ONLINE LEARNING WITH MINORITY CLASS RESAMPLING

Michael J. Pekala, Ashley J. Llorens

Johns Hopkins University Applied Physics Laboratory
firstname.lastname@jhuapl.edu

ABSTRACT

This paper considers using online binary classification for target detection where the goal is to identify signals of interest within a sequence of received signals generated by a shifting background. In this setting, we assume there is significant class imbalance (100:1 or greater), the sequence of examples is arbitrarily long and the distribution of the majority (negative) class is slowly time-varying. This setting is typical in detection and classification problems in which time-varying effects are caused by some combination of shifting channel characteristics and interferers that enter and exit the scene. We show empirically that the addition of caching and minority class oversampling to online learners improves the g-means performance under these conditions by compensating for class imbalance.

Index Terms— Support vector machine, classification, online learning, class imbalance.

1. INTRODUCTION

In online binary classification problems, examples from a possibly infinite sequence \( \{x_i\}_{i \in I} \) are provided incrementally and the corresponding true label \( y_i \) must be predicted as each example \( x_i \) arrives. The underlying data distributions may be changing over time and the classification goal is to minimize the error incurred at each step based solely on knowledge of previously seen examples. This differs from batch classification problems where the entire training dataset is available \textit{a priori} and the classification goal is to minimize the expected risk based on the assumption that future examples will be drawn from the same distribution [1].

We consider a problem setting in which the goal is to identify signals of interest within a sequence of received signals generated by a shifting background. This setting is typical in detection and classification problems in which time-varying effects are caused by some combination of shifting channel characteristics and interferers that enter and exit the scene. Specifically, our approach utilizes kernel machine classification [2] under the following constraints: (P1) effective classification requires nonlinear kernels, (P2) the sequence of input examples is arbitrarily long, (P3) there is 100:1 or greater class imbalance in favor of negative examples (P4) the expected initial distributions are known (the accuracy of this assumption will vary by application, but we’re essentially assuming we have enough \textit{a priori} information to bootstrap the learner) and the positive distribution will remain stationary while the negative distribution is slowly time-varying. In this setting, we seek to learn a classifier that maintains effective performance on the relatively rare positive examples while adapting to maintain an acceptable false positive rate.

In addition to classification performance, another important consideration for online classifiers is computational complexity. Traditional nonlinear kernel-based estimators have the property that the complexity of the classifier grows linearly with the number of training examples [1]. Therefore, constraints P1 and P2 together force us to restrict our attention to kernel-based approaches that provide explicit mechanisms for managing complexity. We present an online learning approach that augments an existing kernel-based algorithm by adding minority class resampling and demonstrate that the resulting algorithm mitigates the effect of extreme class skew.

2. PRIOR WORK

A number of recent approaches address some subset of problem constraints P1-P4 using kernel-based techniques. The NORMA online learning algorithm [2] employs stochastic gradient descent with nonlinear kernels to train a classifier by minimizing the instantaneous risk. Their approach is especially compelling because it provides an explicit mechanism for limiting the complexity of the classifier by truncating less influential kernel centers. They show that this truncation technique provides a natural way of “forgetting” instances \((x_i, y_i)\) which may be old relative to the current example distribution. This forgetting mechanism is precisely what we want for instances of the time-varying negative class, but may have undesirable consequences in the presence of significant class imbalance as positive examples can be dramatically underrepresented. In this paper we adopt NORMA as a baseline and investigate the impact of class imbalance.

The Pegasos online learning algorithm [3] also employs stochastic gradient descent, and adds an additional projection step making it especially well suited for large datasets. Subsequent publications provide empirical results for Pegasos using nonlinear kernels, but none of these datasets exhibit our de-
sired combination of slowly time-varying distributions in the presence of large class imbalance.

Class imbalance has been addressed using a number of machine learning techniques, including class-specific sampling in the batch setting, e.g. [4] [5], and active selection in the online setting, e.g. [6]. In the latter case, the authors use active selection together with the online learning algorithm LASVM [7]; however, LASVM lacks a mechanism for adhering to a strict complexity budget given an arbitrarily long sequence of training examples.

3. APPROACH

In [2] the authors show that the update step for NORMA can be written as

\[ f_{t+1} = (1 - \eta_t \lambda) f_t - \eta_t \partial_f^l (f_t(x_t), y_t) \]  

where \( f_t \) is the function used to predict the label \( y_t \) based on \( x_t \) and previously seen examples, \( \eta > 0 \) is the learning rate, \( \lambda > 0 \) is the regularization parameter, \( \partial_f \) denotes the gradient with respect to \( f \) and \( l \) is the loss that the learning algorithm incurs when predicting \( y_t \). When \( f_t \) is expressed as a kernel expansion \( f_t(x) = \sum_{i=1}^t \alpha_i k(x_i, x) \), updating to \( f_{t+1} \) is equivalent to updating the coefficients \( \alpha_i \),

\[
\alpha_i = \begin{cases} 
-\eta_t \partial_f^l (f_t(x_t), y_t) & \text{for } i = t \\
(1 - \eta_t \lambda) \alpha_i & \text{for } i < t 
\end{cases}
\]  

It is important to note that coefficients associated with older examples are decayed by a factor \( (1 - \eta_t \lambda) \), so after \( \tau \) iterations the coefficient \( \alpha_i \) will be reduced to \( (1 - \eta \lambda)^\tau \alpha_i \). This suggests that kernel centers associated with older examples are likely less influential, and these terms can be dropped with little impact classification performance. The authors of [2] formalize this intuition by providing a bound on the resulting truncation error. This method of retaining the most recent kernel centers therefore controls classifier complexity while simultaneously emphasizing current examples.

A potential downside to NORMA’s truncation method occurs when class imbalance is particularly high and processing resources are limited (implying a strict kernel center budget). In this case, it is likely that \( f_t \) will be dominated by kernel centers representing the negative class. Consequently, performance on positive instances could be severely impacted, limiting effectiveness in the target detection setting where accuracy on positive examples is critical.

One possible remedy is to modify the truncation policy. Instead of discarding the oldest kernels, one could implement policies that preferentially discard kernels based on other criteria, such as the corresponding class label. However, this approach is of limited utility since the coefficients of the retained kernels are still progressively reduced by the decay factor \( (1 - \eta_t \lambda) \) during each iteration. Modifying this decay structure involves modifying the underlying stochastic gradient descent algorithm, which we do not consider here.

Instead, we consider an approach that selectively reintroduces data examples back into the training sequence. We augment the online classifier by adding two caches, one for storing known positive examples and the other for storing known negative examples. Maintaining two separate caches allows us to explicitly control the number of examples from each class we retain, independent of any underlying data skew. As each new example arrives, once its true label is determined and the classifier is updated per equation 1, the appropriate cache is queried for examples from the opposite class. Selected cached examples are then presented to the classifier, providing an opportunity to incorporate those feature vectors into the kernel representation. Since negative examples far outnumber positive examples in our problem, this process will usually draw an example from the positive cache making this an online version of minority class resampling [4].

This approach requires policies for (i) drawing examples from a cache and (ii) removing cached items when a cache’s capacity is reached. For (i), we sample uniformly from each cache and, for (ii), we always remove the oldest cached item. These policies are appropriate in this setting based on assumption P4; however more sophisticated approaches, such as active selection, could be considered when initial distributions are unknown (e.g. training data are scarce or noisy).

We consider variants of this approach that consult the cache less frequently. Changing the cache access rate will alter the expected ratio of positive and negative examples maintained in the kernel expansion. In the extreme, the policy of never consulting the cache reduces to the unmodified NORMA algorithm. Algorithm 1 shows the pseudocode for online learning with resampling.

### Algorithm 1 Online learning with resampling

```plaintext
for t = 1, ... do
    \( x_t \) := get-next-example-from-environment();
    \( \hat{y}_t \) := predict-label(\( x_t \));
    if true label \( y_t \) can be determined then
        update-classifier(\( x_t, y_t \));
        cache-example(\( x_t, y_t \));
    end if
    if sample-cache-this-iteration() \( \Rightarrow \) True then
        (\( x, y \)) := draw-example-from-cache(\( -y_t \));
        update-classifier(\( x, y \));
    end if
end for
```

In classification problems involving significant class imbalance, an alternative to minority class resampling is to downsample the majority class. For online classification, downsampling is easier to implement than minority class resampling, but has the drawback that it discards potentially useful negative examples. Since we assume the negative class
distribution is shifting and we wish to maximally exploit the characteristics of the current distribution, we do not pursue the downsampling approach here.

4. RESULTS

For our experiments we compare the performance of a “static” (batch) classifier to that of an unmodified NORMA and a version of NORMA augmented with minority class resampling. We evaluate these algorithms on a variant of the MNIST handwriting dataset [8]. The unmodified MNIST dataset consists of 60,000 dedicated training examples and 10,000 test examples, where each example is a vector of 784 grayscale values corresponding to a handwritten digit 0-9. In the natural “one versus all” MNIST classification problem, each digit is treated in turn as the positive class and the remaining 9 digits represent negative class examples (e.g. the M2 test case uses digits labeled 2 as positive class instances and all other digits as negative class instances).

By itself, this dataset does not exhibit the desired level of class skew (≥ 100:1) and does not directly encode any concept of a drifting negative class data distribution. To address class skew, we construct an online variant of the MNIST dataset where sequential examples are drawn uniformly with replacement from either the set of positive examples (with probability p) or from the set of negative examples (with probability 1 − p). When a negative example is drawn at iteration t, it is rotated by some angle θ(t) = ct (measured in degrees, c a fixed scalar) before being presented to the classifier. This modification simulates a detection problem with a dynamic interference picture.

Each experiment consists of two phases: first, in the “offline” phase a classifier is trained using examples drawn from the training set with p = 0.1 (the natural skew) and with a fixed distribution for negative examples (i.e. no rotation is applied). Then, in the second, “online” phase, we simulate deploying the classifier by selecting a new value for p and applying the simulated time-varying distribution of negative examples. Each classifier is assessed based on its performance in this “online” phase. For our experiments we used \( \lambda = 1 \times 10^{-7} \), Radial Basis Function (RBF) kernels with center width \( \gamma = 0.1 \), determined by cross-validation, along with a kernel center budget of 250, an example cache size of 1000 and a time-varying \( \eta_t \) as described in [3]. For the modified algorithm, we consider three probabilities of resampling from the positive cache during each iteration: 1.0, 0.5 or 0.05. The baseline static classifier is simply NORMA where all truth labels are withheld during the “online” phase, preventing the algorithm from adapting.

Classification accuracy is not an ideal evaluation metric when class skew is significant [6]. For example, given an imbalance ratio of 99 to 1, a trivial classifier that labels any example as belonging to the negative class will achieve 99% accuracy. Accordingly we use the \( g \)-means metric, which equally weights performance on positive examples (sensitivity/recall) and negative examples (specificity) independent of class imbalance.

Figure 1 shows the results for the M2 problem with 500,000 online examples and a 10,000:1 ratio of negative to positive examples. Values reported for sensitivity, specificity and \( g \)-means indicate the cumulative value as of iteration t. Thus, these quantities tend to fluctuate in early iterations before converging to steady state values. The poor performance of unmodified NORMA on positive examples, shown in Figure 1(a), demonstrates its inability to compensate for a high level of class skew.

The static classifier is non-adaptive and therefore not susceptible to increased bias resulting from the increased skew in the online data stream. In terms of \( g \)-means, the static classifier outperforms unmodified NORMA while underperforming the modified online learning algorithms. The static classifier is inferior to the augmented online algorithms due to its inability to adapt to the changing distribution of negative examples, suggested by its relatively poor performance on negative examples shown in Figure 1(b).

In the target detection setting, reacting to false positives wastes resources and time, making a large number of false positives particularly undesirable. The static algorithm’s inferior performance on the more numerous negative examples leads to more false positives. Unmodified NORMA exhibits fewer false positives, but at the cost of only correctly identifying approximately 1/3 of the positive examples.

Our approach preserves performance on the relatively rare positive examples while mitigating the effects of class skew. Although not included in these experiments, our results suggest that the cache resampling rate could be tuned to satisfy an application-specific constraint on the rate of false positives.

Table 1 summarizes results for the M8 problem under varying levels of class skew. Values reported are the \( g \)-means score obtained at the end of a randomly drawn 200,000 example data stream. These results, averaged over 10 Monte Carlo runs, suggest that the addition of online resampling permits the classifier to retain comparable \( g \)-means performances under more severe levels of class skew.

5. SUMMARY & FUTURE WORK

We’ve developed an online learning algorithm that is well-suited for use in a target detection setting where there is significant class imbalance, the stream of examples can be arbitrarily long and the distribution of the majority class is slowly
Fig. 1. M2 problem, $c = 0.1, p = 0.0001$. The modified algorithm outperforms traditional online learning on the relatively rare positive examples (a), and outperforms the static classifier on the frequent negative examples (b). The net result is an overall improvement in g-means score (c).

Table 1. G-means results for the M8 problem, using parameters $c = 0.1$ and A: $p=0.1$, B: $p=0.001$, C: $p=0.0001$. The modified algorithm with caching retains good performance even as data skew increases.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMA</td>
<td>0.95</td>
<td>0.71</td>
<td>0.43</td>
</tr>
<tr>
<td>static</td>
<td>0.90</td>
<td>0.89</td>
<td>0.89</td>
</tr>
<tr>
<td>NORMA + caching</td>
<td>0.97</td>
<td>0.96</td>
<td>0.96</td>
</tr>
</tbody>
</table>

Possible directions for future research in this area include incorporating adaptive stochastic gradient descent step sizes [9], utilizing more sophisticated cache resampling policies, investigating the impact of various time-varying data distributions and incorporating unsupervised learning techniques in cases where only a subset of the true labels are available for examples encountered in situ.

6. ACKNOWLEDGEMENTS

The authors would like to thank I-Jeng Wang and the reviewers for their helpful comments and suggestions.

7. REFERENCES


