation

Recognizing words in images is an important part of filtering spam. A significant amount of spam is now partially composed of images, and some is composed solely of images. In order to filter this type of spam, we need to understand the characteristics of the images, and determine how they are different from non-spam images.

1.2 Background

Turk and Pentland's paper [4] on using Eigenspaces for facial recognition describes how Eigenspaces are created. For more information on Principle Component Analysis (PCA), Appendix 3 in [2] has an example of the calculation of the eigenvectors and the reconstruction of the original data.

Support Vector Machines (SVMs) have become popular recently for various classification tasks. They are well known for their ability to handle high-dimensional spaces and their excellent classification ability. Burges's tutorial [1] is a good, in-depth starting point for understanding SVMs.

To our knowledge, this is the first attempt at recognizing words in images using this technique.

1.3 Short Description of Methodology

We first selected five keywords commonly found in spam images. Then, we built Eigenspaces for each set of words. Next, we projected each training image into all five Eigenspaces and concatenated the coefficients. Finally, the coefficients were used to train an SVM. The test images were projected into the same five Eigenspaces, concatenated together, and then classified by the SVM.

1.4 Summary of Results

Classification results for the test set were very good. Overall, the classifier correctly classified over 93% of the test images. The classification rates for each individual word set varied from 80% to 95%.

1.5 Paper Outline

Section 2 describes how we applied PCA and an SVM to the problem of word detection in an image. Section 3 presents the results we achieved and offers an analysis of the results, including why certain word sets did better than others. Section 4 concludes the paper and discusses future work that can be done on this technique.

2 Methodology

2.1 Defining a Spam Image

First, we need to define a spam image and determine how it is different from a non-spam image. A simple but un-helpful definition of a spam image is "an image that is associated with spam". This tells us nothing about the characteristics of a spam image, so we must examine spam images to determine what characteristics separate them from non-spam images.

The most obvious difference is the use of text embedded in the image. Spammers need to get a message across with text; there's no way to avoid this fact. The existence of text in an image is a strong indicator that the image is a spam image. However, more certainty can be gained by knowing what the words in the image are.

We wanted our technique to be relatively lightweight, so we decided not to use optical character recognition which can be very complex. After all, we don't need to know every word in the image; we only need to know if certain keywords exist. It dawned on us that we don't need to understand the meaning of the words in the image, we just need to be able to match their general shape to the shape of one of the keywords we're looking for. To get the general shape of a word, we decided to use PCA.

2.2 PCA

PCA is a mathematical technique that is able to capture an "impression" of data. PCA can greatly reduce the dimensionality of a data set and still preserve the most important features. PCA is a purely algebraic technique, and is relatively easy to compute. As show in [4], PCA can capture enough of an impression to achieve excellent results in facial recognition on minimal hardware.

PCA computes a linear transformation that maps high dimensional data into a lower dimensional space.

The methodology behind PCA is as follows:

- Suppose $x_1, x_2, \ldots x_M$ are N x 1 vectors
- Find the mean vector: $\bar{x} = \frac{1}{M} \sum x_i$
- Subtract the mean: $\Phi_i = x_i - \bar{x}$
- Form the matrix: $A = [\Phi_1, \Phi_2, \ldots, \Phi_M]$

- Compute the covariance matrix: $C = AA^T$
- Compute the eigenvalues of C: $\lambda_1 > \lambda_2 > \cdots > \lambda_N$
- Compute the eigenvectors of C: $u_1, u_2, \ldots, u_N$
- Since C is symmetric, its eigenvectors form a basis.
- The linear transformation that performs the dimensionality reduction (K $\ll$ N) is: $\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix} = \begin{bmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_K^T \end{bmatrix} (x - \bar{x})$

2.3 SVM

A full treatment of SVM is beyond the scope of this work; however the main idea behind it is to define a decision boundary between the different classes. The decision boundary is constructed from the data itself. The data points that lie along the decision boundary are called "support vectors".

There are various types of discriminant functions including linear, quadratic, polynomial, and generalized, which can approximate any discriminant function.

In order to avoid the calculation of high-dimensional dot products, a "kernel" function $K(x, x_k) = \Phi(x) \cdot \Phi(x_k)$ can be used to achieve the same result.

For more information on SVM, please refer to [1].

2.4 Image Collection

We started by collecting as many spam images as we could and kept notes on what words were contained in the images. After we had collected a number of images, we counted the occurrences of all the words and came up with a list of the most common. Several of the top words, such as "to", "your", "the", and "and" were ambiguous, so we ignored them. From the remaining top words, we chose "free", "click", "here", "now", and "$" (the dollar sign) to be used for classification.

Next, we cropped and normalized the images containing the words so that the words filled the entire space in the image. The images were then converted to grayscale. Unfortunately, we still didn't have enough images with which to train a classifier, so we created ten to fifteen images of each word in different fonts and using various degrees of capitalization to supplement the word images that came from spam.

After dividing the images into training and testing sets, we computed Eigenspaces for each set of training words. We kept the top ten principle components for each set.

Then, for each image in the training sets, we projected the image into all five Eigenspaces and concatenated the coefficients to form a single, wide feature vector. All of the training feature vectors were then used to train the SVM. Figure 1 illustrates the process of training the SVM.

To test the system, each image in the testing set was projected into all five Eigenspaces and the coefficients were concatenated to form a feature vector that was then given to the SVM for classification. The SVM returns either a "yes" or "no" answer.

The main drawback to this system is that each test image must be properly normalized (resized to the same dimensions as the training images) and registered (the projected image must tightly enclose the keyword and not

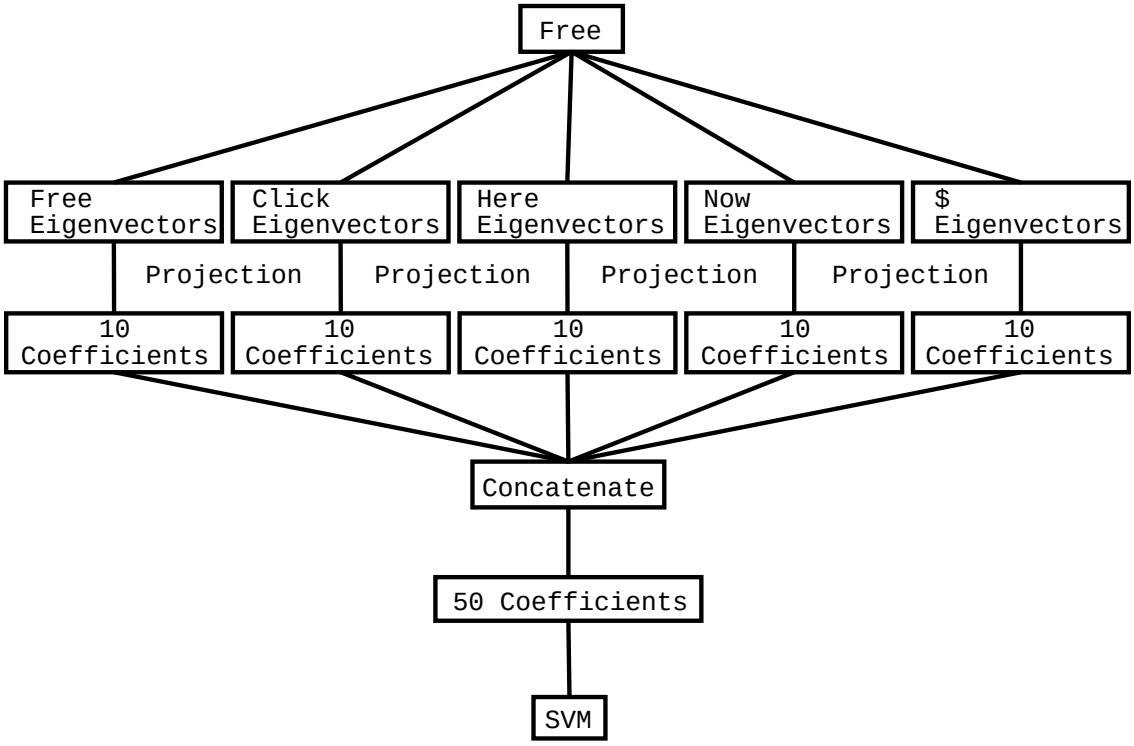

Figure 1: The process of training the SVM. The image "Free" is projected into each Eigenspace. Then the projections are concatenated and fed to the SVM as a single feature vector.

contain any extraneous data). There are a few ways to get around this, however. A text location technique such as [3] or [5] may solve the problem of trying to register the keyword. Once a word has been located, it can be resized to the same dimensions as the training images, even though the resizing may distort the image. The system seems to be able to handle small amounts of distortion without any problem.

3 Results and Analysis

The following table shows the results obtained on the test set:

Word	Correct	Incorrect	Total	% Correct
Free	36	3	39	92.3%
Click	24	6	30	80.0%
Here	24	4	28	85.7%
Now	37	2	39	94.9%
$ (Dollar)	16	1	17	94.1%
Non-text	338	16	354	95.5%
Total	475	32	507	93.7%

As you can see, the classifier[1] does an excellent overall job of classifying the test set images. The "Correct" column indicates how many testing images were correctly classified and the "Incorrect" column indicates how many were incorrectly classified. The "Total" column shows how many testing images were used and the "% Correct" column shows the correct classification rate.

The "Non-text" word set is a set of images that do not contain any of the target words. Some contain other text, and some are composed from images and drawings. This set is by far the largest and the best classified. Although surprising at first glance, the reason for it's excellent classification probably stems from the fact that it didn't have an Eigenspace generated for it; it was only projected onto the Eigenspaces for the other words. The accuracy of the Non-text set skews the overall accuracy upwards, but this is acceptable because we want the spam filter to be able skip sections of images that don't contain keywords.

The words "click" and "here" had a noticeably lower classification percentage than the other words. Upon examination, we noticed that the most of the misclassified images contained words that were entirely capitalized. In future experiments, it would be prudent to ensure that there are many examples of both capitalized and uncapitalized words.

4 Conclusions and Future Work

PCA and SVM have been shown to be effective in classifying properly normalized images that contain keywords. This technique has a lot of potential in the area of spam filtering and other types of image classification and retrieval.

There are many ways this work can be extended and improved. The next step might be to create a program that can classify a non-normalized image that contains one of the keywords. This program would have to locate the text in the image and properly normalize and register it.

Further tweaking of the SVM may be worthwhile, as time only permitted the tuning of approximately half of the available options in the SVM.

[1] The classifier was configured as a C-SVC with a Radial Basis Function kernel, a gamma value of 0.5, and a cost of 100,000.

LDA could also be used in place of PCA to more effectively discriminate between the word classes. This technique may make more sense with each word given its own class instead of the two-class classification described in this paper.

References

[1] Burges, C. (1998) A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery, 2(2): 955-974.

[2] Castleman, K. (1995) *Digital Image Processing.* Prentice Hall: 647-649

[3] Nath, P. (2002) *Extraction of Text from Images.* `http://www.cse.iitk.ac.in/research/btp2002/98263.ps.gz`

[4] Turk, M. and Pentland, A. (1991) *Eigenfaces for Recognition.* Journal of Neuroscience, 3(1): 71–86.

[5] Wu, V., Manmatha R., and Riseman, E. (1999) *TextFinder: An Automatic System to Detect and Recognize Text In Images.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(11).