# Manifold learning for online peer-to-peer flow classification

**Bruno Ribeiro** RIBEIRO@CS.UMASS.EDU
**Don Towsley** TOWSLEY@CS.UMASS.EDU
Department of Computer Science, University of Massachusetts at Amherst, Amherst, MA, USA

## Abstract

This work presents an application of Lagrangian eigenmaps for unwanted network traffic classification. This classification is crucial when enforcing network usage policies aimed at curbing the growing illegal peer-to-peer file sharing activity, which now accounts for a large fraction of the total Internet traffic. In this work we show that this application domain has inherent issues that makes more traditional learning methods unfit for the task but are successfully addressed by manifold learning methods.

## 1. Introduction

This work provides an in-depth view of online peer-to-peer traffic classification. Peer-to-peer (p2p) traffic classification imposes a series of requirements on classifiers. We show that these requirements can be translated into geometric constraints of the feature space. Moreover, we show these constraints to limit the efficacy and applicability of traditional learning methods. The implication of our study is that graph-based manifold learning methods, more specifically Laplacian eigenmaps, a perfect fit for our problem whereas more traditional learning methods, such as Gaussian Mixture Models (GMMs) and $k$-means, are not.

In this work we present a realistic p2p traffic classification scenario and conclude that it requires more advanced Machine Learning tools than the ones used in earlier works (Bernaille et al., 2006). In Section 2 we introduce the peer-to-peer traffic classification problem and its more traditional solutions under a Networks engeneering perspective. The introduction of the problem motivates Section 3 where we present a series of requirements that are then translated into constraints on the classifier feature space. In Section 4 we see that more traditional learning methods are unfit for our application, which takes us to manifold learning methods. Later, we present our experimental results against a Gaussian Mixture Model classifier in Section 5 and show that even under a more idealized scenario where traditional classifiers are not heavily punished by their lack of knowledge of the spacial constraints, the GMM is outperformed by the Laplacian eigenmaps classifier. The GMM classifier was previously reported to be the best classifier for p2p traffic classification (Bernaille et al., 2006). Finally, we introduce the related work in Section 6 and conclude, showing future directions, in Section 7.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

## 2. Background and Motivation

The Internet is designed to allow free information flow between users. While this freedom must be defended, increasing illegal activities should also be addressed. One of the most prevalent forms of misbehavior is the use of peer-to-peer (aka p2p) applications for illegal music and movie sharing. Academic institutions (mainly universities) are among the greatest sources of these problems: typically p2p flows account for 60% of all their Internet traffic, 90% of which consists of copyright violations, according to a recent MPAA report (Taylor, 2005). Automated classification of p2p traffic is needed if network administrators are serious about curbing illegal network activity.

Curbing such illegal activity without violating users' freedom can be achieved by means of a policy that discourages p2p file sharing without censoring it. Such a policy has two main steps: (1) Detect active TCP/UDP connections (aka flows) that belong to p2p applications and (2) limit the aggregate p2p flow throughput (number of bytes per second) of a host while making the best effort to not interfere with other types of traffic. This policy mandates the classification to be online, i.e., performed before the flow ends. A great deal of recent work has targeted the development of application flow content parsers. These techniques consist of parsing packet contents to classify flows into applications (Cisco Systems, 2005; Moore & Zuev, 2005; Karagiannis et al., 2004; Sen et al., 2004), just to cite some approaches. Today's most prevalent p2p protocol parsers are not CPU intensive. Parsing a thousand flows per second can be CPU intensive but still manageable for small to

moderate sized networks (Cisco Systems, 2005). However, given that p2p protocol designers can be seen as adversaries to our policy it is easy to see that in the near future more and more processing power will be needed to classify p2p flows. In this scenario, network routers would not have enough processing power to run complex p2p parsers over all network flows. Even in small networks expensive specialized hardware would be needed. This last statement brings us to the first important rule in p2p flow classification: *Any flow classifier is working against an active adversary*. Ignoring this aspect can condemn a solution to be short lived.

In this work we present a classifier that can be used to implement a policy to discourage p2p Internet file sharing in a campus or an enterprise network. This classifier would complement today's content parsers in a scenario where the latter can only classify a fraction of the flows. Using classifiers against adversaries is a known problem in spam e-mail classification (Dalvi et al., 2004). Spam e-mail classification needs a classifier that can deal with evading adversary strategies to increase the misclassification rate. But our problem is more than just flow misclassification: The rate limiter should be effective enough to discourage p2p file sharing. This motivates us to compile a list of possible attacks on our policy:

**Adversary attacks**

1. The p2p application opens multiple connections and reopens connections that had their throughput artificially reduced.
2. The p2p application creates p2p flows that closely mimic features of non-p2p flows.
3. The p2p protocol designer increases the content parsing detection complexity, drastically reducing the number of parsed flows for a given CPU usage.

The above adversarial attacks and the nature of the problem impose a series of constraints to our classifier that are compiled as a series of *requirements* for our classifier as seen next.

## 3. Classifier requirements

Based on the previous section we can devise a series of requirements for our classifier:

(1) *Classify a flow right after the arrival of its first packets.* The flow throughput limiter should kick in as fast as possible to defeat the **first** adversarial attack of Section 1. TCP flows are designed to exchange data at lower rates during the beginning of their connections. The average flow throughput prior to classification should not be higher than the desired rate limited throughput.

(2) *Classify over a large set of flow features.* Increasing the number of flow features to be classified directly translates into greater security against p2p flows that try to mimic non-p2p flows as described in the **second** adversarial attack of Section 1. Some important features, such as Operating System fingerprints, are not easy to translate into meaningful values that can be used inside a classifier.

(3) *Learn from a partially labeled training data set.* Labeling flows can be an expensive operation if flow content parsing complexity is high due to the **third** adversarial attack of Section 1. A good classifier should be able to inflate its training set with unlabeled samples.

(4) *Guaranteed fast classification.* It needs to work over a high speed Internet links with little CPU resource consumption *per* flow.

Classifier requirement (1) implies that only features from the first $K$ packets in the flow can be used, with $K$ typically very small. These features must be simple to extract while allowing a differentiation of p2p and "non-p2p" flows. This task relies upon a good distance metric between the collection of features of two distinct flows. Small distances between two flows imply that these flows have similar characteristics. Distance metrics in our problem are a non-trivial issue, which bring us to the classifier requirement (2): Learn from "non-numeric" features such as the Operating System type. Assume a feature space "Operating System type" (OS type) and "size of the first packet in the connection". Let $w_1 = (\text{Windows}, 100)$, $l_1 = (\text{Linux}, 200)$, and $s_1 = (\text{Solaris}, 200)$ be samples from this feature space. Vector $w_1$ represents a flow from a Windows machine with first packet size of 100 bytes. Is the distance between $w_1$ and $l_1$, denoted as $|w_1 - l_1|$, larger than $|w_1 - s_1|$? It is unlikely that we will be able to find an answer to this question. We are only sure that two Linux flows should be much closer to each other than Linux and Windows flows. A simple solution to the above problem is to create a classifier for each OS type. However, this is undesirable as it can seriously impair the classifier by reducing the number of training samples. Even in the case where all flows have the same OS type, a definition on how to measure packet sizes distances is needed. Let $w_2 = (\text{Windows}, 110)$. We can probably agree that $|w_1 - w_2|$ is small. But let $w_3 = (\text{Windows}, 956)$ and $w_4 = (\text{Windows}, 1024)$. Is $|w_2 - w_3| < |w_2 - w_4|$ or $|w_2 - w_3| > |w_2 - w_4|$? Vectors $w_1$ and $w_2$ could represent flows with text messages and $w_3$ and $w_4$ could represent flows from a file download, which make us conclude that the difference between packet sizes is only really important when it is small.

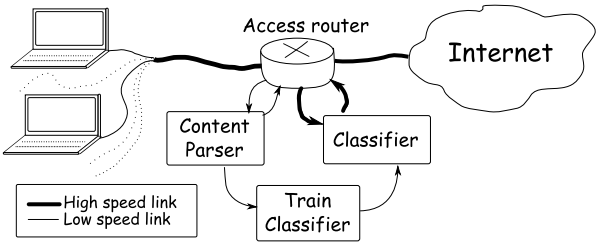


*Figure 1.* Traffic classification schematics

| Name | Definition |
|---|---|
| $l$ | Number of training flow samples |
| $S$ | Set of training flow records |
| $Y$ | Set of training labels |
| $h$ | Number of labeled training samples |
| $l-h$ | Number of unlabeled training samples |
| $m$ | Number of features |
| $x_i$ | Training flow $i$ feature vector |
| $y_i$ | Flow $i$ app. type: 1 if p2p, $-1$ otherwise |
| $\mathcal{X}$ | Set of training feature vectors |
| $k$ | Number of nearest neighbors |
| $L$ | Graph Laplacian |
| $v_i$ | $i$ th smallest eigenvector of $L$ |
| $n$ | Number of eigenvectors used in classifier |

*Table 1.* Notations table.

## 4. Classifier design

Having reviewed the desired classifier requirements let us now develop a classifier that conforms to these specifications. Figure 1 shows a schematic of our approach. The box labeled "Classifier" represents our proposed (main) classifier. The figure displays an *access router* that can be a university gateway concentrating all university traffic to/from the Internet. Links that come in and go out of this type of access router are commonly high speed optical cables. The main classifier is in charge of classifying all flows traversing the *access router*. This classifier needs to be calibrated using a training data set. A subset of this training data comes from a fraction of the total number of flows that is redirected from the access router to the *Content Parser*, such as (Cisco Systems, 2005), and then classified into p2p or non-p2p flows. We assume that the *Content Parser* is able to perfectly classify all flows. The remaining fraction of the training data is comprised of unlabeled flows. This guarantees that the classifier requirement (3) is satisfied: Ability to learn from a partially labeled training data set. Feeding our training data set with content parsed flows ensures that our training data is up-to-date, making the classifier more resilient to changing adversary strategies.

The high level design described above places the box "Classifier" into the core of our p2p file sharing discouragement policy. It is now time to turn our attention to the construction of such classifier. Let $S_l = \{s_1, \ldots, s_l\}$ be the set of flow records corresponding to our training data. Let $y_i \in \{-1, 1\}$ be a label for flow $i$ such that $y_i = 1$ if and only if flow $i$ is a p2p flow, and $y_i = -1$ otherwise. Assume, without loss of generality, that $(s_1, \ldots, s_h)$ are labeled samples and $(s_{h+1}, \ldots, s_l)$ are unlabeled samples. Let $Y = (y_1, \ldots, y_h)$ be the set of labels associated with flow records $S_l$. In what follows we also omit the dependency on $l$ to simplify the notation. Let $\phi_r$ be the $r$ th feature extraction functions and let $x_i = (\phi_1(s_i), \ldots, \phi_m(s_i))$ and $\mathcal{X} = (x_1, \ldots, x_l)$. Note that $m$ is the number of features extracted from the training data set. Table 2 shows four features from an actual trace. The inner product $< x_i, x_j >$ is defined if and only if samples $x_i$ and $x_j$ are "close enough". In the OS fingerprint example of Section 3, a "Windows" flow and a "Linux" flow are not considered to be close enough. Let $\|x_i - x_j\| =< x_i, x_j >^{1/2}$ be a distance metric over $\mathcal{X}$. The fact that the inner product is not defined over all points in space $\mathcal{X}$ is an issue for classifiers such as the K-means, Gaussian Mixture Models, and HMMs used in (Bernaille et al., 2006). All these requirements motivate our search for a classifier that can easily accommodate them.

*Choosing a classifier.* We use the above requirements to select a set of classifiers that can be effectively used in our problem domain. Graph-based methods for manifold learning are clearly a perfect fit [Saul et al, "Dim red survey"]. Among the graph-based methods, isomap (Tenenbaum et al., 2000), maximum variance unfolding (Weinberger & Saul, 2004), locally linear embedding (Roweis & Saul, 2000), and Laplacian eigenmaps (Belkin & Niyogi, 2002) are the best known. Please refer to (Saul et al., 2006) for a compreehesive survey on the subject. We choose Laplacian eigenmaps manifold learning for our classifier as its learning phase scales best, together with locally linear embedding, for moderately large data sets ($< 10,000$) (Saul et al., 2006). Next we describe the training phase of our Laplacian eigenmap classifier.

*Training.* Let $G(X, E)$ be an undirected weighted graph with edges $(x_i, x_j) \in E$, where $x_i, x_j \in \mathcal{X}$ if and only if $\|x_i - x_j\|$ (the distance between vertices $x_i$ and $x_j$) is defined and is among the $k$ th smallest distances from vertices $x_i$ or $x_j$ to all other vertices. An edge $(x_i, x_j)$ is associated with a weight obtained from the function

$$W(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|}{t}\right).$$

Function $W$ decays exponentially as the distance $\|x_i - x_j\|$ grows larger thus enforcing locality. Constant $t$ controls the exponential decay.

Let $A$ be the adjancecy matrix of graph $G$. We denote the corresponding graph Laplacian by $L = A - W$. Let

$v_1, \ldots, v_n$ be the $n$ eigenvectors corresponding to the $n$ smallest eigenvalues of the eigenvector problem $Lv = \lambda v$. Let $v_j(i)$ be the $i$ th element of eigenvector $v_j$. We are looking for a vector of coefficients $\alpha = (\alpha_1, \ldots, \alpha_n)$ that satisfies

$$\underset{\alpha \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{h} \left( y_i - \sum_{j=1}^{n} \alpha_j v_j(i) \right)^2 . \qquad (1)$$

Let $P$ be an $h \times n$ matrix with $P = [v_j(i)]$ for $i \leq h$. The pseudo-inverse of $P$ can be used to find a solution to equation (1):

$$\alpha = (P^{\mathsf{T}} P)^{-1} P^{\mathsf{T}} Y.$$

Note that the training phase can be easily carried out in a backend server and thus does not share the same CPU resource constraints as the classification phase. Next we describe how to use the trained classifier for p2p flow classification.

*Classification.* Let $c$ be a flow record that needs to be classified and $x^{(c)} = (\phi_1(c), \ldots, \phi_m(c))$ be its feature vector. Let $N$ be a set with the $k$ nearest neighbors of $x^{(c)}$, i.e., $\forall x_i \in \mathcal{X}$, $x_i \in N$ if and only if $\|x^{(c)} - x_i\|$ is among the $k$ smallest of all distances $\|x^{(c)} - x_j\|$ $\forall x_j \in \mathcal{X}$. Recall that $\mathcal{X}$ is our training set. Unlabeled flow $c$ is assigned class $y^{(c)}$, computed by the expression

$$y^{(c)} = \sum_{j=1}^{n} \alpha_j \sum_{\forall x_i \in N} W(x^{(c)}, x_i) v_j(i). \qquad (2)$$

Last, we address classifier requirement (4): The classification algorithm must be fast. From the above we can conclude that its slowest part is to compute the $k$-nearest neighbors of $x^{(c)}$ in $\mathcal{X}$. A simple naive algorithm takes $O(l)$ time, which is prohibitively slow for high speed Internet links. But it is possible to find an $\epsilon$ approximation to the $k$-nearest neighbors problem with time that is logarithmic in $l$ (Liu et al., 2004). In practice this approximation is shown to be fast and quite accurate (Liu et al., 2004), which solves classifier requirement (4).

Next we test the accuracy of our classifier over real Internet flow features.

## 5. Experimental evaluation

In this section we evaluate our classifier against the (Bernaille et al., 2006) data set. This data set is, to the best of our knowledge, the only publicly available network trace with p2p flow labels. In what follows we refer to the (Bernaille et al., 2006) data set as data set $U$. We use data set $U$ to compare our methodology against the methodology described in (Bernaille et al., 2006). This comparison shows our classifier to be significantly more accurate in classifying p2p flows (true positives) than the best classifier in (Bernaille et al., 2006) (Gaussian Mixture Model or GMM) while keeping rougthly the same fraction of true negatives (fraction of non-p2p flows correctly classified).

Data set $U$ is comprised of $24,261$ flows divided into $5,697$ p2p flows (bittorrent, edonkey, gnutella, fasttrack) and $18,564$ non-p2p flows (ftp, http, https, ssh, msn, ...). Each flow has its application label ("p2p" or "non-p2p"), its first four packets sizes and directions, and port number. Features such as flow OS fingerprint (Zalewski, 2006) or IP source and destination addresses are unfortunately absent from $U$. The presence of such features would put our classifier in a much more advantageous position relative to the classifiers in (Bernaille et al., 2006) as none of them are suited for the use of "non-numeric" features. Collecting Internet traces with all these features is subject of future work as it is delicate task which raises many security and privacy concerns.

Although port numbers could improve the classification accuracy (most applications use unique port numbers), we choose not to add this feature to the classifier as we believe it could be turned against us. Consider a scenario with 1000 web flows (on port 80), 1000 e-mail flows (on port 25), 1000 ftp flows (on port 21), and 30 p2p flows equally distributed among the same ports: 80, 25, and 21. By definition, a port 21 flow is far away from any other flows with port numbers other than 21. Thus p2p flows are now divided into three groups of 10 flows each and "near" much larger groups of flows with 1000 flows each which can confuser our classifier. As users are adversaries and they can choose p2p flow port numbers, the addition of this feature can have unpredictable consequences.

Section 4 shows that our classifier needs a locally defined distance metric $\|\cdot\|$ over $U$. Let $\phi_r$ be the size and the direction of the $r$ th packet in the flow. The direction of the $r$ th packet is represented by the sign of $\phi_r$. If $\phi_r < 0$, the packet is an incoming packet (from the Internet) and when $\phi_r > 0$ the packet is an outgoing packet (to the Internet). Table 2 shows an excerpt with some values of $\phi_r$, $r \in \{1, 2, 3, 4\}$ contained in our data set. Recall that $\phi_r(s_j)$ is the size of the $r$ th packet of the $j$ th training flow $s_j$. The distance between $x_i, x_j \in \mathcal{X}$ is given by

$$\|x_i - x_j\| = \left( \sum_{r=1}^{4} (\phi_r(s_i) - \phi_r(s_j))^2 \right)^{1/2}$$

if and only if $\|x_i - x_j\| \leq d$. Otherwise the distance between $x_i$ and $x_j$ is not defined. In our experiments we use $d = 500$.

In the first set of experiments we train our classifier using $1,000$ labeled and $1,000$ unlabeled flows selected uniformly at random from $U$ with the restriction that half of

| Application | $\phi_1$ | $\phi_2$ | $\phi_3$ | $\phi_4$ |
|---|---|---|---|---|
| non-p2p | 306 | -250 | -1380 | -487 |
| non-p2p | 418 | -1452 | -31 | -1452 |
| non-p2p | 1460 | 1460 | 604 | -1460 |
| non-p2p | 984 | -161 | 979 | -192 |
| non-p2p | 410 | -1460 | -1460 | -1460 |
| non-p2p | -1460 | -1460 | -1460 | -1460 |
| non-p2p | -23 | 56 | -544 | 544 |
| non-p2p | 26 | -26 | 77 | -157 |
| non-p2p | -6 | -456 | 16 | -68 |
| non-p2p | 69 | -63 | -1460 | -257 |
| non-p2p | 102 | -146 | 67 | 358 |
| p2p | 68 | -68 | -640 | -640 |
| p2p | 68 | -68 | 572 | -572 |
| p2p | 60 | -111 | 68 | -144 |
| p2p | 103 | -68 | 85 | -134 |

*Table 2.* Feature samples *per* application type. $\phi_r$ is the size of the $r$ th packet in the flow for a given application. The value of the packet size signal indicates whether it is an incoming or an outgoing packet.

the samples are p2p flows and the other half are non-p2p flows. Unlabeled flows are obtained by throwing away flow labels. Let $S$ be the training set and $S'$ be a subset of all labeled flows in $S$. We test the classifier sensitivity to $k$ (the number of nearest neighbors), $n$ (the number of basis vectors), and $t$ (the speed of the exponential decay of $W$). The test consists in evaluating its classification accuracy over a data set $Q \subseteq U$ (we evaluate $Q = S'$ and $Q \subset U - S$). The values of $k$, $n$, and $t$ tested are $t \in \{100, 200, 300, 400\}$, $k \in \{6, 8, 12, 14, 16, 18, 20\}$ and $n \in \{100, 150, 200, 250, 300\}$. For each flow $c \in Q$ our classifier outputs $y^{(c)} \in [-1, 1]$. The closer $y^{(c)}$ is to $1(-1)$ the more certain the classifier is that $c$ is a p2p(non-p2p) flow. Then what if $y^{(c)} = 0$? Upon such event we declare flow $c$ to have an *undefined label*. Moreover, this concept can be generalized by defining a range $[-\gamma, \gamma]$ in which flow $c$ is declared to be "undefined" iff $y^{(c)} \in [-\gamma, \gamma]$. Let $\mathcal{I}_\gamma$ be the set of flows declared to be "undefined". Define the true positive(negative) ratio as the fraction of p2p(non-p2p) flows in $Q - \mathcal{I}_\gamma$ to be correctly classified. And define the "undefined label" ratio as the number of flows in $\mathcal{I}_\gamma$ divided by number of flows in $Q$. We expect true positive and negative ratios to increase with the increase in the number of "undefined" flows.

In order to assess how well our classifier represents the training set $S'$, we first compute the classification accuracy over the training set, i.e., $Q = S'$. The best set of parameters in this experiment is $k = 8$, $n = 250$, and $t = 400$. Figure 2 shows a graph of the true positive and negative ratios against the fraction of "undefined" flows for three most representative parameter configurations over 30 inde-

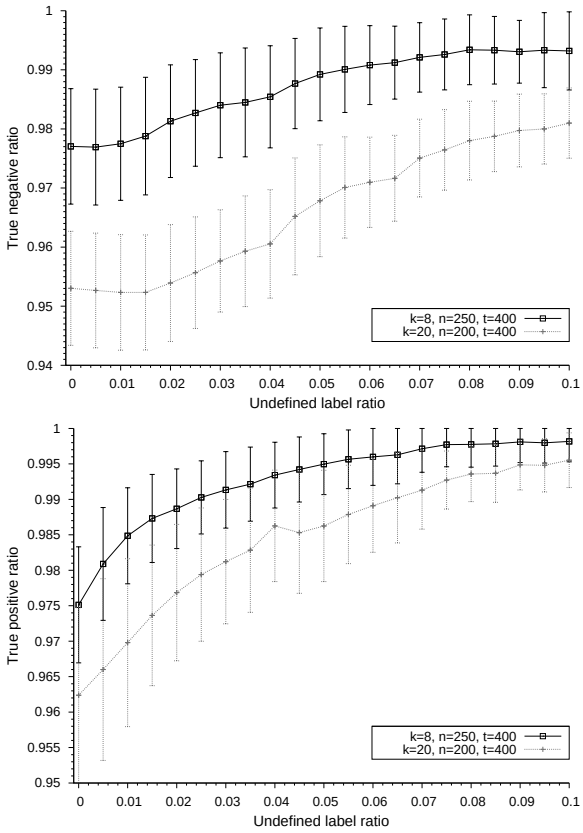


*Figure 2.* Classification of the training set $S'$. Graph of true negative (upper graph) and true positive (lower graph) ratios against the "undefined label" ratio. Most representative curves. From our experiments we conclude that $k = 8$, $n = 250$, and $t = 400$ are the best set of parameters for our classifier.

pendent runs. These curves outline the overall result of our experiment. The vertical bars represent the standard deviation of our measure. From the graph we can see that our classifier is fairly robust to the number of nearest neighbors $k$ (refer to the Appendix section for a similar analysis on the number of basis vectors $n$).

Now we evaluate how well our classifier performs with flows that are **not** part of the training set $S$ and compare the results with the best classifier presented in (Bernaille et al., 2006). Let $Q$ be a set of $2,000$ flows selected uniformly at random from $U - S$. In (Bernaille et al., 2006) three classifiers are presented: k-means, a hidden Markov model and a Gaussian Mixture Model (GMM). According to (Bernaille et al., 2006) the GMM classifier outperforms all other classifiers. We use the author's GMM classifier implementation and compare it to our proposed classifier. Unfortunately no classifier in (Bernaille et al., 2006) is designed to be trained with unlabeled samples. Figure 3 compares the accuracy of our classifier with $k = 8$, $n = 250$, and $t = 400$ to the GMM classifier with its optimal pa-

rameters. These graph show curves of the true positive and negative ratios against the fraction of "undefined" flows. From these results we can conclude that our classifier outperforms the GMM classifier. It also performs quite well even when the number of "undefined" flows is small.

In what follows we repeat the experiment above with no unlabeled training flows, i.e., $S' = \emptyset$. This emulates a scenario where the content parser is able to output more labeled flows than the number of training flows needed for the classifier. Figure 4 shows that our classifier suffers almost no loss in performance and still outperforms the GMM classifier.

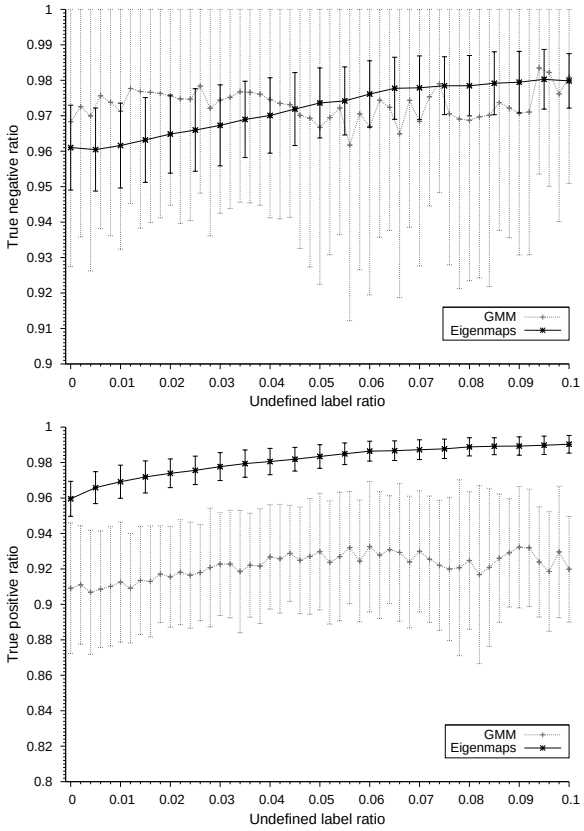


*Figure 3.* Comparison between our classifier and the GMM classifier of (Bernaille et al., 2006). Trained with $2,000$ flows ($1,000$ labeled) and evaluated with $2,000$ flows. Graph of true negative (upper graph) and true positive (lower graph) ratios against the "undefined label" ratio.

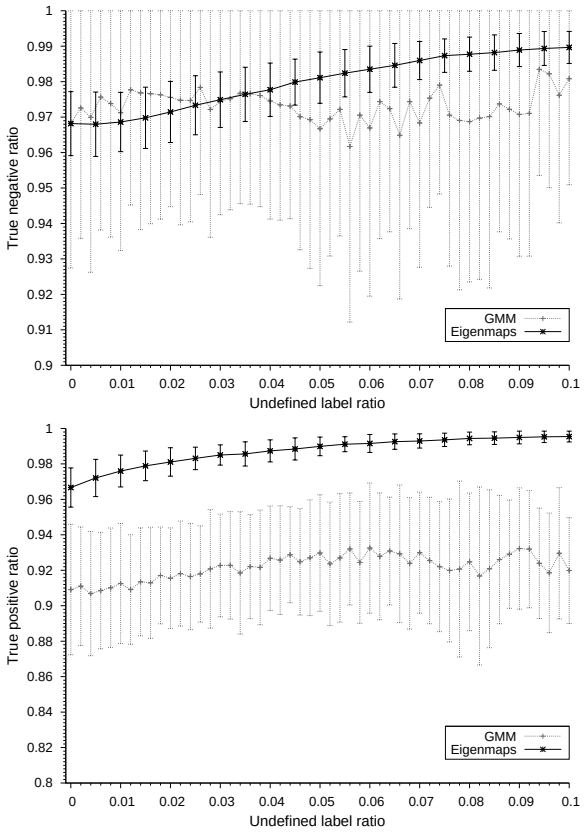


*Figure 4.* Comparison between our classifier and the GMM classifier of (Bernaille et al., 2006). Trained with $1,000$ flows (all labeled) and evaluated with $2,000$ flows. Graph of true negative (upper graph) and true positive (lower graph) ratios against the "undefined label" ratio.

## 6. Related work

The work most closely related to ours is (Bernaille et al., 2006). It uses generative models to label flow applications. Their approach does not fulfill all the classifier requirements presented in Section 3. More specifically, it violates requirements (2) and (3) (refer to Section 3 for the list of requirements). It violates requirement (2) because it does not scale well on the number of flow features and it is unable to use important flow features such as OS fingerprints in a principled manner, without resorting to cumbersome hacks. It also violates requirement (3) for its inability to learn from unlabeled flows.

Networks intrusion and anomaly detection is a well known problem in Machine Learning (Eskin, 2000). Anomaly detection is concerned with identifying patterns that can be considered anomalous with respect to normal system behavior. This is clearly not our case as p2p flows are in fact quite common and cannot be considered an anomaly.

A complementary approach to ours is to classify the role of a host as p2p or non-p2p user according to their network connectivity graph (Karagiannis et al., 2005). It require the analysis to be offline. The analyzed flows/host IPs could be incorporated in the tranning of our algorithm.

## 7. Conclusions and Future work

In this work we study the problem of discouraging p2p file sharing under a scenario where flow content parsing is too CPU intensive to be performed over all network flows. We propose a high level framework comprised of a standard content parser and a learning classifier. We present a set of required characteristics for a learning classifier to countermeasure adversary attacks. We then design a manifold learning classifier that conforms to these specifications in a principled manner. We show our classifier to be more accurate than other classifiers crafted for the same task (Bernaille et al., 2006) (K-means, HMM, and GMM). We finally show that the latter classifiers fail to meet some of the required design principles of online p2p flow classification.

*Future work.* Apply our methodology to Internet traces that contain features such as OS fingerprints, IP addresses, packet sizes and directions, and TCP flags.

## Appendix

In this appendix we look at our classifier's sensitivity to the number of basis vectors $n$. Figure 5 show the accuracy of our classifier for $n \in \{100, 150, 200, 250, 300\}$ when evaluated over the labeled training set $S'$. From the graph we conclude that a greater $n$ gives more accurate results. This is expected as we are evaluating over the labeled training data. But we would like $n$ to be the smallest possible to avoid overfitting the training data and losing generality. The ballance between accuracy and a lower value of $n$ is achieved when $n = 250$. This motivates our choice of $n = 250$ as the best parameter.

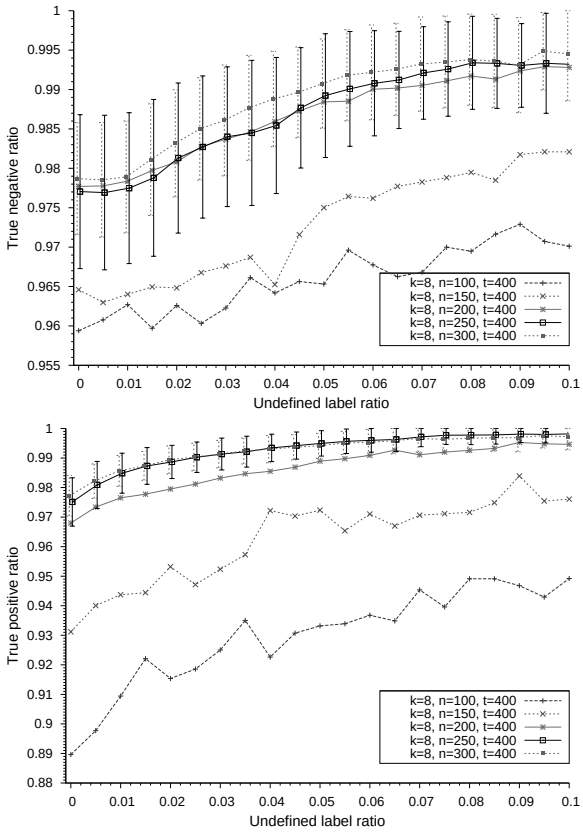


*Figure 5.* Comparison between distinct number of basis vectors ($n$) evaluated over the training set $S'$. We can see from the graph that $n = 250$ is the best choice for parameter $n$ when $k = 8$ and $t = 400$. Graph of true negative (upper graph) and true positive (lower graph) ratios against the "undefined label" ratio.

## References

Belkin, M., & Niyogi, P. (2002). Using manifold stucture for partially labeled classification. *NIPS* (pp. 929–936).

Bernaille, L., Teixeira, R., & Salamatian, K. (2006). Early application identification. *Conference on Future Networking Technologies (CONEXT'06)*.

Cisco Systems (2005). Network based application recognition performance analysis. Cisco 2611, 3745, 7206, 7301, 7505 series routers. Cisco Systems white paper number 0900aecd8031b712.

Dalvi, N., Domingos, P., Mausam, Sanghai, S., & Verma, D. (2004). Adversarial classification. *Proceedings of the ACM SIGKDD conference* (pp. 99–108). New York, NY.

Eskin, E. (2000). Anomaly detection over noisy data using learned probability distributions. *ICML*.

Karagiannis, T., Broido, A., Fabloutsos, M., & claffy, K. (2004). Transport layer identification of P2P traffic. *Proceedings of the ACM/SIGCOMM Internet Measurement Conference* (pp. 25–27). Taormina, Sicily, Italy.

Karagiannis, T., Papagiannaki, K., & Faloutsos, M. (2005). BLINC: multilevel traffic classification in the dark. *Proceedings of the ACM SIGCOMM* (pp. 229–240).

Liu, T., Moore, A., Gray, A., & Yang, K. (2004). An investigation of practical approximate nearest neighbor algorithms. *NIPS* (pp. 825–832).

Moore, A. W., & Zuev, D. (2005). Internet traffic classification using bayesian analysis techniques. *Proceedings of the ACM SIGMETRICS* (pp. 50–60). New York, NY, USA.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*, 2323–2326.

Saul, L. K., Weinberger, K. Q., Ham, J. H., Sha, F., & Lee, D. D. (2006). Spectral methods for dimensionality reduction. In O. C. B. Schoelkopf and A. Zien (Eds.), *Semisupervised learning*. MIT Press.

Sen, S., Spatscheck, O., & Wang, D. (2004). Accurate, scalable in-network identification of P2P traffic using application signatures. *Proceedings of the 13th International World Wide Web Conference* (pp. 17–22). New York City, NY.

Taylor, R. (2005). Piracy on Campus: An overview of the problem and a look at emerging best practices to reduce online theft of copyrighted works. *Presented to the U.S. House of Representatives Subcommittee on Courts, the Internet, and Intellectual Property*. Washington, DC.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Weinberger, K. Q., & Saul, L. K. (2004). Unsupervised learning of image manifolds by semidefinite programming (pp. 988–995. ). Los Alamitos, CA, USA: IEEE Computer Society.

Zalewski, M. (2006). P0f v2 passive OS fingerprinting tool: `http://lcamtuf.coredump.cx/p0f.shtml`.