
Machine Learning, 60, 97–115, 2005

Online Multiclass Learning with *k*-Way Limited Feedback and an Application to Utterance Classification

HIYAN ALSHAWI hiyan@google.com
Google, Inc., 21st Floor, 1440 Broadway, New York, NY 10018, USA*

Editors: Dan Roth and Pascale Fung

Abstract. This paper introduces a setting for multiclass online learning with limited feedback and its application to utterance classification. In this learning setting, a parameter k limits the number of choices presented for selection by the environment (e.g. by the user in the case of an interactive spoken system) during each trial of the online learning sequence. New versions of standard additive and multiplicative weight update algorithms for online learning are presented that are more suited to the limited feedback setting, while sharing the efficiency advantages of the standard ones. The algorithms are evaluated on an utterance classification task in two domains. In this utterance classification task, no training material for the domain is provided (for training the speech recognizer or classifier) prior to the start of online learning. We present experiments on the effect of varying k and the weight update algorithms on the learning curve for online utterance classification. In these experiments, the new online learning algorithms improve classification accuracy compared with the standard ones. The methods presented are directly relevant to applications such as building call routing systems that adapt from feedback rather than being trained in batch mode.

Keywords: online learning, limited feedback, utterance classification, call routing

1. Introduction

Data collection and manual labeling can be the most expensive components, in terms of human effort, of applying classification systems to real world problems. This is particularly true if the characteristics of the application (input features or classes or the way inputs are mapped to classes) change over time requiring further collection and training. Online learning algorithms hold the promise of systems that learn and adapt gradually during operation without distinct phases for collection, labeling, training, and operation.

The general setting adopted in the machine learning and pattern recognition fields for online classification tasks involves a sequence of trials. At each trial the "learner" receives an input, predicts the class of the input, and then receives the correct class from the "environment". The learner then uses this feedback from the environment to update its internal state with the aim of improving the chance of correctly classifying the input from the next trial in the sequence. There are variations on the basic setting such as the binary

*The work reported in this paper was carried out while the author was at AT&T Labs.

classification case in which the input belongs to one of two output classes, the multiclass case in which it belongs to exactly one of K possible classes, and the multiclass multilabel case in which inputs may belong to zero or more possible classes. In addition to classification, some of the same online algorithms have been adapted to perform regression tasks with real number output.

This idealized general setting is unrealistic in some learning situations since the correct class is not always available after making a prediction attempt. A good example of a situation with limited feedback from the environment is the one we consider in our experiments in this paper. Here the learner is an utterance classification system directing calls based on the caller's first utterance. After receiving the utterance, the system presents k (with $k \leq K$) possible options to the caller who selects the appropriate destination (if presented) from this limited set of options; presenting all possible options for selection would be too burdensome to the user. (In the special case of $k = 1$, the system can only ask the user a yes-no question corresponding to confirmation or disconfirmation of one of the possible K alternatives.) These considerations led us to investigate a "k-way" version of online classification in which feedback is limited by the parameter k.

Some of the simplest and most efficient, yet most successful, online learning algorithms are linear classifiers that maintain a set of weights. Each weight can be thought of informally as indicating the strength of association between an input feature and a target class. The weights are applied to the input for a trial, typically using a simple linear threshold decision rule, and are then updated in response to receiving the correct class for the input. One of the earliest schemes for updating weights was the simple additive one used in the perceptron algorithm for pattern recognition (Rosenblatt, 1958). A major improvement introduced by Littlestone (1988) was the use of a multiplicative weight update scheme. This algorithm, called Winnow, has the advantage of reducing the weights of irrelevant input features quickly toward zero, making the algorithm more effective when there are a large number of features but only few of them are important to the classification task. A number of variations of the Winnow algorithm have since been studied, both in terms of provable error bounds (Littlestone & Warmuth, 1994; Kivinen & Warmuth, 1997; Crammer & Singer, 2001; Mesterharm, 2002) and empirical performance on natural language processing tasks such as document categorization (Dagan, Karov, & Roth, 1997) and spelling correction (Golding & Roth, 1999). However, the error rates for both the additive and multiplicative algorithms are significantly higher when feedback is limited, especially for the important case of $k = 1$ (simple confirmation), as illustrated by the empirical results presented in this paper. This is the motivation for introducing modified versions of these algorithms that improve performance on our task under the k-way limited feedback setting.

Utterance classification has previously been applied to call routing using batch training (Gorin, Riccardi, & Wright, 1997; Carpenter & ChuCarroll, 1998). The work presented here can be thought of as trying to achieve a more extreme degree of automation compared with our previous work on utterance classification described in Alshawi (2003). In that work, one aspect of human effort used conventionally for the utterance classification task was eliminated, specifically, the manual effort of transcribing training audio files into text. This was done by automatically training an unsupervised phonotactic model from training utterances and building a phone string classifier from the output of this unsupervised

model. In the experiments presented here, we also eliminate the human effort of assigning class labels to training audio files; the system learns to classify based on feedback from users during system operation. In these experiments, we further assume that no domain-specific speech audio files are available prior to the start of online learning for training domain specific phonotactic or word n-gram models, so generic recognition models (both word and subword models) are used to provide utterance features for online learning. Overall then, the experiments investigate the suitability of a method for bootstrapping a call routing system without prior application-specific data collection, class labeling, or batch training.

In Section 2 we review the setting for online multiclass learning and define the k-way limited feedback version of this setting. Section 3 presents basic and modified versions of additive and multiplicative online algorithms. The spoken language call routing application using online utterance classification with feedback is explained in Section 4 together with the features from generic speech recognition output used by the classifiers. The setup for our online utterance classification experiments is described in Section 5, and the results of these experiments are presented in Section 6.

2. Online multiclass learning with k-way limited feedback

2.1. *Online multiclass learning*

The most basic setting for online learning studied in machine learning is a type of binary classification, i.e. learning a function to map an input instance x to a class $y \in \{\text{true}, \text{false}\}$. Under the online setting, learning proceeds in a sequence of *trials* which we will number with positive integers starting at 1. At each trial t, the "learner" first receives an input instance x_t to be classified and predicts the class of the instance to be y_t^1 based on its current state. (Here the superscript 1 simply indicates the class ranked highest by the learner.) The learner then receives the "correct" class assignment y_t for this instance from the "environment", and it uses this information to update its state. The updated state is used for predicting the class of x_{t+1} in the next trial of the sequence. Some algorithms used for this setting only update the learner's state when a prediction mistake is made, i.e. when $y_t^1 \neq y_t$, though in general the learner is not constrained in this way.

This setting extends naturally to *multiclass* classification, under which the number of classes K may be larger than 2. (Where the sequence of trials is infinite, the number of classes can be unbounded, though that case has not received much attention.) The online multiclass setting can be further extended to a *multilabel* setting in which each instance is classified (by the environment) into zero or more classes rather than a single class as in the basic multiclass setting. We will not be considering the multilabel case in the present work even though the techniques presented here can be modified to accommodate that setting as well. In other words, we will assume here that the environment provides exactly one class y_t for the input instance x_t of trial t.

Following the mistake-bounded online learning paradigm, the main way of evaluating how well an algorithm performs against a sequence of trials is in terms of maximizing the number of trials t in the entire sequence for which $y_t^1 = y_t$. (In the experiments, we will

also look at how well the algorithms perform on a fixed number of trials at the end of the sequence.)

2.2. *k-way limited feedback*

In this paper, we define *k-way limited feedback*, with $K \geq k > 0$, as an online multiclass learning setting in which each trial t proceeds as follows:

1. The learner receives a new input instance x_t.
2. The learner presents an ordered list of at most k classes, $y_t^1 \ldots y_t^k$ to the environment, y_t^1 being its prediction for classifying x_t.
3. The environment indicates which of the proposed classes corresponds to the correct classification y_t of x_t, or else explicitly indicates that none are correct.
4. The learner updates its internal state based on the information provided by the environment.

The restriction to presenting k classes in the second step makes learning under this setting more difficult than the standard setting: The learner does not always have access to the correct class for every trial since that class may not have been presented and therefore cannot be identified by the environment in the third step.

We will make an additional assumption that, in evaluating the performance of the learner, it is penalized as much for proposing the wrong class as for not proposing any classes. This assumption simplifies the setting and implies that there is no advantage to "passing" on a trial. (It might be interesting to consider algorithms for settings in which this is not the case, for example ones in which the learner is penalized less for passing than for an explicit error.)

Under this assumption, when $K = 2$ (and provided the learner knows there are only two classes), online classification with k-way feedback is essentially the same as online binary classification, since the environment's response will always unambiguously indicate the correct class for a trial. For $K - 1 > k$, compared with the standard online multiclass setting, the most important difference is that the learner does not necessarily find out the correct classification of the input for each trial. In the limiting case of $k = K$, multiclass k-way feedback is equivalent to the standard multiclass setting since the learner will always receive the same information for each trial under either setting. (Incidentally, the case $k = K - 1$ is equivalent to $k = K$ if K is fixed and known to the learner since the identity of the correct class will always be discernible even if none of the presented classes is indicated as correct by the environment.)

Online learning with k-way feedback is an appropriate setting for applications in which it is impractical or impossible for the environment to provide information on the correctness of the entire set of classes. For example, in the application we consider in the paper, k corresponds to the number of routing destinations a user is asked to select between during the operation of a call routing system that is trained online. Increasing k increases the burden on the user; high values of k are possible but impractical since this would make the system unusable. A similar example is a tune-guessing system in which the user hums a

tune and the system only has time to play a few (i.e. k) candidate recordings. Since both the set of available recordings and users' taste change over time, online learning is appropriate for this application.

2.3. Initial and boostrapping conditions

There are several different ways of providing the initial conditions for multiclass online learning. For example, one possibility (which we do not adopt) would be to assume that all the categories and features the system will encounter are known in advance. However, since the aim of online learning is to adapt automatically, there are situations in which this assumption would be too inflexible: an online text categorizer might encounter new words, or new categories could be added during the lifetime of the system.

To keep the learning setting very general and true to the spirit of online learning, we adopt the following protocol for bootstrapping the k-way online learner. Prior to the first trial in the sequence, the learner does not have any knowledge about the feature set. An open ended set of features is assumed and features become known to the learner as it encounters new inputs (as with the infinite attribute space of Blum, 1992). The set of target classes known to the learner prior to the first trial is also empty. Then, whenever the correct class for a trial is not in the set of classes known to the learner, that trial is treated as a conventional (not k-way limited) online learning trial: the environment provides this class to the learner even though it will not (in fact cannot) be one of the k classes proposed by the learner in this trial. We can think of such trials as "definitional" trials. To illustrate, in the text categorization example, a definitional trial could be inserted into the stream of trials by an operator who wishes to introduce a new category to the system.

3. Online classification algorithms

3.1. Prediction function

To predict the class for trial t, the algorithms discussed in this paper construct a function:

$$f_t : (x, y) \mapsto r$$

where r is a real number. Since we do not assume that the set of possible classes is known in advance of the trials, f_t will in general be a partial function on the cross product of possible inputs and classes. At trial t, the algorithm will predict class y_t^1 as

$$y_t^1 = \operatorname*{argmax}_y f_t(x_t, y)$$

where the maximization is over classes encountered so far. Similarly, y_t^2 is the class with the second highest value of $f_t(x_t, .)$, and so on.

The function f_t will be computed as a sum of real-valued feature weights which the algorithms maintain (implicitly or explicitly). More specifically, let $w_t^{y,a}$ be the weight

associated with feature (or attribute) a and class y prior to trial t, and let s_a be a function from input instances to non-negative real numbers representing the value ("strength") of feature a in the instance. We will assume that for any input x, the number of features for which $s_a(x) > 0$ is finite. f_t will be computed as follows:

$$f_t(x_t, y) = \sum_a w_t^{y,a} s_a(x_t).$$

Thus, assuming the feature functions s_a are fixed, as we take them to be here, the task of this class of online learning algorithm is simply to compute the weights $w_{t+1}^{y,a}$ given $w_t^{y,a}$ and the outcome of trial t.

3.2. *Contingency accumulators*

In most accounts of online algorithms with feature weight updates, the set of weights $w_t^{y,a}$ are described (if this level of description matters to the discussion) as being maintained explicitly as a set of real valued quantities. The stored weights $w_t^{y,a}$ are used to compute the weights $w_{t+1}^{y,a}$ for the next trial according to the outcome of trial t, and the specific update rule, and the new weights then replace the old ones in the storage data structure. However, for both numerical computation reasons as well as increased flexibility in the choice of update algorithms, we prefer to treat the weights as functions defined on "contingency accumulators" related to trial outcomes.

The contingency accumulators keep track of the (strength-weighted) occurrences of a feature a and a class y when y is the prediction (the first in the list of proposals) and is correct, y is the prediction but is incorrect, or when y is the "true" class but is one of the second to k-th proposals made by the learner. Specifically, the accumulators are defined as follows, where t' ranges over the trials so far:

$$\begin{aligned}
p_t^{y,a} &= \sum_{1 \le t' \le t} \delta(y, y_{t'}) \delta\left(y, y_{t'}^1\right) s_a(x_{t'}) \\
\tilde{p}_t^{y,a} &= \sum_{1 \le t' \le t} \tilde{\delta}(y, y_{t'}) \delta\left(y, y_{t'}^1\right) s_a(x_{t'}) \\
\tilde{n}_t^{y,a} &= \sum_{1 \le t' \le t} \delta(y, y_{t'}) \delta\left(y, \left\{y_{t'}^2, \ldots, y_{t'}^k\right\}\right) s_a(x_{t'}).
\end{aligned}$$

Here $\delta(y, z)$ is 1 if $y = z$ or $y \in z$ and 0 otherwise, and $\tilde{\delta}(y, z)$ is 1 if $\delta(y, z)$ is 0 and 0 otherwise. To reiterate, with limited feedback, the identity of $y_{t'}$ will not be known for some trials. Informally, we can think of the accumulators $p_t^{y,a}$, $\tilde{p}_t^{y,a}$, and $\tilde{n}_t^{y,a}$, as the accumulated weight of a for which y is, respectively, a true positive, a false positive, and a false negative. The factor $\delta(y, \{y_{t'}^2, \ldots, y_{t'}^k\})$, for the false negative case, reflects the fact that the learner does not always have access to the "true" class for every trial since only k classes are presented. When $k = K$, i.e. the full set of classes, $\tilde{n}_t^{y,a}$ simplifies to:

$$\sum_{1 \le t' \le t} \delta(y, y_{t'}) \tilde{\delta}\left(y, y_{t'}^1\right) s_a(x_{t'}).$$

It should be clear that these three accumulators can be maintained incrementally after each trial based only on the features in the input for the trial, the classes proposed in the trial, and the outcome of the trial. (There is no iteration over the entire set of possible features, or even over classes not included in the k proposed classes as there would be for a true negative accumulator $n_t^{y,a}$ defined in a parallel way.) If the feature strength functions are integers, as they are in the experiments reported below, then the accumulators will be integers. In the general case, however, these accumulators will be real valued quantities.

3.3. Weight computation

As mentioned, the online classification algorithms compute the prediction function $f_t(x_t, y)$ as a sum of weights $w_t^{y,a}$. We compute these weights from accumulator values on an as needed basis as required for computing the prediction function. As the last step of each trial, the accumulator values relevant to the trial are updated according to the outcome of the trial. To complete the description of the different online algorithms, it only remains to describe the specific method used by each algorithm to compute the weights $w_t^{y,a}$ from the accumulator values. We provide this for four algorithms used in our experiments: Additive, Multiplicative, Modified Additive, and Modified Multiplicative. For all algorithms, if all the accumulators for a class-feature pair (y, a) are zero, then the weight $w^{y,a}$ is taken to be zero.

Additive

$$w_{t+1}^{y,a} = \tilde{n}_t^{y,a} - \tilde{p}_t^{y,a}$$

This is a multiclass version of the well-known Perceptron algorithm (Rosenblatt, 1958). The weight increases when y is a false negative with respect to a trial and decreases when y is a false positive with respect to a trial. One way to think of this algorithm is that it is trying to set the weights so that false positive errors are balanced by false negative errors. In particular, if x_t is correctly classified, $w_{t+1}^{y,a}$ will be the same as $w_t^{y,a}$.

Multiplicative

$$w_{t+1}^{y,a} = \alpha^{\tilde{n}_t^{y,a} - \tilde{p}_t^{y,a}}$$

This algorithm is parametrized by a real constant $\alpha > 1$ which can be used to control the learning rate. It is a multiclass version of the Winnow algorithm (Littlestone, 1988). The weight for class y in the presence of feature a is increased by multiplying it by α when y is a false negative with respect to the trial. Conversely, the weight is reduced by multiplying it by $1/\alpha$ when y is a false positive with respect to the trial. In the Winnow algorithm, a different constant $0 < \beta < 1$ may be used when reducing weights, so here we are fixing $\beta = 1/\alpha$, giving the formula above for the weight computation. (Empirically, we have observed that lifting the restriction $\beta = 1/\alpha$ does not necessarily

lead to improved performance, and brings the additional difficulty of choosing β.) The motivation for the Winnow algorithm is to quickly push down the weights of irrelevant features toward zero: if a feature is irrelevant to a class but its weight is too high, this leads to false positive predictions and hence (repeated) multiplication of the feature weight by $1/\alpha$.

Modified additive

$$w_{t+1}^{y,a} = \tilde{n}_t^{y,a} - \tilde{p}_t^{y,a} + \frac{1}{k} \min\left(p_t^{y,a}, \tilde{p}_t^{y,a}\right)$$

The inclusion of the last term in the weight expression, $\frac{1}{k} \min(p_t^{y,a}, \tilde{p}_t^{y,a})$, is intended to overcome a problem that arises for the simple additive algorithm for low values of k. To motivate the term, consider the case $k = 1$ (simple confirmation) for which $\tilde{n}_t^{y,a}$ will be zero and the weight computation becomes simply $p_t^{y,a} - \tilde{p}_t^{y,a}$ (or 0 if $p_t^{y,a} > \tilde{p}_t^{y,a}$). In other words, for $k = 1$ we are using the true positive accumulators to make up for not being able to make use of false negative feedback to balance false positive feedback. As k increases, false negative feedback becomes more likely, so the need for using the true positive accumulators decreases, hence the $1/k$ coefficient. Since we would like to avoid making large adjustments to the weights when there are no errors, we introduce the minimization over true and false positives to prevent true positives from continuing to push the weights upward even if they vastly outnumber false positives. As k increases (and so long as y appears in conjunction with a as a false positive less frequently than as a true positive), the modified additive algorithm approaches the simple additive algorithm.

Modified multiplicative

$$w_{t+1}^{y,a} = \alpha^{\tilde{n}_t^{y,a} - \tilde{p}_t^{y,a} + \frac{1}{k} \min(p_t^{y,a}, \tilde{p}_t^{y,a})}$$

Compared with the simple multiplicative algorithm, the additional term in the exponent is motivated by the same considerations as it is for the modified additive algorithm: It compensates for the reduction in negative feedback when k is small compared to K, while still limiting undesirable increases in the weights when there are few errors in predicting y in the presence of feature a. As k increases, the modified multiplicative weight computation approaches that of the simple multiplicative algorithm provided that $\tilde{p}_t^{y,a} < p_t^{y,a}$. In particular, weights are not adjusted when this condition holds for all accumulators for the features in x_t and the learner correctly classifies x_t.

3.4. Computational complexity

All four algorithms have low runtime computational complexity. Let N be the number of features in an input x. Assuming that it takes constant time to store or retrieve an accumulator value, the number of arithmetic operations needed to process a trial after feature extraction is $O(K(N + \log(K)))$. (Feature extraction, will, of course, depend on the nature of the input and the complexity of the features used.) To see this, recall that computing $f(x, y)$ requires

the sum of $O(N)$ weights. For the prediction step, we need to compute f for all K classes. Since computing each weight requires only a constant number of arithmetic operations, this takes $O(KN)$ operations. (We are assuming that exponentiation is a primitive operation for this purpose.) We also need to identify the k highest ranked classes for presentation to the environment; this can done by sorting with $O(K \log(K))$ comparisons. In the accumulator update step, we only need to update $O(N)$ accumulators for the prediction and/or for the correct class, each such update requiring a constant number of arithmetic operations, giving $O(N)$ arithmetic operations. This gives the total number of arithmetic operations for a trial as $O(KN) + O(K \log(K)) + O(N)$ which is the same as $O(K(N + \log(K)))$.

3.5. Remarks on error bounds

The reader may be wondering about formal bounds on the classification error of the update algorithms under the limited feedback setting. To date, we have only investigated these algorithms empirically and are unaware of error bounds for the algorithms, though we now point out some properties of the algorithms that preclude immediate transfer of known bounds for batch and online classification algorithms. First, unlike batch learning algorithms, in the online setting of most interest to us, the update algorithms are aimed at applications for which the target function is not assumed to be fixed. The prediction function is meant to adapt in order to track changes in the target function over time, rather than converging to a fixed prediction function that approximates a fixed target as closely as possible. Specifically, we do not assume that the example input-output pairs (x, y) in our trials are generated i.i.d. at random from a fixed underlying distribution. In the online setting, error bounds have been derived, under various assumptions, initially for the binary case (e.g. Littlestone, 1988) and later for the multiclass case (e.g. Crammer & Singer, 2001). The analysis of the classic binary Winnow algorithm (with $K = k = 2$ in our notation) makes use of the so called conservative property of that algorithm, i.e. weights are adjusted only for trials in which errors are made by the learner. In contrast, our modified algorithms are not conservative in this sense since the weights depend on the values of true positive accumulators, a modification that our experiments (Section 6) show to be beneficial in the call routing application. Nevertheless, there is a growing understanding of the formal properties of online algorithms and their connection to other types of learning. Particularly promising in this regard is the analysis of the "drifting games" framework by Schapire (2001), which subsumes some well known error bound results for boosting and on-line learning.

4. Online utterance classification

4.1. Call routing application

In the application we focus on in this paper, automated call routing, online learning with k-way feedback is a natural fit: the learner is the call routing system and the environment is the user placing the call. The first step in the trial corresponds to the user speaking

an utterance expressing their desired action (typically transfer to a call destination). The utterance is processed by generic speech recognition models (i.e. ones not trained on this application), and the resulting recognition output is passed to the learner. In the second step, the learner identifies the k actions which it takes to be most likely to correspond to the user's request. The step of presenting the k classes to the environment corresponds to asking the user to clarify the desired routing action by selecting one of k actions, it being too burdensome to the user to select from a list consisting of all possible actions. The case of $k = 1$ corresponds to asking the user to confirm the system's best guess of the correct destination.

4.2. Utterance features

For a truly online setting, we are constrained to not using training material for the specific application domain prior to running the sequence of online training trials. Applying this principle to online utterance classification, we do not use any domain specific speech audio files for the domain prior to the online training run, and no human transcriptions of the speech files are used at any stage. (A less extreme alternative would be to make some audio files available for recognition training and only learn the mapping between domain-specific recognition output and call destinations.)

Using a generic speech recognizer increases the chance that an utterance will include words outside the recognizer's vocabulary, or ones that appear in n-gram contexts that the recognizer deems improbable, leading to word recognition errors. The features we provide to the learner are therefore a mixture of word and phoneme n-grams. These features are derived from the output of two recognizers applied to an input utterance:

- A word trigram recognizer for spontaneous telephone speech for which the acoustic and language models were trained on data from domains different from the online trials domain. The features derived from the best-path recognition output were word unigrams, word bigrams, and gapped ngrams of length two (i.e. trigrams with the middle word missing).
- A phonetic string recognizer for spontaneous telephone speech, again trained on domains different from the online trials domain. The "language model" for this recognizer is a 5-gram phone model; the phone sequences used to train this model were produced from transcriptions using the AT&T text-to-speech system. The features derived from the best-path output of this recognizer were phone bigrams, phone trigrams, and phone 4-grams.

Both recognizers use an acoustic model with discriminatively trained 3-state HMMs with 10 Gaussians per state similar to the models described in Ljolje et al. (2000). The word trigram model and phone 5-gram language models in these recognizers were constructed using the stochastic language modeling techniques described by Riccardi, Pieraccini, and Bocchieri (1996).

Although only feature counts were used as feature strengths, it is possible that better classification accuracy could result from continuously weighted features, including those taken from weighted word or phone lattices.

5. Setup of experiments

We would like to measure the accuracy and learning rates of k-way feedback online learning algorithms for utterance classification on real data. To do so we make use of utterances collected from a large number of callers accessing existing spoken language systems in two domains. We will only classify the first user utterance from each call placed to the system. These initial user turns are in response to a general prompt such as "Welcome to . . . How may I help you?". The spoken language systems from which these utterances were collected use batch classification methods and specific interaction strategies for processing the initial utterance and subsequent utterances in the interaction. We need not be concerned with these methods and strategies—using the first utterances only in our experiments means that the results reported here are independent of the techniques used in the original spoken language systems.

The collected utterances each have associated with them a single action or routing destination assigned by a human labeler. This association between the utterances and actions provides us with the experimental data to evaluate the accuracy and learning rates of our algorithms as well as allowing us to simulate the "environment" in our experiments. The sequence of utterances input to the learner are ordered chronologically (by collection date), providing a more realistic setting for an online system.

For both the Multiplicative and Modified Multiplicative algorithms, the learning rate α was set to 1.2, a setting found to give reasonable results in informal experimentation with data from a third domain.

We will refer to the two domains for the online utterance experiments as Domain M and Domain T:

- Domain M. The online sequence for Domain M consisted of 11930 trials. The average length for the 11930 utterances was 9.5 words. There were 89 target actions. As an indication of drift in the target function, the KL divergence between the class distributions for the first 2000 trials and for the last 2000 trials was 0.23. The word trigram model and phone 5-gram model used for recognition of Domain M utterances were trained on 257024 utterances (3574223 words) from other domains. These out of domain training utterances included utterances from both human-human and human-machine conversations. Recognition accuracy was 64% for words and 68% for phones.
- Domain T. The online sequence for Domain T consisted of 13902 trials. The average length of input utterances in these trials was 7.8 words. There were 59 target actions. The KL divergence between the class distributions for the first 2000 trials and for the last 2000 trials was 0.21. The word trigram model and phone 5-gram model used for recognition of Domain T utterances were trained on 318407 utterances (4175679 words) from other domains. Recognition accuracy was 54% for words and 66% for phones.

Table 1. Classification accuracy (%) over entire learning sequence for Domain M.

	$k = 1$	$k = 2$	$k = 3$	$k = K$
Additive	8.05	30.7	35.8	57.7
Modified additive	39.0	40.7	43.6	57.8
Multiplicative	16.6	44.3	49.7	60.0
Modified multiplicative	43.9	53.7	55.4	59.9

Table 2. Classification accuracy (%) over entire learning sequence for Domain T.

	$k = 1$	$k = 2$	$k = 3$	$k = K$
Additive	31.9	51.0	54.7	68.7
Modified additive	62.1	59.0	59.1	68.9
Multiplicative	31.2	57.8	60.4	69.8
Modified multiplicative	65.6	65.1	62.5	70.3

Table 3. Classification accuracy (%) for last 2000 trials of learning sequence for Domain M.

	$k = 1$	$k = 2$	$k = 3$	$k = K$
Additive	13.4	35.1	41.1	61.9
Modified additive	49.0	51.6	51.3	62.2
Multiplicative	14.4	47.6	53.6	62.3
Modified multiplicative	52.1	57.8	59.4	62.8

6. Results of experiments

Table 1 shows the classification accuracy of the algorithms for domain M for the entire learning sequence with k set to 1, 2, 3, and K. (Recall that K is the number of possible classes for the domain.) Table 2 shows the same results for Domain T.

Table 3 shows the classification accuracy for different algorithms for domain M for the last 2000 trials of the learning sequence. Compared with Table 1, Table 3 gives an impression of the attained accuracy of the algorithms with the benefit of most of the learning sequence, though the accuracy is still increasing near the end of the sequence for some of the algorithms. Table 4 gives the corresponding results for Domain T.

To give a better impression of how quickly the different algorithms are learning to classify utterances, we present cumulative learning curves for the different algorithms for $k = K$ for Domain M in Figure 1 and Domain T in Figure 2. These results are cumulative in the sense that each point in the plot shows (on the vertical axis) the accuracy on all prior trials after processing the percentage of the trials shown on the horizontal axis. (The points on the plot are at 1% increments of the number of trials processed; they do not represent individual trials.) The accuracy attained at the end of the curves thus corresponds to the results given

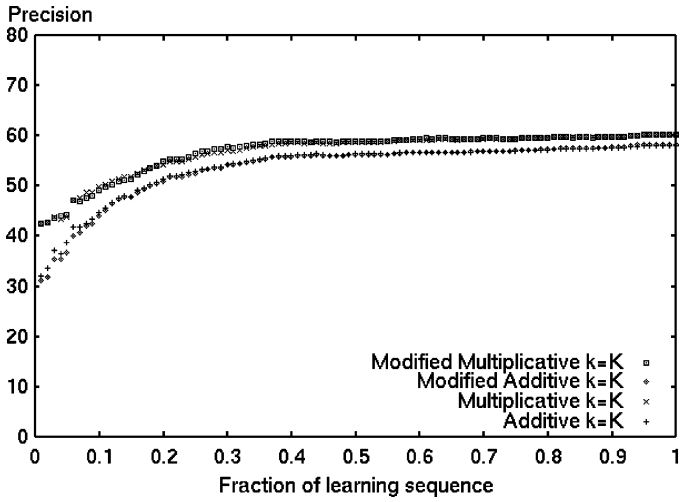

Figure 1. Cumulative learning curves for Domain M with $k = K$.

in Tables 1 and 2 rather than the higher accuracies shown in Tables 3 and 4 for the last 2000 trials of the sequences. Figures 3, 5, and 7 show the cumulative learning curves for Domain M when k is set to 1, 2, and 3, respectively. The corresponding cumulative learning curves for Domain T are shown in Figures 4, 6, and 8.

A number of patterns emerge from these results which are broadly true for both domains but seem to be more pronounced for Domain M which appears to be the more difficult task. For $k = K$, i.e. when feedback is not limited, the multiplicative algorithms have a slight edge over the additive algorithms. This observed difference between the additive and multiplicative algorithms is consistent with the conclusions of Kivinen and Warmuth (1997) since a large number of the input features are not relevant to our classification task. It also appears that for $k = K$ there is little difference in accuracy between the basic and modified versions of the additive algorithm or between the basic and modified versions of the multiplicative algorithm.

As can be expected, the algorithms are less accurate for the limited feedback settings of k than for the more standard full-feedback setting of $K = k$. Differences between the algorithms increase for lower values of k. In particular, for $k = 1$ the basic additive and basic multiplicative algorithms perform poorly on the entire sequence as well as on the

Table 4. Classification accuracy (%) for last 2000 trials of learning sequence for Domain T.

	$k = 1$	$k = 2$	$k = 3$	$k = K$
Additive	43.8	57.9	58.9	73.2
Modified additive	67.6	64.4	63.9	73.2
Multiplicative	24.6	64.6	6855	73.8
Modified multiplicative	67.8	66.9	70.7	74.0

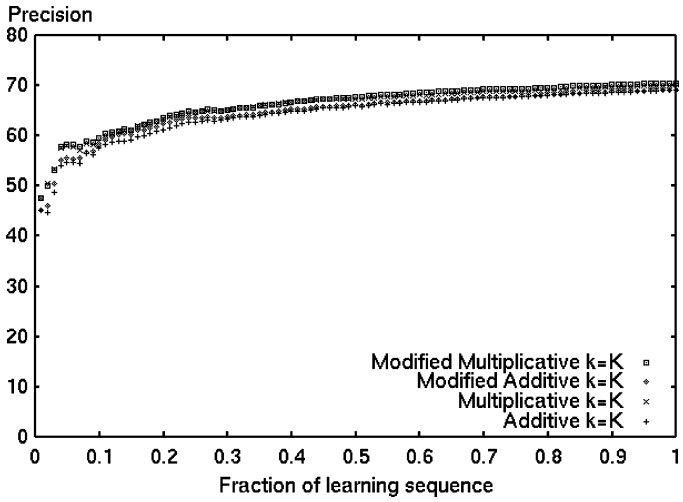

Figure 2. Cumulative learning curves for Domain T with $k = K$.

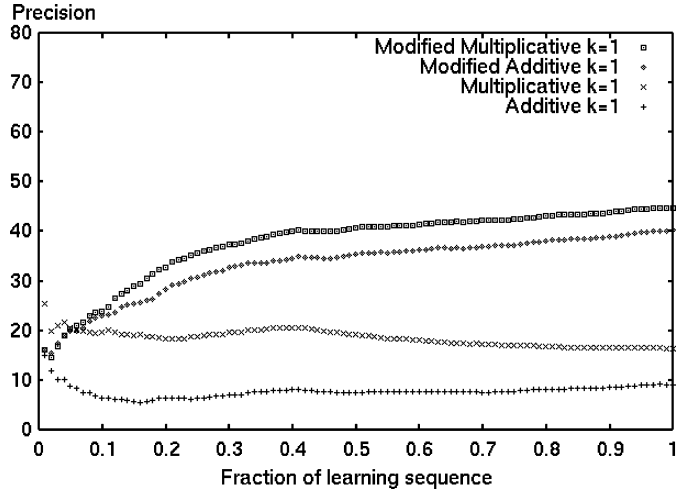

Figure 3. Cumulative learning curves for Domain M with $k = 1$.

last 2000 trials. In general, the multiplicative versions of the algorithms perform better than their additive counterparts (except for Domain T with $k = 1$). More importantly for the conclusions of this paper, for all three limited feedback settings ($k = 1, 2, 3$), the modified additive algorithm outperforms the simple additive algorithm and the modified multiplicative algorithm outperforms the simple multiplicative algorithm. As expected, these effects are stronger for $k = 1$ than for $k = 2$ and $k = 3$.

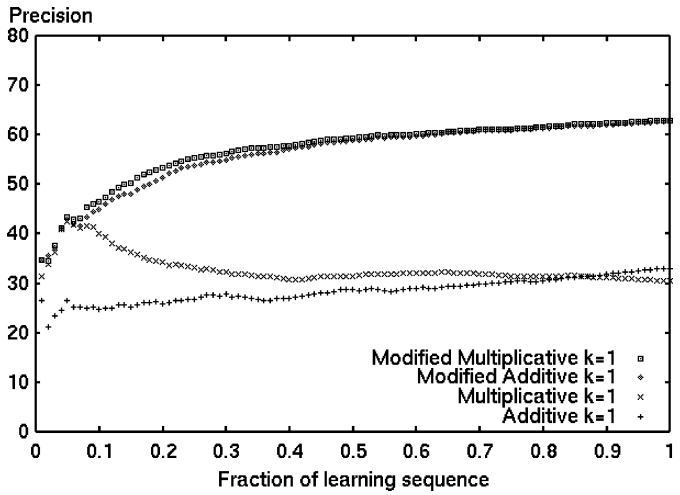

Figure 4. Cumulative learning curves for Domain T with $k = 1$.

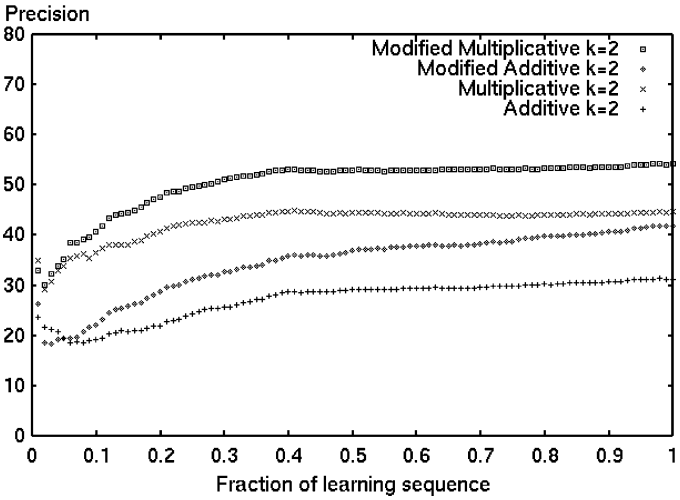

Figure 5. Cumulative learning curves for Domain M with $k = 2$.

On a practical note, in the most common current applications of utterance classification, such as customer care call routing, a confidence measure is used to reject the predicted classification if the confidence value falls below a threshold. When this happens, the user is asked to simplify the phrasing of their request or is passed to a human operator. In the current setting, one confidence measure is the difference between the values of the prediction function for the two highest ranked classes, normalized by dividing by the number of features in the input. (This normalization is necessary so that a single threshold

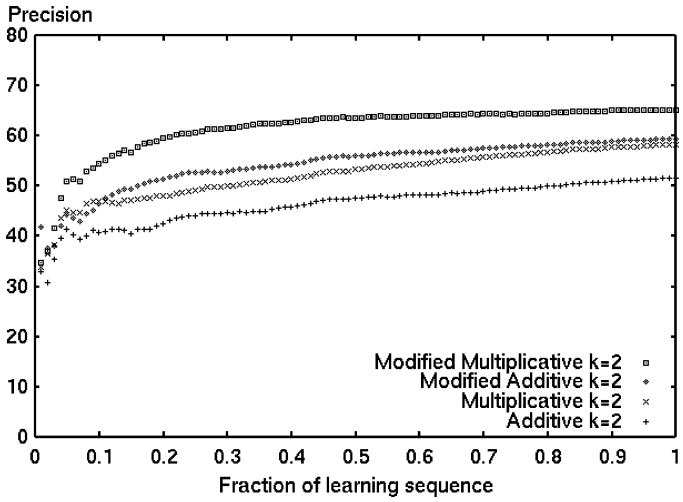

Figure 6. Cumulative learning curves for Domain T with $k = 2$.

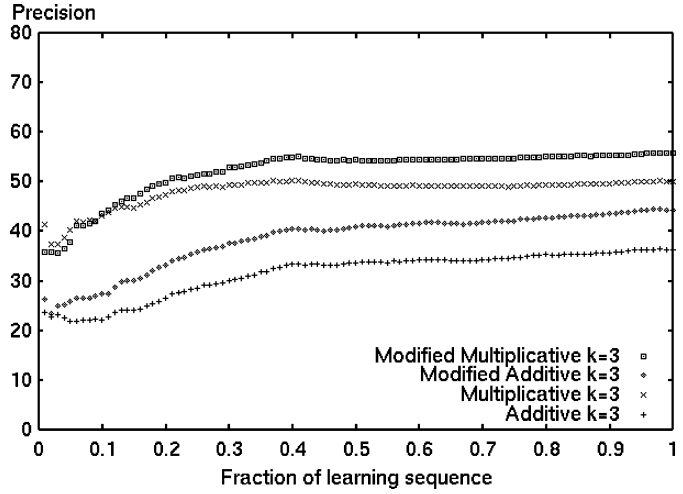

Figure 7 Cumulative learning curves for Domain M with $k = 3$.

is applicable across utterances of different lengths.) If, for example, the threshold is set so that 30% of the inputs are rejected, then the classification accuracy for the algorithms on the remaining 70% of the last 2000 utterances is as shown in Table 5 for domain M and Table 6 for Domain T. As might be expected, the accuracy figures are generally higher than when rejection is not in play, suggesting that the algorithms can be used in conjunction with confidence thresholds when this is appropriate to the application.

Table 5. Classification accuracy with 30% rejection for last 2000 trials, Domain M.

	$k = 1$	$k = 2$	$k = 3$	$k = K$
Additive	17.1	49.1	56.4	77.7
Modified additive	66.1	70.6	69.1	78.0
Multiplicative	16.7	61.3	68.6	78.7
Modified multiplicative	69.6	75.3	76.3	78.2

Table 6. Classification accuracy with 30% rejection for last 2000 trials, Domain T.

	$k = 1$	$k = 2$	$k = 3$	$k = K$
Additive	58.9	79.4	81.2	90.8
Modified additive	86.6	87.9	86.4	91.8
Multiplicative	28.1	83.5	86.5	90.5
Modified multiplicative	87.7	88.4	90.3	90.3

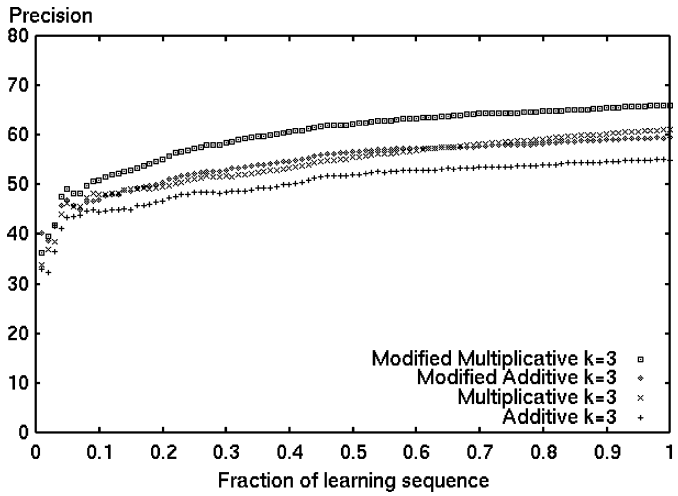

Figure 8. Cumulative learning curves for Domain T with $k = 3$.

7. Conclusions and future directions

We have introduced a limited feedback setting for multiclass online classification where the amount of feedback is controlled by a parameter k which limits the number of classes available for possible confirmation after each trial. This setting is effectively equivalent to the standard one when $k = K$, the size of the full set of classes. With k-way limited feedback, well known additive and multiplicative update algorithms for online linear classifiers perform poorly for low values of k. This is particularly true in the important case of $k = 1$

for which false negative information is not available to the learner. We therefore introduced modified versions of these algorithms aimed at overcoming this problem in applying the basic update algorithms to our setting. The modified algorithms, like the standard ones, have low computational complexity: after feature extraction, processing each trial requires only $O(K(N + \log(K)))$ arithmetic operations, where N is the number of features in the input for the trial.

We applied the basic and modified algorithms to utterance classification for call routing in two domains. Generic (domain independent) speech recognizers were used to produce word n-gram and phone n-gram features from input utterances, thus avoiding the need to have domain-specific speech data available prior to the online learning trials. The results of these experiments showed that the modified algorithms give significant classification accuracy improvements over the basic algorithms for low values of k without reducing accuracy when k is not limited.

Work in progress includes extending limited feedback online multiclass algorithms to handling the multiclass multilabel setting. Derivation of non-trivial formal error bounds for the algorithms still remains to done, and it would be interesting to investigate k-way feedback versions of the more complex additive online algorithms proposed by Crammer and Singer (2001). Another avenue worth exploration might be the application of k-way online algorithms to speech recognition lattices in conjunction with complex kernel functions such as the rational kernels defined by Cortes, Haffner, and Mohri (2003).

We believe that the empirical results presented here demonstrate the feasibility of efficient utterance classification systems that do not make use of batch data collection or class labeling, but instead learn to classify utterances from user feedback. This is made possible by the combination of domain-independent speech recognition with new limited feedback online learning algorithms.

Acknowledgments

Thanks are due to three anonymous reviewers for detailed comments which helped improve the paper.

References

Alshawi, H. (2003). Effective utterance classification with unsupervised phonotactic models. In *Proc. of the 2003 NAACL-Human Language Technology Conference*. Edmonton, Canada: ACL.

Blum, A. Learning Boolean functions in an infinite attribute space. *Machine Learning*, *9*, 373–386.

Carpenter, R., & Chu-Carroll, J. (1998). Natural language call routing: A robust, self-organizing approach. In *Proc. of the International Conference on Speech and Language Processing*, Sydney, Australia.

Cohen, W. W., & Singer, Y. (1996). Context sensitive learning methods for text categorization. In *Proc. of the 19th Annual International ACM Conference on Research and Development in Information Retrieval*.

Cortes, C., Haffner, P., & Mohri, M. (2003). Positive definite rational kernels In *Proc. Annual Conference on Computational Learning Theory*, LNCS 2777, Springer.

Crammer, K., & Singer, Y. (2001). Ultraconservative online algorithms for multiclass problems. In D. Helmbold, and B. Williamson (Eds.), COLT/EuroCOLT 2001, LNAI 2111 (pp. 99–115). Springer.

Dagan, I., Karov, Y., & Roth, D. (1997). Mistake-driven learning in text categorization. In *EMNLP '97, 2nd Conference on Empirical Methods in Natural Language Processing*.

Golding, A. R. & Roth, D. (1999). A winnow-based approach to spelling correction. *Machine Learning, 34*, 107–130.

Gorin, A. L., Riccardi, G., & Wright, J. H. (1997). How may I help you?. *Speech Communication, 23:1/2*, 113–127.

Kivinen, J., & Warmuth, M. (1997). Additive versus exponentiated gradient updates for linear prediction. *Journal of Information and Computation, 132:1*, 1–64.

Lewis, D., Schapire, R. E., Callan, J. P., & Papka, R. (1996). Training algorithms for linear text classifiers. In *SIGIR 96: Proc. of the 19th International Conference on Research and Development in Information Retrieval*.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning, 2*, 285–318.

Littlestone, N., & Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation, 108*, 212–216.

Ljolje, A., Hindle, D. M., Riley, M. D., & Sproat, R. W. (2000). The AT&T LVCSR-2000 System. In *Speech Transcription Workshop*, Univ. of Maryland.

Mesterharm, C. (2002). Tracking linear-threshold concepts with winnow. In *Proc. of the Annual Conference on Computational Learning Theory*, LNAI 2375, (pp. 138–152). Springer.

Riccardi, G., Pieraccini, R., & Bocchieri, E. (1996). Stochastic automata for language modeling. *Computer Speech and Language, 10*, 265–293.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*, 386–407.

Schapire, R. E. (2001). Drifting Games. *Machine Learning, 43*, 265–291.

Received October 8, 2003
Revised May 8, 2004
Accepted May 24, 2004