Just the Fax–Differentiating Voice and Fax Phone Lines Using Call Billing Data

Haim Kaplan, Martin Strauss and Mario Szegedy
AT&T Labs 180 Park Avenue Florham Park, NJ 07932-0971

Summary

We consider the problem of classifying a telephone line as either a voice or fax telephone line based on its long-distance calling patterns (i.e., the time of day and duration of its long distance calls) and based on knowledge about other lines with which calls are placed. We show that both kinds of information are important. In particular, neither choosing new learning algorithms nor collecting more detailed calling pattern data can compensate for the lack of the most important attributes, call duration and neighbor identity. Thus, in our application using the algorithms we studied, we find feature selection to be more important than the choice of learning algorithm.

Introduction

Motivation

We consider the problem of distinguishing fax lines and voice lines on a long-distance telecommunications network. To identify faxes, an algorithm would primarily use call billing data which contains, for each call, the originating and terminating numbers, the date and time the phone call started (the *connect time*), and the duration of the call. This data, available in abundance to telecom providers, is collectively referred to as the *call graph*, since it can be viewed as a directed multigraph in which telephone lines are nodes and telephone calls are arcs. In addition to call graph data, we used a corporate-wide database of voice and fax lines as a source of labeled examples.

In general, we are interested in classifying phone lines into categories such as data, fax, voice, customer help lines, etc., based on the calling records. We are interested in the extent to which this is possible and we are interested in determining good techniques. While in this paper we are particularly concerned with classifying lines as either voice or fax, we point out that the techniques we use may apply in other classification problems as well. The ability to classify lines helps a telecom company to predict revenue and infrastructure needs, to formulate new service offerings, and to detect fraud. For more discussion on analysis of call data (especially in connection with detecting fraud), see [CEWB97].

On one hand one would like to take advantage of well-known supervised learning software such as C4.5. This algorithm requires a flat data set of information about the examples, i.e., there must be a fixed set of attributes whose values summarize all information about the example phone lines. On the other hand, in our data set the information about a particular phone line is encoded in the call billing records associated with the line and the graph structure of our data set contains information valuable to a classification effort.

In our experiment we do flatten the file and compute a set of attributes for each phone line. We take advantage of the graph structure by encoding parts of it that are relevant to our classification problem as attributes.

Focusing on this global approach we encounter two challeges. The first is coming up with the right attributes to use. These attributes should encapsulate as much as possible information from the call graph that is valuable for obtaining a good classification. The second is choosing the right learning algorithm to use on a set of known fax and voice lines to produce a classifier with small error rate on unseen lines. We consider a number of attributes, described in the sequel, and we consider two learning algorithms: our own hyperplane separator algorithm and C4.5, a well-known algorithm that produces a decision tree.

We must compute attributes and choose learning software that takes into account that call billing data is massive. For the most part, one is limited to algorithms that can stream through the data, making full use of each disk block read. Typically, one sorts data once so that related records are contiguous, then operates on runs of related records.

Overall Strategy

We obtained and cleaned a list of labeled voice and fax lines. We collected call data for these lines for a 22-day period, and computed from the call data various attributes for the voice and fax lines. Finally, we ran machine-learning software on the profiles and assessed their error rates.

Data Gathering

We collected the telephone and fax numbers from the employee database of a large company, approximately 80,000 voice lines and 15,000 fax lines. From this, we eliminated by hand:

- numbers listed both as voice and as fax (about 600)
- numbers outside the U.S. and Canada
- obviously incorrect numbers (for example, several employees listed their telephone number as 555-1212, the number for directory assistance in North America.)

By gathering data from one source this way we hope to avoid bias. For example, our fax and voice data have the same geographic distribution, are used by the same human population, and are labeled by the same process. By contrast, if we were successful at distinguishing the fax lines of one phone directory from the voice lines of another, we may be basing our classification on some irrelevant discrepancy between the two databases. Working with homogeneous data has drawbacks as well—success at separating voice from fax lines within a single corporation and its subculture does not necessarily scale to more heterogeneous data. Still, we regard fax/voice differentiation within a homogeneous subpopulation to be an important problem.

This particular data set has another peculiarity. Many outbound calls appear in the billing data as originating from a single central line, so, effectively, we lose outbound data for these lines. Furthermore, since voice lines and fax lines are pooled this way, we cannot use intra-corporate calls of this type to identify the type of

the terminating line based on the type of the originating line. We attempted to remove some of these central lines from the list, but were not thorough. (We removed under 10 numbers altogether, based on high usage levels.)

We collected the telephone billing data for these lines over a 22 consecutive day period in November and December 1997 which included the U.S. holiday of Thanksgiving.

Finally, we removed from the list of examples those lines making fewer than 25 calls in our 22-day period. This leaves only 30% of the examples, but the examples that remain account for 90% of the call traffic. In applications such as fraud detection or market share analysis, it is appropriate to give the three-way classification into low-usage, reasonable-usage fax, and reasonable-usage voice.

Profile Computation

A vector of attributes for a phone line is called a *profile* for the line. We distinguish three types of attributes, whose computation have different algorithmic and data-structural characteristics: accumulative attributes (whose values can be updated based on the current value and the data in a single call), historic attributes (that require call data from more than a single call to update) and propagating attributes (that depend on and therefore propagate our knowledge about other examples). We describe these in more detail below.

Accumulative Attributes

For each phone line, we gathered information about the calls made to it and from it. We gathered the duration and connect time of calls, the total number of weekday and weekend calls, and the total usage in seconds. This information is stored in an aproximate form—for example, we stored the number of calls made during the i'th hour of the day, for $0 \leq i < 24$. Each of these attributes can be computed by maintaining its current value and passing through a stream of call data such that each update needs only the current value and the information in a single call.

We computed the following accumulative attributes:

- Connect time hourly histogram
- Log duration histogram (i.e., the i'th histogram bar is the number of calls of duration 2^i to 2^{i+1} seconds)
- Total usage, in seconds
- Total number of calls
- Total number of calls to 800/888 numbers (toll-free calls)
- Number of calls broken down into weekend v. weekday and inbound v. outbound

Historic Attributes

We also computed some non-accumulative attributes, i.e., attributes that require more than a single call to update. Some of these attributes are more difficult to compute on larger sets of call data. We now describe several of theses attributes together with a theory for why we'd expect these to be useful.

First, we proposed that fax and voice lines may differ in the variety of numbers with which they place calls. For this reason, we collected data about the multiplicities of the various phone numbers called. Specifically, we computed the number of lines called exactly once, the number called exactly twice, etc., up to some limit. From these, we compute and report the number of lines called *at least* twice, at least three times, etc. (the *frequency tails*). We also compute the *frequency moments* for calls to lines. That is, suppose a_i is the number of times our example line called line i. We compute $m_j = \sum_i a_i^j$ for $j = 0, 1, 2, \ldots$, (but note that m_1 is just the total number of calls made by the example line). Together or separately, the first frequency tails or frequency moments give information about the multiplicities of calls in a flat file (fixed number of fields) representation—this way we captured some of the more important directed multigraph information in a useful format. We repeated the computation for inbound calls and combining inbound and outbound calls.

We also hypothesized that fax output is often misplaced so a user may fax a document several times to the same line, resulting in two or more calls to the same line of the same duration (to within a tolerance). Similarly, a server may fax a user a form that the user would fill out and fax back, resulting in two calls in opposite directions between a pair of numbers such that the second call is slightly longer than the first. We therefore computed frequency moments and tails for the lengths of calls just as we did for the destinations of calls, i.e., we let a_i be the number of calls of duration $i \pm \epsilon$ and computed frequency moments and tails on these a_i. We also computed frequency moments and tails for the joint frequency of destination and duration—the frequency of multiple calls of the same duration to the same destination line.

Ultimately, these attributes did not prove to be useful (so we don't fully describe them). We mention them here because they may be more useful in other, related applications or even in fax/voice differentiation efforts that have access to the outbound data that we lack. These attributes are also noteworthy for their algorithmic characteristics.

These attributes are called *historic* since, to update the attribute value v upon seeing call c, more data than v and c is needed.

Propagating Attributes

We also computed the number of voice lines and fax lines called by a given example line, as well as the corresponding values for lines two calls away. These attributes are called *propagating* since, in order to update value v upon seeing call c, one needs to know a classification for the line at the other end of c. Thus our knowledge (or lack thereof) about some lines propagates to other lines through consideration of these attributes.

For each line l, we looked at the set S_l of lines other than l that made or received any call with l and computed the fax:voice ratio among S_l (the first neighborhood statistics). We also looked at the set S_l^2 of lines other than l that made calls with S_l and computed the fax:voice ratio on S_l^2 (the second neighborhood statistics). There is substantially more second neighborhood data than first neighborhood data. Note that,

by definition, neither S_l nor S_l^2 contains l, but S_l and S_l^2 may overlap.

Cross Validation

We performed ten-fold cross validation. That is, we divided the examples into ten parts, for each $1 \leq i \leq 10$ we removed part i, trained on the other nine parts, then tested on part i. We report the average of the error rates on the test data.

Each training set naturally has more voice than fax examples. Before training we subsampled the voice training examples so that we ultimately train on a set with an equal number of voice and fax examples. This builds a hypothesis that does about as well on voice as on fax test data.

For both test and training examples, we computed the neighborhood statistics on neighbors in the training data only. That is, we recorded the sizes of the voice and fax neighborhoods restricted to the training data. For example, consider three fax lines, l_1, l_2, and l_3, each of which calls the other, such that l_1 is a test example and l_2 and l_3 are training examples. We would say that l_1 has two (undirected) fax neighbors and l_2 and l_3 have one fax neighbor each.

Algorithmic Considerations

Our data files (and the data files of future experiments) are large enough that disk I/O is a bottleneck. It is important, therefore, that we can usefully process entire disk blocks of data at once. Suppose there are $N = 1$G items to process and $B = 1$K items per disk block (so there are $N/B = 1$M blocks). We'd like an algorithm to make at most 1M $= N/B$ or even 30M $= (N/B) \log N \ll N =$ 1G I/O's. In practice, this means

- There is sufficient time to sort the data
- Sorting (i.e., grouping) the call data by line is desirable so that each record relevant to one or more given lines can be read, processed, and written in $O(1/B)$ amortized disk I/O's.

For further discussion, see [AV88].

Note that the strategy of grouping and streaming the data is sufficient to compute accumulative attributes.

To compute the several non-accumulative attributes we need to regroup the data. That is, we assume that the call data is grouped according to line number in our list of voice and fax lines. To compute the frequency moments for lines called by line l, for example, we sort the calls secondarily by called-line after which it is straightforward to compute the frequency moments efficiently. Following this, to compute the frequency moments for the durations of calls made by l, we need to resort the call data so that it remains sorted primarily by line but now becomes sorted secondarily by duration. Our algorithm only performs the necessary resorting. With this kind of preprocessing, it is also straightforward to compute our non-accumulative attributes statistics, assuming we have access to all the call data at once.

Our computation of non-accumulative attributes does not scale so well as the the computation of accumulative attributes since the former requires more than the current attribute value and a single call record to make

an update. Specifically, one cannot decide if a call made today is to the same destination as a call made in the last month without storing a month's worth of data. Similarly, one cannot decide if today's call to a known fax is to a *new* known fax without storing all of the previously-called fax lines. When we scale our algorithm to all fax lines in the country, for example, we may not have access to more than one or two day's data at a time and so to scale up we will have to modify our attibutes somewhat, for example, by considering the number of calls to known faxes rather than the number of known faxes called.

Learing Algorithms

We ran machine learning software on the file of profiles. We compared two learning algorithms of different types, that we now discuss.

C4.5

C4.5 is a program written by J. Quinlan [Qui93] that generates a decision tree classifier when given a set of labeled examples. The input to the program is a flat file of examples. All information about each example must be expressible in terms of a fixed collection of attributes. Each observation is also labeled by the name of the class containing it.

The output generated by the program is a decision tree in which each internal node specifies some test to be carried out on a single attribute value, and each leaf is labeled by a name of a class. A decision tree can be used to classify an example by starting at the root of the tree and moving through it until a leaf is encountered. At each nonleaf decision node, the example's outcome for the test at the node determines the child whose test should be carried out next. The class of the case is predicted to be the class associated with the leaf in which the process ends.

We now describe the C4.5 algorithm in detail. Assume, without loss of generality, that we have only two classes, namely *fax* and *voice*. For a set S of training cases, the entropy of S is $\mathrm{entropy}(S) = -(\mathrm{fr}(\mathrm{fax}, S) * \log_2(\mathrm{fr}(\mathrm{fax}, S)) + \mathrm{fr}(\mathrm{voice}, S) * \log_2(\mathrm{fr}(\mathrm{voice}, S)))$, where $\mathrm{fr}(\mathrm{fax}, S)$ is the fraction of the observations in S labeled fax, and similarly for voice. The infomation gain associated with partitioning S into S_1 and S_2 according to the outcome of a test T is

$$\mathrm{gain}(S, T) = \mathrm{entropy}(S) - ((|S_1|/|S|) * \mathrm{entropy}(S_1) + (|S_2|/|S|) * \mathrm{entropy}(S_2)).$$

The entropy associated with any split of S into S_1 and S_2 of the given sizes is defined as

$$\text{split-entropy}(S, T) = -(|S_1|/|S|) * \log_2(|S_1|/|S|) - (|S_2|/|S|) * \log_2(|S_2|/|S|).$$

C4.5 chooses for each node a test T of the form $X \leq Z$ where X is an attribute and Z is a threshold value, that maximizes the $\mathrm{gain\ ratio}(S, T) = \mathrm{gain}(S, T)/\text{split-entropy}(S, T)$ for the training set S associated with the node.

In a second stage C4.5 prunes its initial tree by replacing subtrees with one of their branches. Pruning tends to reduce overfitting thereby improving the classification on unseen examples.

Note that the bare bones C4.5 algorithm scales well, at least for continuous data such as we have. This is because for each node, in order to find a test T of the form $X_i \leq Z_i$, the algorithm needs to consider each of

d attributes X, sort the examples by X-value, then rescan the resulting list of examples to find the threshold that optimizes the information gain. Such sorting and scanning algorithms are I/O-efficient.

Hyperplanes

Given two sets of data $X \subseteq \mathbb{R}^d$ and $Y \subseteq \mathbb{R}^d$ (the vectors of attributes for voice and fax lines), we may regard the data as lying in a d-dimensional plane at unit distance from the origin in $\mathbb{R}^{d+1}$. We can then attempt to separate the vectors by a hyperplane P through the origin in $\mathbb{R}^{d+1}$. We measure the success of P by the number of misclassified points,

$$\mathrm{miscl}(X, Y, P) = |\{x \in X : \langle x, n_P \rangle \leq 0\}| + |\{y \in Y : \langle y, n_P \rangle \geq 0\}|,$$

where n_P is the normal vector of P.

The separation is *perfect* if $\mathrm{miscl}(X, Y, P) = 0$. The classical perceptron algorithm is a simple, efficient algorithm that finds a perfectly separating hyperplane iff exists [Ros58, Ros62, MP69]. If such a hyperplane does not exist, then finding a P such that $\mathrm{miscl}(X, Y, P)$ is minimal is NP-hard. Several algorithms in the literature try to find a hyperplane P such that miscl(X,Y,P) is as small number as possible [DH73, Bis95]. We use our own algorithm, which yields good results. The algorithm iteratively finds hyperplanes with increasing performance, such that each local optimization step is a version of line search ([Vap95]) in some random direction.

We create a sequence of hyperplanes $P_0, P_1, \ldots$ recursively, all going through the origin. We start with an arbitrary hyperplane P_0, which goes through the origin.

Construction of P_i from P_{i-1} :

We pick a random vector $r \in \{0, 1\}^d$, and we minimize $\mathrm{miscl}(X, Y, P_{i-1}(\lambda))$ over λ, where $P_{i-1}(\lambda)$ is defined as the hyperplane orthogonal to $n_{P_{i-1}} + \lambda r$. We then put $P_i = P_{i-1}(\lambda)$. Since $P_{i-1} = P_i(0)$, we have, that $\mathrm{miscl}(X, Y, P_i)$ is monotone-decreasing in i.

Algorithm to find $\min_\lambda \mathrm{miscl}(X, Y, P_i(\lambda))$:

1. For every $x \in X$ find λ_x such that $\langle x, n_{P_{i-1}} + \lambda_x r \rangle = 0$. For every $y \in Y$ find λ_y such that $\langle y, n_{P_{i-1}} + \lambda_y r \rangle = 0$. (For the analysis, we assume all the λ's are distinct).
2. Sort $\{\lambda_x \mid x \in X\} \cup \{\lambda_y \mid y \in Y\}$ to obtain a sequence $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{|X \cup Y|}$.
3. Let $I_0 = (-\infty, \lambda_1)$, $I_j = (\lambda_j, \lambda_{j+1})$ for $1 \leq j < |X \cup Y|$, and $I_{|X \cup Y|} = (\lambda_{|X \cup Y|}, \infty)$. Define $\mathrm{COUNT}(i)$ for $0 \leq i \leq |X \cup Y|$ recursively: $\mathrm{COUNT}(0) = 0$; $\mathrm{COUNT}(j) = \mathrm{COUNT}(j-1) + \epsilon_j$, where $\epsilon_j = 1$ if λ_j orginates from a λ_x, where $\langle x, r \rangle < 0$ or if λ_j orginates from a λ_y, where $\langle y, r \rangle > 0$, and $\epsilon_j = -1$ otherwise.
4. Find j which minimizes $\mathrm{COUNT}(j)$, and set $\lambda \in I_j$ arbitrarily.

The cost of an iteration of the hyperplane algorithm is at most that required to sort n values and to perform n dot products of vectors of length d, where n is the number of examples and d is the number of attributes. The algorithm can proceed by scanning and sorting the data, which is I/O-efficient.

Overall, the hyperplane and bare-bones C4.5 algorithms are comparable—roughly a sort and scan of all the data for each hyperplane iteration or each decision tree node. When applying these algorithms to larger datasets, one can limit the number of iterations/nodes either to conserve resources or because a limited number of iterations/nodes suffices, and so we expect both techniques to scale well. For further discussion of I/O-efficient learning algorithms, see [MAR96].

Results

We ran C4.5 and our hyperplane algorithm on various combinations of attributes. The results are given in Figure 1.

Figure 1: Results of C4.5 and the Hyperplane algorithm, in %-error, using combinations of attributes indicated by +.

	Attributes								C4.5			Hyperplane		
	n	us	nc	wk	n8	dur	tod	fm	voice	fax	total	voice	fax	total
1	+	+	+	+	+	+	+	+	10.5	12.0	10.6	8.8	12.5	9.2
2	+	+	+	+	+	+	+	-	10.1	11.9	10.4	8.8	13.2	9.3
3	+	+	+	+	+	+	-	+	9.2	11.4	9.5	8.1	10.5	8.4
4	+	+	+	+	+	-	+	+	14.0	17.4	14.4	11.2	29.6	13.5
5	+	+	+	+	-	+	+	+	10.1	11.7	10.3	9.5	12.7	9.9
6	+	+	+	-	+	+	+	+	10.8	11.7	10.9	9.1	14.1	9.7
7	+	+	-	+	+	+	+	+	10.0	12.3	10.3	9.7	13.6	10.1
8	+	-	+	+	+	+	+	+	10.0	11.7	10.1	8.8	13.0	9.3
9	-	+	+	+	+	+	+	+	12.5	13.4	12.6	13.2	15.4	13.3
10	+	-	-	-	-	+	-	-	10.3	12.0	10.5	8.0	14.0	8.8
11	+	-	-	-	-	-	-	-	14.8	22.2	15.8	13.0	28.3	15.1
12	-	-	-	-	-	+	-	-	13.0	16.5	13.4	10.6	16.9	11.4

Key:

- n: first and second neighborhood statistics, inbound and outbound (6 attributes)
- us: total usage, seconds
- nc: total number of calls
- wk: weekend versus weekday usage, inbound and outbound (4 attributes)
- n8: number of calls to toll-free numbers
- dur: call duration histogram (13 attributes)
- tod: call time-of-day histogram (24 attributes)
- fm: frequency moments and frequency tails (15 attributes)

We consider twelve experiments, each a combination of families of attributes. Experiment 1 consists of all attributes, experiments 2–9 consist of all attribute families but one, and experiments 10–12 consist of subsets of nearest-neighbor and duration attribute families.

First, note that experiment 3, in which the time-of-day information is omitted, does somewhat better than experiment 1 in which all attributes are present. This may be due to overfitting.

Of all attribute families, the nearest-neighbor and duration information seem to be the most important single families. First, we note that experiments 4 and 9 (that exclude these attributes) yield significantly worse results than experiment 1. Furthermore, experiment 10 (only the duration and neighbor information) does about as well as experiment 1 (except that the hyperplane algorithm suffers somewhat on faxes).

Note that the overall performance of C4.5 and the hyperplane algorithm are correlated, that is, attributes that are important for one algorithm tend to be important to the other algorithm.

Finally, we note that, for most subsets of the attributes, the algorithms give similar performance.

Conclusions

The C4.5 algorithm and our hyperplane algorithm have similar error rate in all our experiments despite the fact that they compute very different classifiers. This suggests that many learning algorithms would have comparable performance on our data. Clearly to establish this conjecture a more thorough experiment with more learning algorithms is required. (For example, we should try support vector machines [CV95]).

In contrast with the observation that the learning algorithm has little affect on the error rate we also observe that the particular choice of attributes seems to be critical. In our case duration and neighborhood information are the important attributes and if we omit either of them we do not see the low error rate we achieve by using both. Other attributes seem to contain less information. Thus, discovering the right attributes seems to be the main challenge in classification problems of this kind. (We note that the relative importance of attributes may be an artifact of our particular dataset and the loss of most outbound traffic data; other attributes may be more important when outbound traffic data is reliable.)

Adding attributes that do not contain information relevant to the classification may cause overfitting and thereby increase the error rate. It also slows down the training process and unnecessarily complicates the classification device. Therefore, a feature selection phase is likely to improve the results. In our experiments we performed this process manually but one can use automatic mechanisms for this purpose (See [Man97] for an automatic algorithm for feature reduction that fared well on the breast cancer database [WSM95, MA92]).

References

[AV88] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, September 1988.

[Bis95] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon, Oxford, 1995.

[CEWB97] K. C. Cox, S. G. Elick, G. J. Wills, and R. J. Brachman. Visual data mining: Recognizing telephone calling fraud. *Data Mining and Knowledge Discovery*, 1(2):225–231, 1997.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classifications and Scene Analysis*. John Wiley & Sons, New York, 1973.

[MA92] P. M. Murphy and E. W. Aha. UCI repository of machine learning databases. `http://www.ics.uci.edu/AI/ML/MLDBRepository.html`, 1992. Dept. Information and Computer Science, University of California, Irvine.

[Man97] O. L. Mangasarian. Mathematical programming in data mining. *Data Mining and Knowledge Discovery*, 1997.

[MAR96] M. Mehta, R. Agrawal, and J. Rissanen. SLIQ: A fast scalable classifier for data mining. In *Proc. of the Fifth Int'l Conference on Extending Database Technology*, March 1996.

[MP69] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

[Ros62] F. Rosenblatt. *Principles of Neurodynamics*. Spartan, New York, 1962.

[Vap95] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[WSM95] W. H. Wolberg, W. N. Street, and O. L. Mangasarian. Wisconsin prognostic breast cancer database. `ftp://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/WPBC/`, 1995. Computer Sciences Department, University of Wisconsin, Madison.