Partitioned Logistic Regression for Spam Filtering

Ming-wei Chang
University of Illinois
201 N Goodwin Ave
Urbana, IL, USA
mchang21@uiuc.edu

Wen-tau Yih
Microsoft Research
One Microsoft Way
Redmond, WA, USA
scottyih@microsoft.com

Christopher Meek
Microsoft Research
One Microsoft Way
Redmond, WA, USA
meek@microsoft.com

ABSTRACT

Naive Bayes and logistic regression perform well in different regimes. While the former is a very simple generative model which is efficient to train and performs well empirically in many applications, the latter is a discriminative model which often achieves better accuracy and can be shown to outperform naive Bayes asymptotically. In this paper, we propose a novel hybrid model, *partitioned logistic regression*, which has several advantages over both naive Bayes and logistic regression. This model separates the original feature space into several disjoint feature groups. Individual models on these groups of features are learned using logistic regression and their predictions are combined using the naive Bayes principle to produce a robust final estimation. We show that our model is better both theoretically and empirically. In addition, when applying it in a practical application, email spam filtering, it improves the normalized AUC score at 10% false-positive rate by 28.8% and 23.6% compared to naive Bayes and logistic regression, when using the exact same training examples.

Keywords

logistic regression, naive Bayes, email spam filtering

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; H.4.3 [**Communications Applications**]: Electronic mail

1. INTRODUCTION

Due to having the same linear functional form when used as a classifier, naive Bayes and logistic regression have been called a generative–discriminative pair of classifiers. The former approach learns a model that estimates the joint probability, $P(\mathbf{X}, Y)$, where Y is the class label and $\mathbf{X}$ is the example feature vector. The posterior probability of the class, which is estimated by applying Bayes rule, is used to predict the most likely label. On the other hand, the latter approach tries to learn a model to directly estimate the posterior, $P(Y|\mathbf{X})$. In many applications, discriminatively trained models outperform their generatively trained counterparts (e.g., [25, 19]). This empirical result is also supported from the perspective of computational learning theory [32]. Nevertheless, a generatively trained classifier, such as naive Bayes, still enjoys several advantages in practice. For example, the learning procedure is usually very simple and efficient and several techniques can improve its performance and make it very competitive when compared with logistic regression (e.g., [27, 19]). When the size of the training data is small, naive Bayes can even outperform logistic regression, regardless of whether its strong conditional independence assumption holds or not. This is due to the fact that naive Bayes can converge to its asymptotic accuracy (i.e., the model trained on infinitely number of examples) much faster than logistic regression [24].

In this paper, we introduce a novel learning method, *partitioned logistic regression* (PLR), which is a hybrid model that can be viewed as part generative and part discriminative. Given training examples, the PLR approach first separates the feature space into k disjoint subsets of features, which are assumed conditionally independent given the class label. Correspondingly, k different models are trained discriminatively using logistic regression on these subsets of features only. The final prediction is based on the multiplication of the posterior probabilities estimated by these k individual models, following the conditional independence assumption assumed by the naive Bayes model.

Because the PLR model only assumes that the features of different groups are conditionally independent, but not features within the same group, it outperforms naive Bayes in our experiments on both synthetic and real datasets. Perhaps more surprisingly, we also found that the PLR model outperforms its discriminative counterpart, logistic regression, even when the conditional independence assumption does not hold. We provide theoretical evidence that suggests why this can happen and provide a set of experiments on synthetic data to illustrate the effectiveness of the PLR model. Furthermore, we demonstrate that, when applied to a large real dataset from the email spam filtering domain, the PLR model significantly outperforms both logistic regression and naive Bayes.

Improved prediction accuracy is not the only advantage of the PLR model. The model is especially suitable for real-world applications. The linear functional form of partitioned logistic regression is fast and easy to implement. Many production systems use models of this form, which means that a PLR model could potentially be easily adapted for use in these systems. In fact, typically the only change needed is to replace the weights with those learned using partitioned logistic regression. Having the features separated into several groups enables one to easily tune smoothing parameters for different groups of features. As we will demonstrate, this can lead to further significant performance enhancements. Another advantage of partitioned logistic regression is its ability to incorporate multiple information sources together. This property is especially important when we apply the PLR model to the task of spam filtering. Although traditionally spam filtering is treated as a special problem of text classification, using only the email content information often provides unsatisfactory results. This is mainly due to the fact that spam is an adversarial problem and spammers can easily manipulate messages by adding "good words" to fool the content-based spam filter [22, 21]. In this paper, we also demonstrate how the PLR model can easily incorporate information such as sender reputation and user preference in a highly scalable way to enhance the performance of a spam filter.

The rest of the paper is organized as follows. We first describe our partitioned logistic regression model in Sec. 2. Theoretical analysis and experiments on synthetical data are presented in Sec. 3 and 4 respectively, which explains why the PLR model can perform better than both naive Bayes and logistic regression. We then depict how we use our model in the task of email spam filtering, with the focus on how it combines several sources of important information, and show the experiments in Sec. 5 and 6. In addition, methods of modeling user preference given only partial information are investigated in Sec. 7. Finally, we describe some related work in Sec. 8 and conclude the paper in Sec. 9.

2. PARTITIONED LOGISTIC REGRESSION

Conceptually, our *partitioned logistic regression* (PLR) model can be defined as a set of classifiers that are trained by logistic regression using the same examples, but on different *partitions* of the feature space. Given a testing example, posterior probabilities are first estimated by these classifiers and the product of these values are then used to make the final prediction. Although it can be easily extended to multi-class problems, to simplify the presentation, we focus on the binary case. Formally, let $Y \in \{0, 1\}$ be the label of an example $\mathbf{X} \in \mathcal{R}^n$ that consists of n features. Assume the feature space can be separated into k disjoint sets according to a predefined partitioning. Namely, $\mathbf{X} = (x^1_1, \cdots, x^1_{n_1}, x^2_1, \cdots, x^2_{n_2}, \cdots, x^k_1, \cdots, x^k_{n_k})$, where for the j-th group, x^j_i is its i-th feature and n_j is the number of features in this group. The total number of features in this example is thus the same as the sum of the numbers of features in these groups. That is, $n = \sum_{j=1}^k n_j$. Let $\mathbf{X}_j$ be the sub-vector that consists of the j-th group of features. The original example can be represented as $\mathbf{X} = \mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_k$. During training, k linear models are learned using logistic regression on the same training examples, but each model learns on its corresponding set of features. We denote the weights and the bias term of the j-th model as $\mathbf{W}_j$ and b_j, and the posterior estimated by this individual model is:

$$\hat{P}(Y = 1|\mathbf{X}_j) = \frac{\exp(\mathbf{W}_j \cdot \mathbf{X}_j + b_j)}{1 + \exp(\mathbf{W}_j \cdot \mathbf{X}_j + b_j)}$$

Let $\hat{o}_j = \hat{P}(Y = 1|\mathbf{X}_j)/\hat{P}(Y = 0|\mathbf{X}_j)$ be the posterior odds estimated by the j-th model, $\hat{o} = \hat{P}(Y = 1)/\hat{P}(Y = 0)$ the estimated prior odds, which can be derived using the empirical class distribution in the training data. Given a testing example $\mathbf{X}$, the estimated posterior odds is defined as:

$$\frac{\hat{P}(Y = 1|\mathbf{X})}{\hat{P}(Y = 0|\mathbf{X})} \equiv \hat{o}^{(1-k)} \cdot \prod_{j=1}^k \hat{o}_j \qquad (1)$$

The label of this example is predicted as 1 when the value of Eq. 1 is larger than 1, and 0 otherwise. When this function is used for ranking, the term $\hat{o}^{(1-k)}$ can be ignored since it is a constant that does not change as the testing example changes.

Although it may not seem obvious, the PLR model can be viewed as a set of individual logistic regression models, combined following the naive Bayes principle. In other words, the groups of features are assumed conditionally independent given the class label, and can be described as Eq. 2.

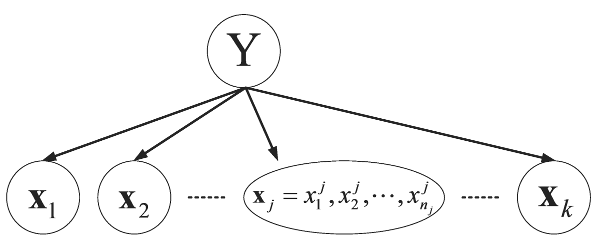

Figure 1: Features of different groups $(\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_k)$ are assumed conditionally independent given the class label Y

$$P(\mathbf{X}|Y) = P(\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_k|Y) = \prod_{j=1}^k P(\mathbf{X}_j|Y) \qquad (2)$$

When the estimated odds and posteriors are correct, it can be shown that Eq. 1 is indeed the true posterior odds when this assumption holds. Let o be $\frac{P(Y=1)}{P(Y=0)}$ and o_j be $\frac{P(Y=1|\mathbf{X}_j)}{P(Y=0|\mathbf{X}_j)}$.

$$\begin{aligned}\frac{P(Y = 1|\mathbf{X})}{P(Y = 0|\mathbf{X})} &= \frac{P(Y = 1)P(\mathbf{X}|Y = 1)}{P(Y = 0)P(\mathbf{X}|Y = 0)} \\ &= o \cdot \frac{\prod_{j=1}^k P(\mathbf{X}_j|Y = 1)}{\prod_{j=1}^k P(\mathbf{X}_j|Y = 0)} \\ &= o \cdot \prod_{j=1}^k \frac{P(Y = 0)}{P(Y = 1)} \cdot \frac{P(Y = 1|\mathbf{X}_j)P(\mathbf{X}_j)}{P(Y = 0|\mathbf{X}_j)P(\mathbf{X}_j)} \\ &= o^{(1-k)} \cdot \prod_{j=1}^k o_j \qquad (3)\end{aligned}$$

Note that Eq. 2 is different from the conditional independence assumption in the naive Bayes classifier. Here only features of different groups are conditional independent, but not the features within the same group. The relationships among these features can be represented by Fig. 1.

Although the PLR model consists of k logistic regression models, in practice, it is very easy to replace a regular logistic regression model trained on all features with the PLR model, due to their identical functional form. This property is especially clear through the log-odds function, which is the weighted sum in logistic regression.

$$\begin{aligned}\hat{lo}(\mathbf{X}) &= \log\left(\hat{o}^{(1-k)} \cdot \prod_{j=1}^k \hat{o}_j\right) \\ &= (1-k)\log\hat{o} + \sum_{j=1}^k \log\hat{o}_j \\ &= (1-k)\log\hat{o} + \sum_{j=1}^k (\mathbf{W}_j \cdot \mathbf{X}_j + b_j) \\ &= \left(\sum_{j=1}^k \mathbf{W}_j \cdot \mathbf{X}_j\right) + \left((1-k)\log\hat{o} + \sum_{j=1}^k b_j\right) (4)\end{aligned}$$

In other words, the weights learned by k different models can be used directly as the weights of a global logistic function, and the new bias term is $(1 - k)\log\hat{o}$, plus the sum of all individual bias terms.

The set of partitioned logistic regression models represents a spectrum of models with naive Bayes at one end and logistic regression at the other. For example, when $k = 1$, the PLR model reduces to the regular logistic regression model. On the other hand, when $k = n$, the model is equivalent to naive Bayes modulo smoothing; each component logistic regression model learns the empirical posterior probability for one feature.

The PLR model enjoys several advantages over both naive Bayes and logistic regression. Although naive Bayes is a very efficient learning algorithm and has shown good performance in many applications, its main weakness is the strong independence assumption that seldom holds in reality. In contrast, the PLR model provides a principled way to control the degree of feature dependence. Features within the same group are not assumed conditionally independent given the class label and can be modeled better by logistic regression. More interestingly, the PLR model can also be better than the regular logistic regression model. It has been argued that with enough examples, a discriminative model such as logistic regression can achieve equal or better performance than its generative counterpart (i.e., naive Bayes) asymptotically [24]. However, as we will show in Sec. 3, under some circumstances partitioned logistic regression can in fact learn a better model with fewer training examples. Partitioned logistic regression also has an additional advantage that is crucial in practice – allowing tuning the smoothing priors more easily. As will be demonstrated in Sec. 6, the PLR model with the same prior for all the weights already outperforms the regular linear regression model, but by allowing different priors for individual logistic regression models, the performance improvement can be further increased.

3. THEORETIC ANALYSIS

When the conditional independence assumption holds, following the same argument used in [9], one can show that asymptotically (i.e., with infinitely many training examples) the PLR model performs no worse than the LR model. However, when this assumption does not hold, a discriminative classifier outperforms its generative counterpart asymptotically, as long as its hypothesis space has finite VC dimension [32]. Intuitively, this is because the true discriminate learning algorithm has the freedom to search the whole weight space while the generative model limits the possible weights due to some assumption about the data. Although partitioned logistic regression has a weaker conditional independence assumption as compared to naive Bayes, it does indeed limit its flexibility to learn the model parameters.

Inspired by [24], we argue that even though PLR may learn a worse model asymptotically, it converges faster than logistic regression as the number of training examples increases. We first present a straightforward result of applying Vapnik's uniform convergence bounds [32] to logistic regression.

Theorem 1 ([24]) *Let h_{Dis} be logistic regression learning an n-dimensional weight vector. Then with high probability*

$$\varepsilon(h_{Dis}) \leq \varepsilon(h_{Dis,\infty}) + O\Big(\sqrt{\frac{n}{m}\log\frac{m}{n}}\Big),$$

where m is the number of training examples, $\varepsilon(h_{Dis})$ is the error of the learned model and $\varepsilon(h_{Dis,\infty})$ is the asymptotic error.

Theorem 1 states that the sample complexity of discriminative learning (the number of examples needed to approach the asymptotic error) is at most on the order of n, which is the VC dimension of the logistic regression model. It also implies that when the weight space of the logistic regression learner is smaller, it requires fewer examples to approach its asymptotic error. Consider a component logistic regression learner and a global logistic regression learner. The former sees only the n_j features of each example and learns $n_j + 1$ weights, while the latter is given the whole n features and learns the $n + 1$ weights. Suppose the number of features used in each component logistic regression model is the same. With k components, $n_j = n/k$, which indicates each local model only needs $O(n/k)$ examples to converge, while the regular logistic regression needs $O(n)$ examples.

In practice, the size of the training data is often insufficient for the learner to achieve the optimal error rate, especially when the feature space is large. Being able to converge faster explains why the PLR model may still outperform regular logistic regression, even when the conditional independence assumption (Eq. 2) does not hold.

4. EXPERIMENTS ON ARTIFICIAL DATA

In order to demonstrate the effectiveness of partitioned logistic regression, we show empirically that the PLR model can outperform regular logistic regression (LR), using several artificial datasets. In particular, we verify two claims in this section. First, if the examples are generated from a distribution where the conditional independence assumption (Eq. 2) holds, PLR indeed outperforms LR. Second, even when the conditional independence assumption does not hold, PLR can still be better than LR, especially when the number of training examples is small.

4.1 Data Generation

Let $Y \in \{0, 1\}$ be the class label of an example $\mathbf{X}$, which consists of two groups of d binary features $\mathbf{X}_1$ and $\mathbf{X}_2$. We generate this labeled example using the following procedure:

$$\begin{aligned} Y &\sim \text{Bernoulli}(0.5) \\ \hat{\mathbf{X}} = (\hat{\mathbf{X}}_1, \hat{\mathbf{X}}_2) &\sim N(u_y, \Sigma_y) \\ \mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2) &= (\varphi(\hat{\mathbf{X}}_1), \varphi(\hat{\mathbf{X}}_2)), \end{aligned} \quad (5)$$

where $N(u_y, \Sigma_y)$ is a multivariate normal distribution for a given Y^*. Vector $\hat{\mathbf{X}}$ is a real-valued vector which can be decomposed to $(\hat{\mathbf{X}}_1, \hat{\mathbf{X}}_2)$. $\varphi(.)$ is a deterministic function that converts a real-valued vector $\mathbf{R} = (R_1, R_2, \cdots, R_d)$ to a binary vector. Each element R_j is first transferred to the corresponding base-2 representation. The sign bit and the bits that correspond to $2^2, 2^1, 2^0, 2^{-1}, 2^{-2}$ form the binary sub-vector for this element. Therefore, the length of the final binary vector is $6d$.

We generate the covariance matrix Σ_y, which needs to be symmetric and positive semi-definite, in the following way. A $2d \times 2d$ Gram matrix is first created using $2d$ randomly generated vectors†, which can be re-written as:

$$M = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix},$$

where A, B and C are $d \times d$ matrices. We then introduce a parameter α to create a different matrix M' and use it as the covariance matrix Σ_y, where

$$M' = \begin{bmatrix} A & \alpha B \\ \alpha B^T & C \end{bmatrix}. \quad (6)$$

*To simplify the notation, we assume the multivariate normal distribution generates row vectors directly. Let $\mathbf{e}$ be a vector of d ones. We set $u_y = \mathbf{e}\sqrt{0.5/d}$ when $y = 1$ and $-\mathbf{e}\sqrt{0.5/d}$ otherwise.

†The Gram matrix is all possible inner products of the given vectors, and is therefore symmetric and positive semi-definite.

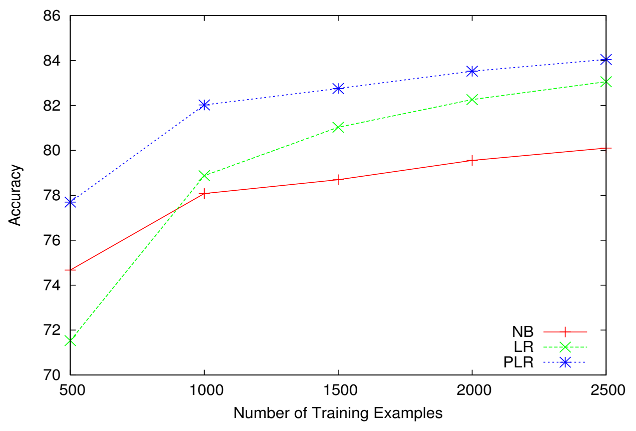

Figure 2: The performance of three methods in accuracy using different numbers of training examples. The two feature groups are conditionally independent in this synthetic data.

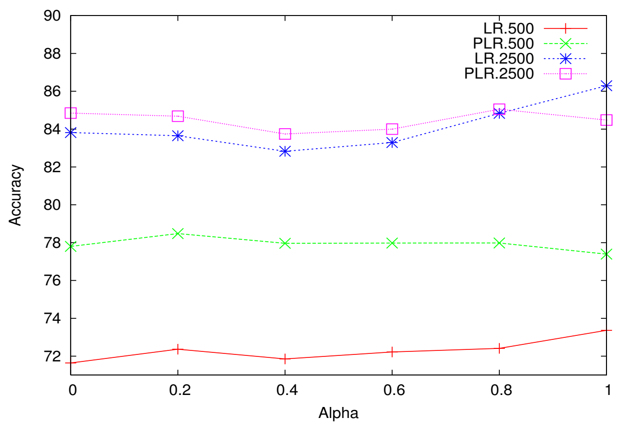

Figure 3: The performance of LR and PLR using different sizes of training data. Whether the groups of features are conditionally independent is controlled by the parameter α.

Note that M' is also positive semi-definite when $0 \leq \alpha \leq 1$, and therefore is a valid covariance matrix.

Notice that our covariance matrix construction is equivalent to setting $cov(\hat{\mathbf{X}}_1, \hat{\mathbf{X}}_1|Y)$, $cov(\hat{\mathbf{X}}_1, \hat{\mathbf{X}}_2|Y)$ and $cov(\hat{\mathbf{X}}_2, \hat{\mathbf{X}}_2|Y)$ as A, αB and C, respectively. Therefore, we can use the parameter α to control the "degree" of independence of $\mathbf{X}_1$ and $\mathbf{X}_2$ given Y. In particular, they are conditionally independent when $\alpha = 0$, but not when $\alpha > 0$. When $\alpha = 1$, they are closely related.

By fixing d to 20 (i.e., 240 binary features) and varying the parameter α, we generate several synthetic datasets for experiments. Each dataset is further split for training and testing. While the number of training examples may vary, we fix the number of testing examples to 2,500. The averaged accuracy of 100 rounds is reported for each configuration.

4.2 Results

We first verify the case when the conditional independence assumption holds. By setting α to zero, feature groups $\mathbf{X}_1$ and $\mathbf{X}_2$ are conditionally independent given Y. However, features in the same group (either $\mathbf{X}_1$ or $\mathbf{X}_2$) are not independent since the covariance matrices that decide $\mathbf{X}_1$ and $\mathbf{X}_2$ are not diagonal matrices (see Eq. 6). Fig. 2 compares the performance in terms of accuracy of three different methods, naive Bayes (NB), logistic regression (LR) and partitioned logistic regression (PLR), when given different sizes of training data. As shown in the figure, when the number of training examples is small (e.g., 500), NB is better than LR. However, LR quickly outperforms NB when there is more training data and the performance gap increases as the size of training data grows. In contrast, PLR is better than both NB and LR in all these experiments, which is consistent to what we expected. Notice that all the comparisons here are statistically significant[‡].

It is interesting to know whether PLR can still perform well when the groups of features are not conditionally independent. For this set of experiments, we vary the parameter α from 0 to 1 and generate two different sets of data. One of them contains 500 training examples and the other has 2,500 training examples. Fig. 3 shows the results of LR and PLR. Although it is expected that PLR is better than LR when $\alpha = 0$ (i.e., the conditional independence assumption holds), it is somewhat surprising to see that PLR still outperforms LR in many cases, especially when number of the training examples is small. This implies that LR may need more training examples to achieve its asymptotic performance compared to PLR. Note that all the comparisons are statistically significant except the difference between LR and PLR when $\alpha = 0.8$ and the number of training examples is 2,500.

[‡]We conduct a paired-t test when comparing two learning methods using the same training and testing data. The difference is considered statistically significant when the p-value is less than 0.05.

5. EMAIL SPAM FILTERING

Although we have shown theoretically and experimentally (on synthetic datasets) that partitioned logistic regression can perform better than both logistic regression and naive Bayes, it is more important to examine how effective this model is for a real-world application, such as email spam filtering.

Spam filtering is a problem of classifying incoming messages to either spam or good, and has usually been treated as a binary text classification problem. The approach of building a spam filter as a classifier trained by machine learning algorithms (e.g., naive Bayes) using words or tokens in email as features, has been advocated for a decade [28]. Although various feature engineering and sophisticated learning algorithms have been proposed (e.g., [33, 19, 14]), the view that a spam filter is a special text classifier has not been changed much, and a content-based filter is still the core component in many commercial email spam filtering systems.

Unfortunately, due to the adversarial nature of the spam problem, a content-based filter is usually quite vulnerable and needs to be retrained frequently to defend new spam attacks. By adding words that are often seen in normal messages, spammers can manipulate the content of an spam message and fool a content-based filter easily [22, 21]. Because of this weakness, researchers have started to treat this problem as a more general classification problem and explored other types of information instead of just the content. One recent example is the use of sender information [20]. Intuitively, if an IP address never sent spam messages in the past, the sender behind this IP is unlikely to be a spammer. In contrast, messages sent from IPs of bad reputation can be predicted as spam with high accuracy, even without checking the content of the messages. Compared to the content features, the IP address of the email sender is harder to spoof, which makes it a more reliable information source.

In addition to sender reputation, the recipient information can also help spam filtering. It is often assumed (at least implicitly) that there is a clear distinction on whether an email message is spam or good to a human subject. In practice, however, this assumption does not always hold. For example, when receiving unsolicited commercial email, two-thirds of users consider it as a good message as long as they have done business with the sender [11]. Messages belonging to this category are called "gray mail", which could reasonably be considered either spam or good [34]. The label of this type of mail depends on the individual user preference. After all, even the exact same message can be labeled differently when it is sent to different users. For a spam filter to classify it accurately, the user preference needs to be considered.

To build a robust spam filter, it is thus preferable to incorporate all the aforementioned information: *content*, *sender reputation* and *user preference*. While the natural choice is to extract features from each of these three information sources and build a classifier that learns on all features, as we will show in Sec. 6, a PLR model that treats these groups of features separately in fact performs better. In practice, a spam filter trained using labeled messages from one group of users often needs to be applied to messages sent to other users as well. With limited feedback from previously unseen users, how to adjust the filter to handle their user preferences is not trivial. In Sec. 7, we will further investigate this issue and demonstrate that the PLR model can easily incorporate partial information of user preference and still improve the filter.

6. EXPERIMENTS

We compare the PLR model with logistic regression and naive Bayes experimentally on the application of email spam filtering in this section. We first describe the basic setting of our experiments, followed by the results of various configurations in different evaluation metrics.

6.1 Experimental Setting

To compare the PLR model with other models, we conduct several experiments on three email datasets for spam filtering. The first one is a non-public Hotmail dataset that has been used previously [33, 19]. The other two are the 2005 and 2006 TREC Spam Filtering Track datasets [6, 5], which have also been used in previous works (e.g., [19]).

The Hotmail corpus is collected by polling Hotmail volunteers daily. The system randomly picks messages sent to voluntary users and asks them to label the messages as *good* or *spam*. Because the labels are given by the recipients, identical messages that are sent to different users may be labeled differently, according to their own preferences. Therefore, knowing the recipient of the email message can potentially increase the performance of the spam filter.

We use the same Hotmail collection as used in [33] for experiments. The training set contains 765,000 messages received between July-01-2005 and Nov-30-2005. The remaining messages are split into validation and testing sets. The former consists of 30,000 messages received between Dec-01-2005 and Dec-03-2005; the latter has 120,000 messages received between Dec-04-2005 and Dec-15-2005. The features used in our system are as follows. The *content* features are composed of the words in the subject and body that have occurred at least three times in the training set – whether a word occurs in the message is used as a binary feature. The number of content features is therefore decided by the size of the vocabulary. The *sender* features are determined by the the IP address of the sender, which include the first 16 bits, the first 24 bits and the whole IP address. Finally, the *user* features are the recipient ids.

Other than the Hotmail data, we also use the TREC corpora, which consists of chronologically ordered email messages. Despite the fact that the TREC corpora are arguably the largest publicly available email datasets for spam filtering, their sizes are still much smaller compared to the Hotmail collection. Following a similar setting in [19], for the TREC-05 corpus, we use the first 30,726 messages for training, the subsequent 10,242 messages for validation, and the remaining 51,213 messages for testing. For the TREC-06 English corpus, the initial portion of 12,606 messages are used for training. The first 4,202 messages in the remaining portion are used for validation and the rest for testing. Although the messages in these corpora are real email, the way they are collected is somewhat artificial. Because private messages cannot be included, the TREC-05 corpus is a combination of Enron corpus and SpamAssassin corpus, and the messages in TREC-06 corpus are email crawled from the Web. In both corpora, the message labels are not given by the original mail recipients, but instead given by human annotators and some spam filters. Therefore, the user preference information is not preserved. As a result, for the experiments on the TREC corpora, only content and sender features are used and they are extracted in the same way as in the Hotmail dataset.

We show the results of four different learning methods on these three datasets. They are naive Bayes (NB), regular logistic regression (LR), partitioned logistic regression with the same smoothing prior for all feature groups (PLR) and partitioned logistic regression with different smoothing priors for different groups of features (PLR+)[§]. For naive Bayes, we use the regular multivariate Bernoulli model with Laplace smoothing (add-one smoothing). For the logistic regression components in LR, PLR and PLR+, we used SCGIS [12] as the actual training method. However, because the solution space is convex and has a global optimum, the choice of training algorithm makes relatively little difference. On the other hand, setting the right smoothing parameter (i.e., the variance of the Gaussian prior) is crucial for good empirical performance. Therefore, for each set of the experiments, we tested it with values among $\{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30\}$ and selected the best one according to the results on the validation set. The results of these models on the testing set are reported.

Although spam filtering is a binary classification problem, the accuracy of the filter at the 50-50 decision point is usually not a good performance metric[¶]. Because the cost of losing good mail (false-positive) is much higher than receiving spam (false-negative), we present the ROC curves of different models in the low false-positive region, which is a typical method for evaluating spam filters. In addition, we also report two measures that probe the quality of the filters in this region. The first is the normalized AUC score [19], which is essentially the area under the ROC curve in the low false-positive region. Assume t is the target false-positive rate (FPR). The normalized AUC (denoted as AUC_t) is the area in the section of the ROC curve, $\{(FPR, TPR) : 0 \leq FPR \leq t, 0 \leq TPR \leq 1\}$, divided by t. The second measure is the true-positive rate of the filter at a specific false-positive rate t (denoted as TPR@FPR=t), which is a point on the ROC curve. When comparing methods on TPR@FPR=t, we use McNemar's test [7] on the classification results of two compared methods at their decision thresholds that correspond to the false-positive rate t. The difference between two methods is considered statistically significant when the p-value of the test is less than 0.05.

6.2 Results

We first show the ROC curves of different methods on the Hot-

[§]Note that for PLR+, we use the validation set to select the best smoothing parameter for each component logistic regression model *independently*, instead of trying all the combinations of parameters.

[¶]The differences in accuracy between various models on the email datasets are in fact consistent with what we observed in the experiments on synthetic datasets (Sec. 4). However, since accuracy is not the right measure for spam filtering, we choose not to report it.

measure	$AUC_{0.1}$				TPR@FPR=0.1			
learning algorithms	NB	LR	PLR	PLR+	NB	LR	PLR	PLR+
content	0.459	0.512	-	-	0.651	0.722	-	-
content, user	0.466	0.474	0.516	0.555	0.658	0.704	0.723	0.753
content, sender	0.495	0.536	0.575	0.608	0.687	0.762	0.837	0.864
content, sender, user	0.500	0.521	0.588	0.644	0.691	0.766	0.825	0.873

Table 1: $AUC_{0.1}$ and TPR@FPR=0.1 of NB, LR, PLR and PLR+ when using different feature groups on the Hotmail dataset.

Dataset	measure	$AUC_{0.01}$				TPR@FPR=0.01			
	learning algorithms	NB	LR	PLR	PLR+	NB	LR	PLR	PLR+
TREC 05	content	0.520	0.843	-	-	0.626	0.919	-	-
	content, sender	0.536	0.850	0.878	0.878	0.643	0.948	0.951	0.962
TREC 06	content	0.399	0.759	-	-	0.506	0.928	-	-
	content, sender	0.406	0.785	0.858	0.875	0.514	0.928	0.944	0.943

Table 2: $AUC_{0.01}$ and TPR@FPR=0.01 of NB, LR, PLR and PLR+ when using different feature groups on the TREC datasets.

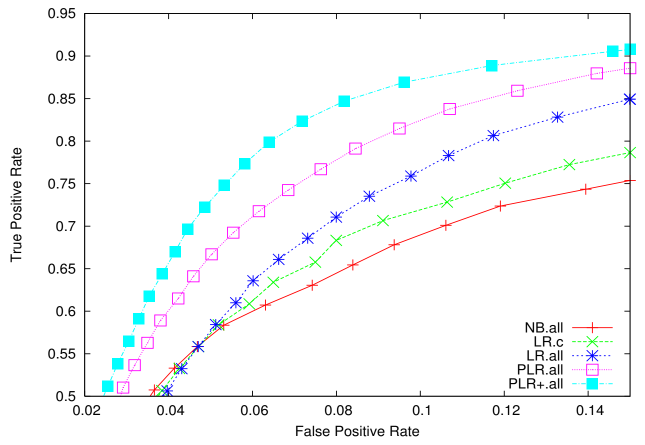

Figure 4: The ROC curves for the Hotmail dataset. Symbol *c* means only content features are used, and *all* means all three types of features are used.

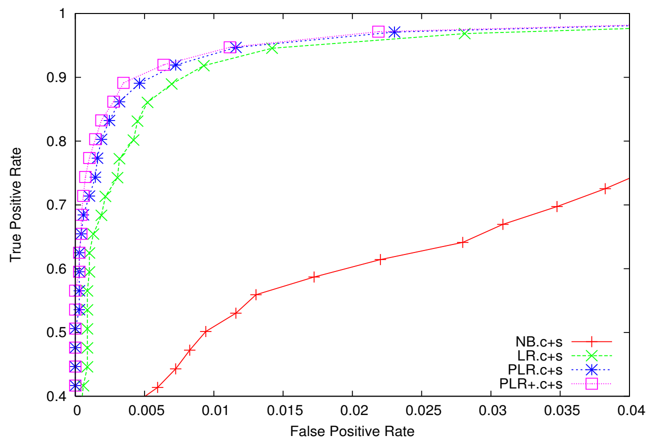

Figure 5: The ROC curves of four learning methods using both content and sender features on the TREC-06 datasets.

mail dataset in Fig. 4, focusing on the low false-positive region. To make the presentation clear, we only show the results when all the feature groups are used, as well as the result of a logistic regression model that is trained using content features only. As we can see in the figure, adding other non-content features does improve the filter. When comparing two logistic regression models, LR.c versus LR.all, the latter is consistently better in most of the region, and these two curves cross at around 0.05 false-positive rate. Learning on all the features together in a single logistic regression model, however, is not the best way to use non-content information. When comparing LR.all and PLR.all, the PLR model achieves 0.1 higher true-positive rate than the logistic regression model consistently throughout the region. More encouragingly, by finding the best smoothing parameter for each of the individual logistic regression components independently, the performance gap does increase quite substantially. This result can be clearly seen when we compare the curves of PLR+.all versus LR.all. Finally, naive Bayes seems to perform worse than logistic regression on this dataset. Even when trained on all features, it is still not as good as the logistic regression model trained on content features only.

We further investigated the impact of adding different groups of features in different learning methods. Tab. 1 shows the results in $AUC_{0.1}$ and TPR@FPR=0.1. The first column lists the feature groups used in the experiments and other columns present the results of different methods. Since the PLR model is equivalent to the regular logistic regression model when only the content features are used, we leave the corresponding results of PLR and PLR+ empty in the table. Generally speaking, adding new groups of features tend to increase the performance for all the methods we tested. However, given the same groups of features, PLR is consistently better than LR. In addition, by allowing different smoothing parameters for individual component logistic regression models, PLR+ is again better than PLR in both quality measures. In contrast, naive Bayes is inferior to any other methods, which is consistent to what we have observed in Fig. 4. Notice that because the number of testing examples is quite large (i.e., 12,000), the differences between any two methods when using the same sets of features are statistically significant for the true positive-rate measure. Similarly, the differences between any two sets of features when used by the same model are also statistically significant.

The experimental results on the TREC datasets also lead to similar conclusions. For example, Fig. 5 shows the ROC curves of four different methods using both content and sender features on the TREC-06 dataset. Because this dataset is much easier than the Hotmail data as previously discovered, we focus on a lower false-positive region. Although the performance gaps are much smaller here, we can still see that PLR+ is slightly better than PLR, which is again better than the logistic regression model. Naive Bayes is still inferior to other methods on this dataset. Tab. 2 shows the results of using different feature group combinations and different learning methods in $\text{AUC}_{0.01}$ and TPR@FPR=0.01. As indicated by the results, we can see that using both sender and content features is indeed better than using only the content features in most of the cases. Moreover, PLR is able to use the additional sender features more effectively and performs better than LR on both TREC datasets. The performance difference between PLR and PLR+ on the TREC datasets is much smaller.

The improvement of choosing different smoothing parameters (i.e., PLR+ vs. PLR) is relatively smaller compared to what we previously observed in the Hotmail dataset. In addition, PLR+ is better than PLR in only TPR@FPR=0.01 on the TREC-05 data, and only in $\text{AUC}_{0.01}$ on the TREC-06 data. Notice that on the TREC-06 data, the differences in TPR@FPR=0.01 between PLR and PLR+, and two LR models using different features are not statistically significant. All other comparisons in TPR@FPR=0.01 between different methods using the same sets of features, or the same method learned on different sets of features, are statistically significant.

7. HANDLE USER INFORMATION

Although in Sec. 6, we have observed that including user information in the model can improve spam filtering (e.g., Tab. 1 and Fig. 4), such information is not always available. As discussed earlier, a spam filter is often trained using data collected from one group of users but may need to be applied to messages sent to others. It is thus interesting to explore how we can still extend the user model in various realistic settings.

In this section, we first revisit how we incorporate user features in our PLR model and suggest alternative learning methods. We next study experimentally two different scenarios when the user information is limited and demonstrate how a modified user model can still help to improve the PLR model.

7.1 User Model

Recall that the features extracted from user information in Sec. 6 are merely the ids of the email recipients. When incorporating this information in the PLR model, the corresponding component logistic regression model estimates $P(Y|\mathbf{X}_u)$ of each testing message, where $\mathbf{X}_u$ is the group of user features of that particular example. Because only one of the binary features that represents the message recipient is active, this model basically predicts how likely a message received by this user is spam, without knowing any content or sender information regarding the message.

Although this user model can be learned using logistic regression as usual, it is not hard to show that without regularization and the bias term, the estimation of $P(Y|\mathbf{X}_u)$ can be derived using frequency counting directly:

$$\hat{P}(Y=1|\mathbf{X}_u) = \frac{\#\text{count}(Y=1,u)}{\#\text{count}(u)}, \tag{7}$$

where $\#\text{count}(Y=1,u)$ and $\#\text{count}(u)$ are the numbers of spam messages and all messages user u receives, respectively. Eq. 7 provides us a more straightforward method to build the user model. As we will see later, by augmenting different smoothing techniques, Eq. 7 can easily be revised to handle incomplete information of user preference.

7.2 Scenarios of Partial User Information

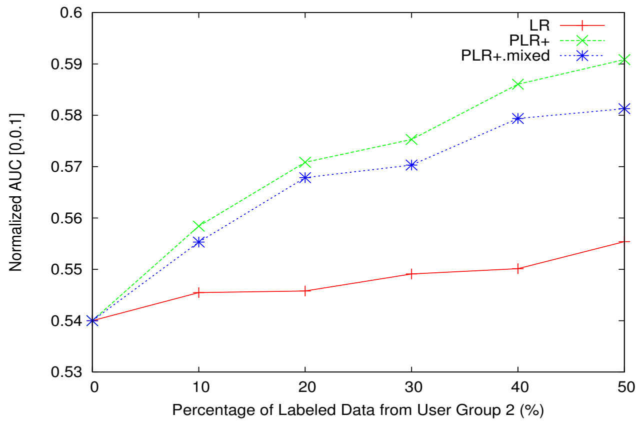

Figure 6: The results of three models by varying the amount of training data from user group 2.

In order to create scenarios where user information is limited, we split the training and testing data of the Hotmail collection in the following way. We first randomly divide all the email recipients into two user groups of roughly the same size. Messages in the original training set sent to these two groups of users are denoted as U_1 and U_2 respectively, where U_1 contains 381,091 messages and U_2 has 383,909 messages. For the experiments in this section, the training data includes all messages in U_1 and zero or some messages in U_2. For testing, only the email in the original test set sent to the recipients in user group 2 is retained, which consists of 45,092 messages in total. To simplify the tests, we use only content features and user features.

The first scenario we test is when fewer messages in U_2 are available for training. We generate different sets of training data by adding all the U_1 messages, along with randomly selected portions of U_2 messages in various sizes. Besides the LR and PLR+ model as described in the previous section, we also test a new model, PLR+_{mixed}. This model is similar to PLR+, but its content-based component model is trained using only messages in U_1, while its user-based component model is trained using all the given training messages. In other words, the PLR+_{mixed} model can estimate $\hat{P}(Y=1|\mathbf{X}_u)$ for group 2 users using the limited training data, but does not have access of the email content. Since PLR+_{mixed} uses the same learning method but is given less information, its performance should be bounded by what PLR+ can achieve.

Fig. 6 shows the performance in terms of normalized AUC at 0.1 false-positive rate of these three different models in this setting. When there is no training message from U_2, three models reduce to the same logistic regression model that only uses content features from messages in U_1. However, when more messages from U_2 are added for training, the regular logistic regression model can only gain limited improvement in $\text{AUC}_{0.1}$, compared to the PLR+ model. Perhaps more interestingly, the performance gap between PLR+ and PLR+_{mixed} tends to be small, which suggests that the improvement of the PLR model mainly comes from the additional user information instead of the email content of messages in U_2.

The second scenario assumes that the training data consists of all U_1 and U_2 messages, but all the messages from U_2 are *unlabeled*. An interesting question here is whether we can apply the "bootstrapping" technique in semi-supervised learning to "guess" the labels of these unlabeled messages from U_2, and use them to derive the corresponding user model. Suppose $\hat{P}_c$ is the content-based model trained using messages from U_1, m_u is the subset of unlabeled messages in U_2 that are sent to a specific user u, and $\mathbf{X}_c(i)$ represents the content features of the message i. We use the following formula to predict how likely a message sent to user u is spam.

$$\tilde{P}(Y=1|\mathbf{X}_u) = \frac{\sum_{i \in m_u} \hat{P}_c(Y=1|\mathbf{X}_c(i)) + \beta P(Y=1)}{|m_u| + \beta}, \tag{8}$$

where β is the smoothing parameter.

Eq. 8 can be viewed as an enhanced version of Eq. 7. It uses a content-based model to derive the expected number of spam messages received by this user, and estimates the ratio of the number of spam messages to the number of total messages, with Dirichlet smoothing. Recall that when there is no message from U_2 in the training set, the $\text{AUC}_{0.1}$ score is 0.540 as indicated in Fig. 6. However, by including the user model provided by Eq. 8 with $\beta = 5.0$ (tuned using the validation set) in the PLR+ model, the $\text{AUC}_{0.1}$ score is increased to 0.566. From Fig. 6, we can see that this has the same effect as including nearly 20% of the labeled messages from U_2 for training.

Notice that Eq. 8 is just one baseline method to derive the user model when the labeled data from the specific user is unavailable. In practice, there could be better sources to collect information of user preference. For example, most modern email systems provide various mechanisms for users to report junk mail. By using the count of junk mail, we can estimate the number of spam messages received by this user more robustly to further improve the results.

8. RELATED WORK

Our PLR models can be viewed as a spectrum of hybrid generative/discriminative models that range between naive Bayes and logistic regression. Furthermore, every model in this family has the same linear functional form. Ng and Jordan [24] analyzed logistic regression and naive Bayes theoretically and concluded that although not performing better than logistic regression with enough training data, naive Bayes needs much fewer examples (i.e., $O(\log(n))$ vs. $O(n)$, where n is the dimensionality of the weight space) to approach its asymptotic performance. Following the same arguments, the PLR model may also converge to its asymptotic result faster than regular logistic regression, although the performance could be suboptimal compared to what logistic regression can potentially achieve.

An alternative hybrid model was proposed by Raina et al. [26] who learn component naive Bayes classifiers first and then combine them using the weights learned by logistic regression with leave-one-out estimate. They demonstrated that by separating features that obviously violate the conditional independence assumption into different groups, their hybrid model can outperform naive Bayes, but is still inferior to logistic regression on most of the benchmark datasets. Their model can be treated as a way to improve naive Bayes by slightly relaxing the conditional independence assumption, which can be analogous to several previous efforts on enhancing the prediction accuracy of naive Bayes (e.g., [2, 27, 19]). In contrast, our PLR model applies the naive Bayes assumption to combine multiple logistic regression models. It is not only better than naive Bayes, but also somewhat unexpectedly outperforms logistic regression on a real-world application, email spam filtering, on a very large dataset. The good empirical performance of the PLR model may also be explained following the same argument suggested by Domingos and Pazzani [9], where they argued that naive Bayes can be optimal under 0-1 loss in several learning problems which do not satisfy the conditional independence assumption.

The PLR model can also be viewed as a special form of model combination, where multiple classifiers are learned through either different learning algorithms or different samples of the training data and the final prediction is made by the combined results. There exists a considerable amount of literature on this topic and interested readers can find a survey in, e.g., [18, 8, 3]. In the model combination paradigm, partitioned logistic regression can be categorized as a model of logarithmic opinion pools in Hinton's *products of experts* framework [16], where the final prediction is made by averaging the estimated log-odds of individual classifiers (i.e., experts). Following the same naive Bayes assumption, Kahn proposed an improved version of the logarithmic opinion pools by calibrating the probability estimation of the experts and also learning the combining weights discriminatively [17]. Compared to the general model combination framework, the main difference in partitioned logistic regression is that the individual models or experts are learned specifically on disjoint feature spaces. The overall functional form is exactly the same as learning a logistic regression model on all the features. The prediction function does not need to be changed at all, only the weights and the bias term are set differently. These properties make our PLR model very easy to use in reality when a linear classifier has been implemented and used.

As for the application of spam filtering, statistical approaches have been claimed more effective than rule-based systems and have been used extensively [1]. Linear classifiers, such as naive Bayes [28, 23], logistic regression [13, 33] and linear SVMs [10, 30], are especially popular for this task due to their ability to handle large feature spaces efficiently. Various model combination methods have also been explored. For example, Hershkop and Stolfo [15] experimented with different model combination strategies for models trained using different algorithms and on different features. More traditional ensemble-like methods such as a cascade of classifiers [33] and stacking classifiers [29] have also been applied to spam filtering. Most work on spam filtering treats it as a regular text classification problem. The use of non-content information has been less studied. Using the sender information (i.e., the IP address sending the mail) is first studied in [20], which analyzes the whole SMTP path and uses it to enhance the filter.

Personalized email spam filtering has typically been viewed as training a model that fits better individual user's mail distribution, instead of adjusting the filter to learn user preference. For example, Bickel and Scheffer [4] use a Dirichlet process model to re-sample the training data for each user and to make the distribution of this new training dataset close to the messages this user receives. However, this strategy is quite expensive in computation and may not be feasible for a Web mail system that has hundreds of millions user accounts. Instead of creating new training data, Segal [31] proposed a method to combine a globally trained model with a model trained using only personal email. While the globally trained spam filter always outperforms the locally trained one, the combined approach still gains some improvement. Notice that in both cases, the class label of an email message is assumed to be independent of the recipient of the mail. In other words, the user preference issue of the gray mail problem described previously is not handled by either of these approaches. In comparison, our personalized spam filtering system using the PLR model deploys a simple light-weight user model, which is highly scalable for practical applications.

9. CONCLUSIONS

In this paper, we present partitioned logistic regression, which is a novel hybrid model of the generative model, naive Bayes, and its discriminative counterpart, logistic regression. By assuming that features can be grouped into disjoint subsets that are conditionally independent given the class label, individual models are learned by logistic regression using only the corresponding subsets of features, and combined following the naive Bayes principle. Our model not only outperforms both naive Bayes and logistic regression in experiments on synthetic and real data, but also enjoys several advantages that make it a suitable learning method in practical applications. Its identical functional form makes it easy to be used. By grouping potentially dependent features together, it is also easy to incorporate different types of information and learn a good model with fewer training examples – both are crucial in practice to improve the performance of the final application.

We also demonstrate the superiority of our proposed model on the task of email spam filtering. On a fairly large real data set, the method can easily learn a model that combines the content information and sender reputation, as well as the individual user preference. Its performance in various evaluation metrics is better than both naive Bayes and logistic regression when learning on the same set of features. Moreover, by tuning the smoothing prior of each individual model, the performance gap can even be further increased.

Our work shows a promising direction for creating hybrid generative/discriminative models, and also raises several interesting questions for future research. In particular, we would like to explore methods that can automatically group features, which would reduce the burden of a domain expert to decide the best feature grouping. In addition, we would also like to study whether the individual models can be better combined to yield higher overall accuracy.

10. ACKNOWLEDGMENTS

We thank Paul Bennet and Joshua Goodman for many helpful discussions.

11. REFERENCES

[1] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *SIGIR-2000*, pages 160–167, 2000.

[2] P. N. Bennett. Using asymmetric distributions to improve text classifier probability estimates. In *SIGIR-2003*, 2003.

[3] P. N. Bennett. *Building Reliable Metaclassifiers for Text Learning*. PhD thesis, Carnegie Mellon University, 2006.

[4] S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems 19 (NIPS-2006)*, pages 161–168. 2007.

[5] G. Cormack. TREC 2006 spam track overview. In *Proceedings of TREC-2006*, 2006.

[6] G. Cormack and T. Lynam. TREC 2005 spam track overview. In *Proceedings of TREC-2005*, 2005.

[7] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[8] T. G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000.

[9] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.

[10] H. Drucker, D. Wu, and V. Vapnik. Support vector machines for Spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

[11] D. Fallows. Spam: How it is hurting email and degrading life on the Internet. *Pew Internet and American Life Project*, October 2003.

[12] J. Goodman. Sequential conditional generalized iterative scaling. In *ACL-2001*, pages 9–16, 2001.

[13] J. Goodman and W. Yih. Online discriminative spam filter training. In *CEAS-2006*, 2006.

[14] J. He and B. Thiesson. Asymmetric gradient boosting with application to spam filtering. In *CEAS-2007*, 2007.

[15] S. Hershkop and S. J. Stolfo. Combining email models for false positive reduction. In *KDD-2005*, pages 98–107, 2005.

[16] G. Hinton. Products of experts. In *Proc. of the 9th International Conference on Artificial Neural Networks (ICANN99)*, pages 1–6, 1999.

[17] J. M. Kahn. A generative bayesian model for aggregating experts. In *UAI*, pages 301–308, 2004.

[18] J. Kittler, M. Hatef, R. P. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[19] A. Kolcz and W. Yih. Raising the baseline for high-precision text classifiers. In *KDD-2007*, 2007.

[20] B. Leiba, J. Ossher, V. T. Rajan, R. Segal, and M. N. Wegman. SMTP path analysis. In *CEAS-2005*, 2005.

[21] D. Lowd and C. Meek. Adversarial learning. In *KDD-2005*, pages 641–647, 2005.

[22] D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *CEAS-2005*, 2005.

[23] V. Metsis, V. Androutsopoulos, and G. Paliouras. Spam filtering with naive Bayes – which naive Bayes? In *CEAS-2006*, 2006.

[24] A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Proceedings of NIPS 14*, 2002.

[25] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999.

[26] R. Raina, Y. Shen, A. Ng, and A. McCallum. Classification with hybrid generative/discriminative models. In *Proceedings of NIPS 16*, 2004.

[27] J. Rennie, L. Shih, J. Teevan, and D. Karger. Tackling the poor assumptions of naive Bayes text classifiers. In *ICML-2003*, 2003.

[28] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[29] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. D. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. In *EMNLP-2001*, pages 44–50, 2001.

[30] D. Sculley and G. M. Wachman. Relaxed online SVMs for spam filtering. In *SIGIR-2007*, pages 415–422, 2007.

[31] R. Segal. Combining global and personal anti-spam filtering. In *CEAS-2007*, 2007.

[32] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[33] W. Yih, J. Goodman, and G. Hulten. Learning at low false positive rates. In *CEAS-2006*, 2006.

[34] W. Yih, R. McCann, and A. Kolcz. Improving spam filtering by detecting gray mail. In *CEAS-2007*, 2007.