Feature Selection for Building Cost-Effective Data Stream Classifiers * (Extended Abstract)

Like Gao, X. Sean Wang
Department of Computer Science, University of Vermont, Vermont, USA
{*lgao, xywang*}*@cs.uvm.edu*

Abstract

A stream classifier is a decision model that assigns a class label to a data stream, based on its arriving data. Various features of the stream can be used in the classifier, each of which may have different relevance to the classification task and different cost in obtaining its value. As time passes by, some less costly features may become more relevant, but the time needed for decision may be considered as a cost. A challenge is how to balance the different costs when building a cost-effective classifier. This paper proposes a new feature selection strategy that extends the traditional Relief *algorithm in two aspects: (1) estimate the classification cost associated with each feature, and (2) order all the features with a score that combines both cost estimation and classification relevance. A classifier is then built with the selected features using a traditional classification method. Experimental results show that classifiers constructed with this strategy are indeed cost effective.*

1 Introduction

Data stream classification has been recognized useful in many applications, and interesting research results have appeared [1, 2, 6]. However, most reported research is concerned with "online" classification model (or classifier) building, in which training data takes the form of a data stream. In this paper, we assume that we can build a classifier in an "off-line" fashion. The classifier is used online to assign class labels to data streams based on their arriving data. This can be useful in situations where we need to distinguish stream sources by looking at the data. A specific problem we are interested in is the minimization of the cost associated with this classification task.

Generally, we may build classifiers by using different features of data streams. Some features may be more relevant to classification but more costly to acquire, while others may be less costly to obtain but less useful for classification. Although a similar cost-based classification problem has been studied [7], a unique property in the data stream situation is that some less relevant features may become more relevant with the passing of time. Therefore, the cost of using a feature may include the amount of time (called *classification time*) needed to reach a classification conclusion. We need to know how soon the less relevant features become useful, and whether it is worthwhile to trade classification time with feature acquisition cost.

To build a classifier with the least classification cost, we propose a feature selection method by extending original Relief [3]. Traditional feature selection methods [3, 4, 5], including Relief, have shown to be capable of removing irrelevant features and increasing classification accuracy, but they have not considered classification cost. Our proposed method, StreamRelief, includes the estimated classification cost associated with each feature. With features selected by StreamRelief, we can apply a traditional classification method to build the classifier.

To validate our approach, we use experiments to compare StreamRelief with three other methods. The first is a direct use of C4.5 on all the features, the second is a cost-based classification approach that is similar to [7], and the third is a direct use of Relief without considering costs. Results show the effectiveness of StreamRelief.

With this paper we make two contributions: Firstly we introduce the concept of classification cost for data streams, which we believe is a new and very important measure in data stream processing. Secondly, we develop a cost-based feature selection algorithm to choose features that may lead to a least costly classifier.

2 Problem Formulation

Definition *A data stream* is an infinite sequence $\langle v_1, v_2, \ldots \rangle$, each v_i, sampled at time position t_i, being a tuple on the attributes $(D_1, D_2, \ldots, D_n)$.

*Work partly supported by NSF grants IIS-0415023, IIS-0430402, and IIS-0430165.

This paper assumes that a classification system may use features of data streams to build classifiers. Compared to the raw data, feature values may be easier to obtain (e.g., a binary flag whether temperature is above or below a threshold), and/or require less energy to transmit in a distributed environment (e.g., send data only when its value has changed to certain extent in the last time interval).

Definition *A feature stream* is an infinite sequence $\langle e_1, e_2, \ldots \rangle$, each e_i being a tuple on the user-defined features $(F_1, F_2, \ldots, F_m)$, where for each F_j, there is a feature mapping function f_j (also called *feature extraction function*) that maps a data stream up to time t_i (inclusively) to feature F_j, that is, $f_j : (v_1, v_2, \ldots, v_i) \rightarrow e_i(F_j)$, for $i = 1, 2, \ldots$.

At each time t_i, a feature extraction function takes the prefix of a data stream up to t_i as input and generates a feature value.

Definition *A data stream classifier* is a classification model that takes a feature tuple e_i as input and returns a class label and its associated classification confidence.

A data stream classifier works as follows. Feature values obtained from the data stream are continuously fed into the classifier to make a decision, one feature tuple e_i a time. For each e_i (independent from earlier feature tuples), the classifier may conclude a class label together with a confidence value. When the conclusion has a sufficiently large confidence, the classification procedure ends and the last class label obtained is assigned to the data stream. In practice, this procedure may also end on user's request, e.g., in case a user-specified time limit is reached. In this case, no class label is assigned to the data stream. It is assumed that application users will provide a suitable confidence threshold and/or a time limit.

Definition *Data stream classification procedure* is to continuously apply a data stream classifier on a feature stream, until a user-specified confidence is satisfied or time out is reached.

There are two types of costs in obtaining feature F_j's values. One is the *one-time cost* spent on initializing the feature mapping function f_j, denoted C_{int_j}. The other is the *each-time cost* expended every time a feature value is extracted from the data stream, denoted C_{each_j}. This paper assumes that C_{each_j} does not change over time.

Definition *Classification cost*, C_{CDP}, is the total cost incurred until the classification procedure ends. Specifically, given a set of features F_j that is used by the classifier, the classification cost is a function of three kinds of variables: classification time t, one-time cost C_{int_j} and each-time cost C_{each_j}, given as:

$$C_{CDP} = a \cdot t \cdot \sum_j C_{each_j} + b \cdot \sum_j C_{int_j} \qquad (1)$$

where a and b are weights given by users.

Besides classification cost, two other measures are needed for performance assessment of a classifier.

Definition *Classification completeness* is the ratio of streams that are classified (i.e., assigned a class label with enough confidence before time expires). *Classification accuracy* is the ratio of streams that are correctly classified among all the streams being assigned labels.

3 Building Cost-Effective Classifiers

In this paper, we consider four practical methods to build classifiers. For illustration, we take decision tree (e.g., C4.5) building procedure as the example.
(1) Naive C4.5: directly use traditional decision tree construction. The rationale behind this is that, if we can find a classifier with higher predictive accuracy, then we may end the classification procedure earlier, and hence, minimize classification cost. This method tries to reduce the classification time (t) in Equation 1.
(2) Cost-Based C4.5: consider each-time cost only [7]. During the tree construction, we need to decide which feature should be used to split the training data set, e.g., based on information gain (or gain ratio). We may modify the split criterion to take each-time cost into consideration, i.e., to use the ratio of information gain and the each-time cost instead. Clearly, this method ignores the classification time, i.e., $t = 1$, and the one-time cost, i.e., $\sum_j C_{int_j} = 0$, in Equation 1.
(3) C4.5 with Relief: first choose a subset of features with Relief, the original feature selection algorithm, then apply C4.5 construction with the selected features only. This two-step method tries to build a classifier with a higher accuracy than Naive C4.5 does.
(4) C4.5 with StreamRelief: first choose a subset of features with StreamRelief, our proposed cost-based feature selection scheme, then apply C4.5 construction with the selected features only.

The major difference between methods (3) and (4) is that, in (3), Relief only considers feature relevance to classification, while in (4), StreamRelief takes both feature relevance and classification cost into account.

In the next, we will discuss Relief for data streams and present our StreamRelief algorithm, to complete the above methods (3) and (4), respectively.

3.1 The Relief Algorithm

The orignal Relief and its variants [5] compute a relevance weight for each feature in the dataset, as the estimation of its effectiveness in classification process.

Given a training data set, let R_i be an instance in the set, H_i be the near hit of R_i (the instance that is closest to R_i in the same class), and M_i be the near miss of R_i (the instance that is closest to R_i from other classes). If we randomly choose n instances from the set, then feature F_j's weight is given as [3]:

$$W[F_j] = \frac{\sum_{i=1}^{n}(\text{diff}(F_j, R_i, M_i) - \text{diff}(F_j, R_i, H_i))}{n} \quad (2)$$

In the above, diff(...) is a normalized difference function between feature values of two instances. For nominal features, the difference is value 0 (if two feature values are the same) or 1 (if different). For numeric features, the difference is normalized to interval [0,1].

In case of data streams, we may implement Relief as follows: (1) from the given data streams, randomly select some prefix sub-streams with their lengths randomly in a pre-determined range; (2) map each substream into a set of feature instances, one instance for each time position, and construct a training data set with all the feature instances; and (3) apply Relief to this training data set and get feature weights. Basically, the major extension is to generate a training data set from streams. This straightforward implementation is called Relief *for data streams*, or still Relief for short.

3.2 The StreamRelief Algorithm

Relief for data streams does not take classification cost into consideration. The following StreamRelief method extends Relief to include this cost.

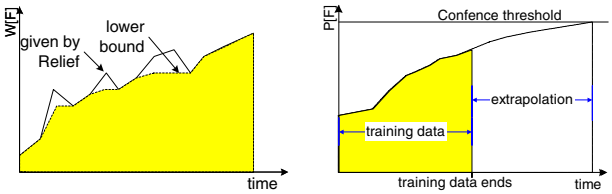
Let t_i ($i \geq 1$) be a discrete time position and assume the classification procedure starts at t_1. Let $P_i[F_j]$ be the probability that, with feature F_j, a data stream classifier gives a class label (with sufficient confidence) before t_i, inclusively. Then the expected classification cost associated with F_j (excluding one-time cost) is

$$C_{F_j} = C_{each_j} \sum_{i=1}^{\infty} (1 - P_i[F_j]) \quad (3)$$

and hence, C_{CDP} (Equation 1) becomes:

$$C_{CDP} = a \sum_j C_{F_j} + b \sum_j C_{int_j} \quad (4)$$

From these two equations, we can see that, in order to minimize classification cost C_{CDP}, we need to know each-time cost C_{each_j}, one-time cost C_{int_j}, and probability $P_i[F_j]$, for each feature F_j. Generally, C_{int_j} is given by the application system, or can be known by analyzing the corresponding feature extraction function. C_{each_j}, being assumed time invariant, can be estimated as the averaged cost in obtaining feature values. We now concentrate on estimating $P_i[F_j]$.



(a) (b)

Figure 1. (a) lower bound $W_i[F_j]$ and (b) extrapolate $P_i[F_j]$

The main idea is to use the relevance weight given by Relief to estimate $P_i[F_j]$. Intuitively, features having higher weights are more effective in classification, and a classifier using them is more likely to reach a decision quickly. We assume the probability that a classifier makes a decision is linear to feature weight, that is,

$$P_i[F_j] = \frac{1}{2}(W_i[F_j] + 1) \quad (5)$$

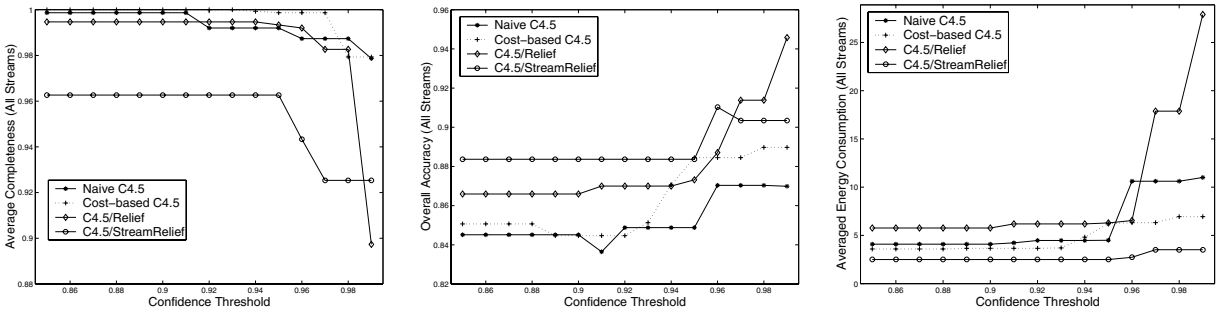
Two practical issues need to be addressed when estimating $P_i[F_j]$.

Firstly, $W_i[F_j]$ may decrease as time passes by (i increases). Intuitively, the longer the time, the clearer if a feature is relevant to classification. Hence, if feature F_j is indeed relevant, then $W_i[F_j]$ should monotonically nondecreasing as i increases. In contrast, if F_j is independent of classification, then its weight is more reliable at a later time: a high weight at the beginning is more likely to be an accident. For both cases, future weights of F_j should be considered when to determine the current. This paper uses the lower bound of all future weights to update the current one, i.e., $W_i[F_j] = min\{W_k[F_j] | i \leq k\}$, as shown in Figure 1(a).

Secondly, $P_i[F_j]$ may not reach the required confidence threshold due to limited training data, and we need to predict $P_i[F_j]$ in the future. We here use extrapolation techniques, e.g., linear, spline, Hermite, and other extrapolation methods, to speculate the training data until the estimated probability converges to the threshold, as illustrated in Figure 1(b).

In summary, the major steps of StreamRelief are:

1. Generate a training data set at each time t_i: apply feature mapping functions on streams for each time position before t_i and obtain a training data set.
2. Get weight of each feature at each time t_i: apply Relief to the training data set, and update $W_i[F_j]$ with its lower bound (Figure 1(a)).



(a) Completeness (b) Accuracy (c) Classification cost

Figure 2. Performance comparison.

3. Calculate the cost associated with each feature: use Equations 3 and 5 to get the estimated cost C_{F_j} for F_j. Use extrapolation (Figure 1(b)) if necessary.
4. Return the ordered list of features: order features based their cost impact on C_{CDP} (Equation 4).

4 Experimental Results

We use experiments with a same-source detection problem to evaluate the four classifier construction methods discussed in Section 3: (1) Naive C4.5, (2) Cost-based C4.5, (3) C4.5 with Relief, and (4) C4.5 with StreamRelief. The performance measures are classification completeness, accuracy and cost.

The same-source detection problem is to decide whether or not a pair of wireless sensors are sensing the same source. The sensor nodes are battery-powered, and the cost concerned is the energy consumption by them. For simplicity, we use the relative frequency of data transmission as the cost, since data transmission consumes the major energy. We omit the details due to length limit.

We set different confidence thresholds to control the classification procedure: a class label is assigned only if its associated confidence is no less than the threshold.

The experimental results are shown in Figure 2. From Figure 2(a) and (b), we can see that StreamRelief provides the same classification quality as the others. From Figure 2(c), we can observe that the energy consumption of StreamRelief is much smaller. As the confidence threshold goes higher, the energy consumption of StreamRelief just increases a little bit, while the others increase much faster.

Figure 2 implies that it may not be a good idea to use some very large confidence thresholds to control the classification procedure, since completeness may suffer significantly. We may find a suitable threshold through experimental study, e.g., in this experiment, 0.85 to 0.92 seems be a good range to choose the threshold.

5 Conclusions

This paper proposed a cost-based feature selection method, StreamRelief, to choose features that yield a classifier with a small classification cost. We validated this method using a same-source detection problem in sensor networks, and performed experiments to evaluate StreamRelief. The experimental results confirm the cost-effectiveness of StreamRelief.

References

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. On demand classification of data streams. In *ACM SIGKDD*, pages 503 – 508, 2004.

[2] Q. Ding, Q. Ding, and W. Perrizo. Decision tree classification of spatial data streams using Peano Count Trees. In *ACM SAC*, pages 413 – 417, 2002.

[3] K. Kira and L. Rendell. The feature selection problem: traditional methods and a new algorithm. In *National Conference on Artificial Intelligence*, pages 129–134, 1992.

[4] H. Liu, H. Motoda, and L. Yu. Feature selection with selective sampling. In *ICML*, pages 395–402, 2002.

[5] M. Robnik and I. Kononenko. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 52(2):23–69, 2003.

[6] W. N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *ACM SIGKDD*, pages 377–382, 2001.

[7] P. D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, (2):369 – 409, 1995.