Practical Online Active Learning for Classification

Claire Monteleoni*
Department of Computer Science and Engineering
University of California, San Diego
cmontel@cs.ucsd.edu

Matti Kääriäinen
Department of Computer Science
University of Helsinki
matti.kaariainen@cs.helsinki.fi

Abstract

We compare the practical performance of several recently proposed algorithms for active learning in the online classification setting. We consider two active learning algorithms (and their combined variants) that are strongly online, in that they access the data sequentially and do not store any previously labeled examples, and for which formal guarantees have recently been proven under various assumptions. We motivate an optical character recognition (OCR) application that we argue to be appropriately served by online active learning. We compare the practical efficacy, for this application, of the algorithm variants, and show significant reductions in label-complexity over random sampling.

1. Introduction

The emerging problem of OCR on small devices is one of several real-world motivations for online active learning algorithms. As of 2004, a quarter of US physicians were already using handheld computers.[1] In the 2004 US presidential election, several major political organizations equipped canvassers going door-to-door with handheld computers to collect neighborhood voting data. Limited computing power may constrain the OCR training of these handhelds to be online. In an online active learning setting, the device could occasionally ask the user to provide a classification label for a written character that was difficult to interpret. The human could then enter the character's label through the keypad, for example. Human usage would likely favor algorithms that minimize the number of such correction events during the learning process.

At a general level, the *active learning* model is applicable to any domain in which unlabeled data is easy to come by and there exists a (potentially difficult or expensive) mechanism by which to obtain their labels. By recasting a supervised problem into the active learning framework, the label-complexity, the number of labeled examples required to learn a concept via active learning, can be significantly lower than the PAC sample complexity. While the query learning model has been well studied theoretically (see e.g. [1]), it is often unrealistic in practice, as it requires access to labels for the entire input space. It has been shown in domains such as OCR and text that the synthetic examples on which the learner has the most uncertainty may be difficult even for a human to label [14]. In the selective sampling model (originally introduced by [7]) the learner receives unlabeled data and may request certain labels to be revealed, at a constant cost per label. We will operate in this model, which we will refer to below as active learning.

The *online learning* framework is motivated in part by resource limitations facing computational learners, and the vast amounts of data involved in many machine learning applications. In the online learning model, the learner receives observations one at a time and is constrained against storing all previously seen observations and then simply performing batch learning. In particular, neither the time complexity of the hypothesis update, nor the learner's memory usage, may grow with the number of seen examples. Active learning can be modeled in an online or sequential fashion, in which unlabeled examples are received one at a time and the learner must make a one-time choice whether to pay for the current label. We refer to this framework as *online active learning*. Algorithms for sequential active learning that also respect online constraints on time and memory, we will refer to as strongly online active learners, though with a slight overload of terminology we will also refer to them simply as online active learners.

The problem of online active learning has received recent attention in the machine learning theory literature. We will focus on algorithms that both have some form of theoretical guarantee and perform sequential active learning in a strongly online fashion: neither the time-complexity of their hypothesis update step, nor their memory usage, scale with

*Work done primarily while at MIT Computer Science and Artificial Intelligence Lab.

[1] McAlearney AS, Schweikhart SB, Medow MA, Doctors' experience with handheld computers in clinical practice: qualitative study. British Medical Journal. 2004 May 15;328(7449):1162.

1-4244-1180-7/07/$25.00 ©2007 IEEE

the number of examples seen. In fact, as they are both based on Perceptron variants, they each store only a single vector and their algorithmic form is very light-weight and easy to implement. The two algorithms have not been compared theoretically and since some of their analysis assumptions are rather limiting, we evaluate them on real data, in order to assess their performance when these assumptions are relaxed. Moreover, we illustrate the useful application of online active learning to optical character recognition.

2. Related Work

Several active learning algorithms have been shown to work in practice, for example Lewis and Gale's sequential algorithm for text classification [14], which has batch access to the remaining unlabeled datapoints at each iteration. Tong and Koller [17] introduced several active learning algorithms, that use a support vector machine (SVM) as the underlying classifier, which work well in practice. At each step, the SVM algorithm is used to update the hypothesis, and the next query is selected from the pool of remaining unlabeled data by optimizing a margin-based heuristic that varies between different versions of the algorithm. Although, as very recent work has shown [3], it is possible to define and analyze variants of these heuristics that operate in the sequential setting, the use of the SVM sub-algorithm breaks the online constraints on time and memory. Thus, neither of these approaches are strongly online.

Among the active learning algorithms that have been shown to obey theoretical guarantees, several schemes with provable upper bounds on label-complexity are actually intractable. Dasgupta provided a general result in a non-Bayesian, realizable setting (i.e. there exists a perfect separator for the data, in the concept class over which the learning is perfomed) for a scheme that requires exponential storage and computation [8]. In a non-Bayesian, agnostic setting, Balcan, Beygelzimer and Langford provided a label-complexity upper bound for learning linear separators under the uniform input distribution, that relies on an algorithm that is computationally complex [2].

Several formal guarantees have been shown for active learning algorithms that can actually be implemented. Under Bayesian assumptions, Freund et al. [10] gave an upper bound on label-complexity for learning linear separators under the uniform, using Query By Committee [16], a computationally complex algorithm that has recently been simplified to yield encouraging empirical results [11]. Cesa-Bianchi, Conconi and Gentile provided regret bounds on an active learning algorithm for learning linear thresholds [5] from a stream of iid examples corrupted by random class noise whose rate scales with the examples' margins. Both algorithms store all previously labeled points, and thus are not online.

We focus on two algorithms for sequential active learning that are strongly online, and whose performance has been analyzed formally under various assumptions. Dasgupta, Kalai and Monteleoni (DKM) [9] provided an online active learning algorithm with a label-complexity upper bound for learning linear separators under the uniform input distribution, in a non-Bayesian, realizable setting. We will explore the empirical performance of this algorithm when these assumptions of separability and uniform input distribution are violated, by applying it to real data from OCR, an application that, as we explained above, is particularly appropriate for strongly online active learning. We compare DKM's performance to another state-of-the-art strongly online active learning algorithm due to Cesa-Bianchi, Gentile and Zaniboni (CBGZ) [6], which has regret bounds in the individual sequence prediction context.

Initialization: $s_1 = \frac{1}{\sqrt{d}}$, $\tau = 0$, $t = 1$, $v_1 = x_0 y_0$, $\tau = 0$.
Do
  Receive x.
  Predict $\hat{y} = sign(x \cdot v_t)$.
  If $|x \cdot v_t| \leq s_t$ **then:**
    Query the label $y \in \{-1, +1\}$ of x.
    Set $(x_t, y_t) = (x, y)$.
    If $(x_t \cdot v_t) y_t < 0$, **then:**
      $v_{t+1} = v_t - 2(v_t \cdot x_t) x_t$
      $s_{t+1} = s_t$
      $\tau = 0$
    else:
      $v_{t+1} = v_t$
      $\tau = \tau + 1$
      If $\tau \geq R$, **then:**
        $s_{t+1} = s_t / 2$
        $\tau = 0$
      else: $s_{t+1} = s_t$
    $t = t + 1$
Until $t == L$

Figure 1. The DKM algorithm, parameterized by the dimension d, and R, the waiting time before halving the active learning threshold.

3. Algorithms

The algorithms we consider are both for learning linear separators through the origin, in the online active learning framework. We note that they can both be kernelized to handle richer concept classes, however for clarity of exposition, we will focus on learning linear separators in $\mathbb{R}^d$. Each algorithm can be decomposed into two parts: an active learning mechanism, wrapped around a sub-algorithm that implements supervised learning.

3.1. The DKM algorithm

The DKM active learning algorithm is shown in Figure 1. The formal framework of the Dasgupta et al. work [9] is a non-Bayesian setting in which no prior distribution is assumed over hypotheses, yet the problem is assumed to be realizable i.e. there exists a target linear separator that perfectly classifies all examples. The stream of examples is assumed to be drawn iid from the uniform distribution on the surface of the ball in $\mathbb{R}^d$. This is without loss of generality, as only the angle, not the magnitude, of a vector determines its classification by a halfspace through the origin. Labels y can be either $+1$ or -1.

The supervised learning sub-algorithm of DKM is a modification of Perceptron. The same logic is used in deciding whether to perform a hypothesis update: updates are only made if the seen example was a mistake. However the update rule differs from standard Perceptron's update as follows. Starting from hypothesis v_t, as opposed to the Perceptron update $v_{t+1} = v_t + \mu\, y_t x_t$, with learning rate μ, the DKM update is $v_{t+1} = v_t - 2(v_t \cdot x_t)x_t$. This is equivalent to replacing the learning rate with $2|v_t \cdot x_t|$, since updates are only made on mistakes, in which case $y_t = -\,sign(v_t \cdot x_t)$. This essentially tunes the size of the update to be proportional to a measure of confidence on v_t's prediction on the example. Using this update, the algorithm monotonically decreases its true error rate with each mistake, unlike Perceptron whose true error rate can actually increase on an update.[2]

In the realizable setting, when the input distribution is uniform, Dasgupta et al. [9] prove a mistake bound for the supervised algorithm in terms of ϵ, the final true error rate attained. The error rate decreases exponentially with the number of mistakes (the mistake bound is logarithmic in $\frac{1}{\epsilon}$), whereas the mistake bound for the Perceptron update in this setting is polynomial in $\frac{1}{\epsilon}$ [4]. They also provide a polynomial lower bound on mistakes (and therefore labels for the active setting) on Perceptron with any active learning rule, under the uniform input distribution.

The active learning component of DKM consists of querying for a label only if $|v_t \cdot x| < s_t$ for an active learning threshold s_t. The threshold is initialized to $\frac{1}{\sqrt{d}}$ in the uniform case. If the learner queries on R consecutive labels without hitting an error, then s_t is halved. Since the algorithm is more likely to err on examples that are close to its current separator, this technique manages the tradeoff between waiting too long to query an actual error (if s_t is too high), and making an update that is too small (if s_t is too low), since $|v_t \cdot x|$ weights the DKM update. Dasgupta et al. [9] prove a label-complexity bound of the same form as the mistake bound, i.e. the true error of the algorithm decreases exponentially with the number of label queries. As in the supervised case, the proof assumes separability and the uniform input distribution. Monteleoni extended this to input distributions that are λ-similar to uniform, i.e. for any subset of the input space, the ratio between the distribution's measure and the uniform measure is upper and lower bounded by constants [15].

[2] As a side effect, the norm of the hypothesis stays constant at one, unlike that of Perceptron.

`Initialization:` $t = 1,\ v_1 = (0, \ldots, 0)^\top$.
Do
  `Receive` x.
  `Set` $\hat{p} = x \cdot v_t$, `and predict` $\hat{y} = sign(\hat{p})$.
  `Toss a coin with` $P(\text{Heads}) = \frac{b}{b+|\hat{p}|}$.
  If Heads
    `Query the label` $y \in \{-1, +1\}$ `of` x.
    If $y \neq \hat{y}$, **then**:
      $v_{t+1} = v_t + \eta y x^\top$
    else:
      $v_{t+1} = v_t$
    $t = t + 1$
Until $t == L$

Figure 2. The CBGZ algorithm, parametrized by $b > 0$ and learning rate $\eta > 0$.

3.2. Application to the non-uniform setting

In applying the algorithm to the non-uniform setting we changed the initial setting of the active learning threshold. Dasgupta et al. used $s_1 = \frac{1}{\sqrt{d}}$ in the uniform case based on a fact about uniform random projections (cf. Appendix of [9]) that need not hold when the distribution is non-uniform. Instead of starting the initial learning threshold so low, we make no assumptions about the input distribution and thus set the initial threshold to the maximum value that $|x \cdot v_t|$ could take, which is one, since $\|x_t\| = \|v_t\| = 1$ for all t. Changing the initial active learning threshold might imply that R should also differ from the value given in [9]. Regardless, since that paper did not focus on optimizing constants, we tuned R, along with the parameters of the other algorithms, as discussed in the evaluation section.

3.3. The CBGZ algorithm

Similar to DKM, the strongly online active learning algorithms proposed by Cesa-Bianchi et al. [6] are based on augmenting Perceptron-type algorithms with a margin-based filtering rule; for the first-order version used in our experiments, see Figure 2. The algorithm queries for a label with probability $b/(b + |\hat{p}|)$, where $\hat{p}$ is the margin of the example with respect to the current hypothesis, and $b > 0$ is a parameter. If a label is queried and the algorithm's prediction $sign(\hat{p})$ is incorrect, a standard Perceptron update is performed. The main result in [6] is a bound on the ex-

pected number of mistakes the algorithm makes on arbitrary input sequences (with respect to the algorithm's randomness). Both this mistake bound and the expected number of label queries depend on b. By optimizing b for the mistake bound, one can match the mistake bound for standard Perceptron. However, this choice of b may result in querying almost all the labels. The optimal choice of b depends on the data, and thus in practice is known only in hindsight. To circumvent this issue, the authors provide and analyze a method for tuning b on the fly, but in practice this adaptive strategy has inferior performance [6].

The theoretical results for DKM and CBGZ are incomparable, as the algorithms are based on different assumptions (uniform distribution with linear separability vs. individual sequences) and the main results give bounds for different quantities (both accuracy and label-complexity for DKM vs. mistake bounds for CBGZ). The lack of unified theoretical results is one motivation for our empirical study of the performance of these algorithms on real data.

4. Evaluation

4.1. Comparison class of algorithms

In designing our evaluation, we considered comparing to the SVM-based active learning algorithms proposed by Tong and Koller [17], as they are well-known benchmarks for active learning. However the online classification framework we consider differs from their pool-based model in which active learners have unlimited access to all unlabeled data. Although their active learning criteria could also be adapted for the sequential setting (see e.g. [3]), their algorithmic form is less constrained: SVMs do not obey the online constraints on storage and running time that we are concerned with in this work.

Given these extra degrees of freedom, we would expect SVM-based methods to outperform all the strongly online algorithms. We confirmed this with experiments using an SVM as the sub-algorithm, paired both with random queries and with the Simple active learning heuristic [17], with batch access to the remaining unlabeled pool. In both cases the number of labels queried was strictly lower than that of all the online algorithms studied, for each associated error rate. Since random sampling from a pool (drawn iid from the input distribution) is equivalent to a stream of iid draws from the input distribution, the random sampling case can be viewed as sequential, in terms of the data observation.

Thus the ability to break the online constraints on time and memory by running the SVM sub-algorithm, suffices to provide improved performance versus the strongly online algorithms. In fact the gains in performance due to using SVMs instead of online methods were greater than those from using the Simple active learning heuristic [17] instead of random sampling. Our conclusion from those experiments is that the online active learning algorithms studied in this paper seem to be most useful in settings with strict online requirements, while methods without online constraints seem to have superior performance when applicable.

Therefore as an appropriate comparison class, we instead consider only algorithms that are strongly online. Thus we compare all six combinations of the two online active learning rules discussed above, as well as random sampling, paired with two strongly online supervised learning algorithms: Perceptron and the supervised update of DKM. We will denote as DKM^2 the exact algorithm from [9], i.e. the DKM active learning logic with the DKM supervised update as the sub-algorithm. Running DKM's active learning logic with Perceptron as the sub-algorithm, we refer to as DKMactivePerceptron. We will denote as CBGZ, the CBGZ active learning rule with Perceptron as the sub-algorithm, as specified in [6]. For the sake of completeness, we also experimented with combining CBGZ's active learning rule and the DKM update, denoted below as CBGZactiveDKMupate. The random sampling methods simply flip a coin as to whether to query the current point's label, and update using Perceptron (randomPerceptron) or the DKM update (randomDKMupdate). This method is equivalent to performing supervised learning with the sub-algorithm in question, as it yields a sequence of labeled examples that are simply iid samples from the input distribution.

4.2. Experiments

We conducted our evaluation on benchmark data from OCR, since OCR on small devices could stand to benefit from strongly online active learning solutions. Additionally, these datasets are known to be non-uniformly distributed over inputs. We used both MNIST [13] and USPS in order to experiment with multiple datasets and dimensionalities ($d = 784$ for MNIST, $d = 256$ for USPS).

We experimented on 7 binary classification problems, 5 from MNIST and two from USPS, each consisting of approximately 10,000 examples. All the problems but two were linearly separable. (Using svmLight [12] we were unable to find separating hyperplanes for the problem $\{1,4,7\}$ vs. all other characters, both in the MNIST and USPS versions). Since the algorithms access the data in one pass, in a sequential fashion, for each problem we ran 5 runs, in which the dataset was uniformly re-permuted, of 10 fold cross-validation.

Several of the algorithms have parameters: Perceptron's learning rate, CBGZ's b and DKM active learning version's R, so each was tuned independently on a separate holdout set for each problem, using 10 fold cross-validation on approximately 2,000 examples. A threshold on test error, ϵ, was chosen for each problem based (qualitatively) on level of separability, and each algorithm's parameters were tuned to minimize the number of labels, averaged over folds, to

MNIST:	0v1 (0.01)	0vAll (0.05)	4v7 (0.05)	6v9 (0.025)	147vAll (0.15)
DKM^2	28.02±25.26	105.30±42.39	150.02±49.61	163.12±42.45	275.34±72.00
DKMperc	**13.78±5.88**	**57.26±15.92**	**44.00±15.32**	**20.44±11.75**	217.06±75.85
CBGZ_DKM	130.12±116.45	183.78±120.83	194.36±80.20	218.28±94.95	379.16±138.38
CBGZperc	32.78±19.52	62.66±30.48	63.32±30.76	30.66±13.31	**170.02±63.61**
randDKM	87.72±101.84	173.44±114.55	276.92±150.59	367.24±191.25	375.46±164.33
randPerc	83.76±78.47	103.74±76.25	107.98±57.41	104.06±75.10	214.16±93.12

USPS:	0vAll (0.05)	147vAll (0.1)
DKM^2	174.22±63.85	190.72±80.09
DKMperc	**87.56±28.97**	137.86±58.21
CBGZ_DKM	156.08±66.75	193.70±96.35
CBGZperc	115.14±61.09	**116.28±60.68**
randDKM	235.10±129.11	210.40±109.39
randPerc	173.96±98.96	151.32±72.65

Figure 3. Mean and standard deviation (over 5 runs of 10 fold cross-validation) of the minimum number of labels to reach the test error threshold (in parentheses) for the problem.

reach test error ϵ.

In Figure 3 we report the means and standard deviations, over all the experiments run, of the minimum number of labels after which each algorithm reached the test error threshold (ϵ listed in parentheses) for that problem. On every problem, online active learning shows a clear advantage over online random sampling. Compared to the best active learning method for each problem, at the comparison test error threshold, Perceptron with random sampling used more labels by a factor between 1.26–6.08, and for more than half of the problems the factor was 2 or higher.

In our discussion of the results, we will indicate which conclusions can be drawn from the mean label values reported in the table, with statistical significance. We tested statistical significance using Wilcoxan signed rank hypothesis testing, which is non-parametric and thus robust, and which takes into account the magnitude of the difference in mean labels, between the algorithms compared, for each problem.

The minimum number of labels was attained by DKMactivePerceptron, in all but two of the problems, in which the minimum was achieved by CBGZ. DKMactivePerceptron also reports the smallest variance on this figure, in all but one problem. Both methods used significantly fewer labels than the random sampling methods. We tested this by assuming as the null hypothesis that the active learning method in question did not reduce label-complexity beyond that of Perceptron with random sampling (the best-performing random sampling method), yielding, for CBGZ, a p-value of 0.0156, entailing that the null hypothesis is rejected for significance levels of 1.56% and higher, and, for DKMactivePerceptron, at significance levels of 3.13% and higher ($p = 0.0313$). The difference in these two active learning algorithms' performance, compared pairwise per problem, was not statistically significant however. Interestingly, the problems for which CBGZ was the top performer are the only two unseparable problems. Although both algorithms use Perceptron as the sub-algorithm, we did not have enough unseparable problems to conclude, with statistical significance, whether CBGZ's active learning rule is better equipped for the non-realizable case than that of DKM.

Similarly, using the DKM active learning rule showed significant improvements over using the DKM update with random sampling. DKM^2 used fewer labels per problem than randomDKMupdate, at significance levels of 1.56% and higher. This is another result that online active learning incurs significant label savings, compared to random sampling with the same supervised online sub-algorithm.

The DKM supervised update, and methods that used it as their sub-algorithm tended to perform worse than their Perceptron counterparts. This is statistically significant at significance levels of 1.56% and higher, for each pairwise comparison between the Perceptron and DKM updates, with the active learning rule fixed. In the unseparable cases, this may be explained by DKM's update being much more aggressive than Perceptron's, when the point has a large margin: the DKM update adds to its hypothesis the same quantity, $y_t x_t$, as Perceptron, however scaled by a factor of $2|x_t \cdot v_t|$, which could be greater than one, as opposed to the Perceptron's learning rate which is less than one. The comparison may not be completely fair however, as the DKM update was the only algorithm without parameters, and thus the only algorithm not at all tuned per problem. In fact, both of the active learning variants of standard Perceptron were actually doubly tuned per problem, as the Perceptron learning rate was first tuned, and then the active learning parameter (R or b) was tuned for the problem, using the tuned Perceptron as the sub-algorithm. It is also important to note that mistake bounds implying better performance of the DKM update than Perceptron have only been shown under uniform [9], and λ-similar to uniform [15], input distributions, and here the input distribution is

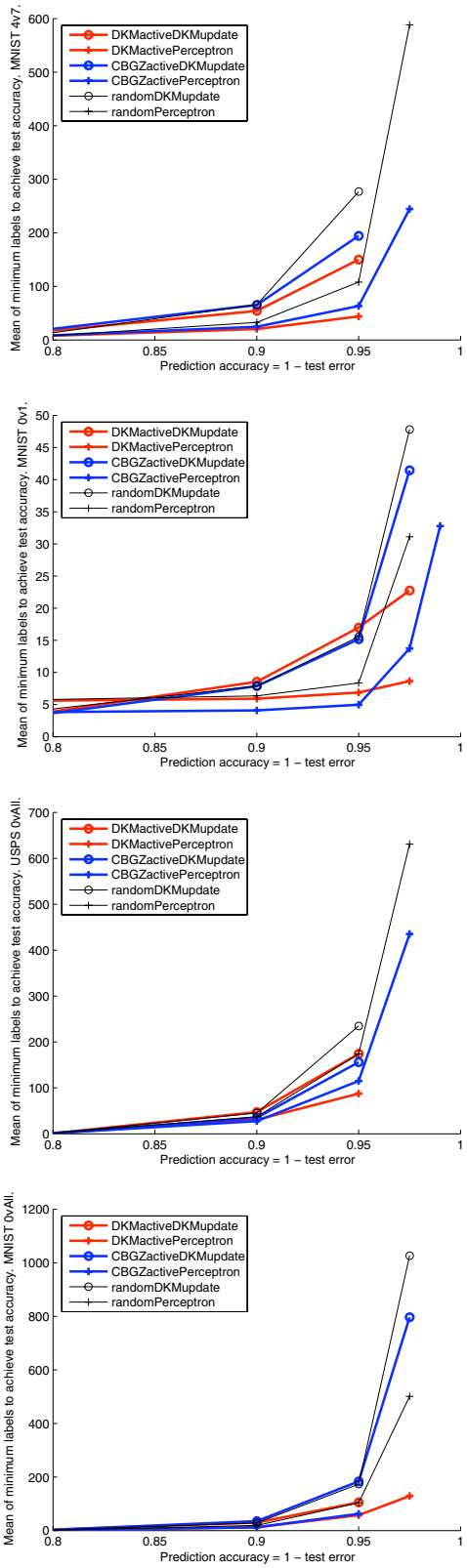

Figure 4. Statistical efficiency. Mean minimum labels to attain test accuracy (i.e. $1 -$ test error) above each threshold is over 5 folds 10 runs if all folds/runs reached that test accuracy. a). MNIST 4v7. b) MNIST 0v1. c) USPS 0vAll. d) MNIST 0vAll.

known to be highly non-uniform.

For both sub-algorithms, the DKM active learning rule tended to outperform the CBGZ active learning rule; with the DKM update as the sub-algorithm, the DKM active learning rule (DKM^2) used fewer labels than that of CBGZ (CBGZactiveDKMupdate) in all problems but one. As mentioned above, for the Perceptron-based methods, this observation does not have statistical support. However for the algorithms using the DKM update as the sub-algorithm, the advantage of the DKM active learning rule over that of CBGZ is statistically significant at significance levels of 4.69% and higher.

In Figure 4 we plot statistical efficiency curves. Points indicate the average over all the experiments of the minimum number of labels to attain test error lower than a given threshold on test error (i.e. one minus the value plotted on the x-axis), only if all experiments reached that threshold. It is important to note that the algorithms were only tuned to minimize labels to reach one of these test error thresholds; an algorithm that was minimal at the chosen threshold need not be minimal at all thresholds plotted. Some of the plots illustrate a slower rate of label increase for algorithms using DKM as their active learning rule. Not all the plots were as conclusive, but an interesting example is Figure 4 b) in which the DKM active algorithms have higher label usage, at most of the thresholds measured, than their CBGZactive counterparts, however the rate of label increase, as the test error decreases (x-axis increases), appears to be much slower.

To provide a qualitative perspective we present some representive learning curves, with respect to labeled examples, in Figure 5. We show a) a problem that was particularly easy and b) a problem that we did not find to be linearly separable. Figure 5 c) and d) compare the MNIST and USPS versions of the problem of 0 vs. all other characters, which is separable but has a large label imbalance with very few positive examples. In these problems, while DKMactivePerceptron reduces test error at a faster rate than all the other algorithms, DKM^2 and CBGZ continue querying for more labels, eventually reaching lower error.

5. Discussion and Conclusions

Our experimental framework may have been susceptible to overfitting of the parameter settings to the holdout tuning set, per problem, which may have prevented some of the algorithms from generalizing well to the actual problem data. Supervised DKM has no parameters, but for all other algorithms, tuning was involved.

Another tuning issue relates to the behavior observed above in Figure 5 c) and d), of DKMactivePerceptron attaining an initial test error threshold faster than all other algorithms, but then not querying for any more labels in the rest of the fold. This issue is related to how one should set

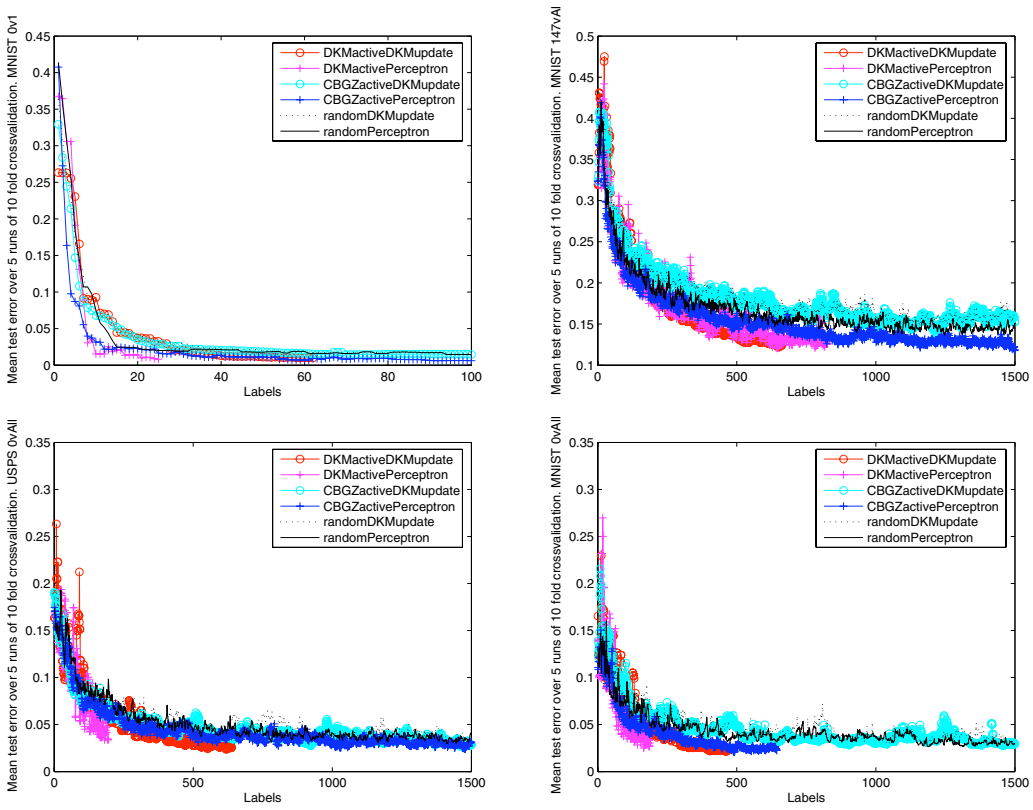

Figure 5. Learning curves. a) An extremely separable problem, MNIST 0v1. b) An unseparable problem, MNIST 147vAll. c) USPS 0vAll. d) MNIST 0vAll.

the waiting time threshold R for the DKM active learning algorithm. With a very small value of R, the algorithm has very little tolerance for labeled examples that do not yield a mistake and thus an update, and so will quickly halve its active learning threshold. Labeling an example with a smaller margin with respect to the current hypothesis is more likely to yield a mistake. Although this can cause a steep descent of error with respect to the number of labels, once the active learning threshold becomes too small, the algorithm will hardly ever make label queries. Since we are experimenting on a finite set of data as opposed to an endless stream, this means the algorithm may not query for any more labels on the fold. Ideally, we would like to optimize the constants in DKM so that the parameter R need not be tuned, but this is left for future work. Additional future work would entail modeling other domains using online active learning, and testing the performance of these algorithms therein.

At a general level, we conclude that online active learning provides significant performance gains over random sampling the same number of labels, when the random sampler must obey online constraints on memory and computation. In particular, we provide an application of DKM, and to our knowledge this algorithm had not yet been applied in practice. We study the performance of DKM when the input distribution is non-uniform, a question left open by the work of [9], as the performance guarantees were shown under the assumptions of realizability and a uniform input distribution. When these assumptions are violated, we observe that, in this application, DKM active learning has better performance when paired with standard Perceptron as the supervised sub-algorithm, as opposed to the update proposed in [9]. This is an outcome we did not predict, due to the strikingly better performance guarantees of the update proposed in [9], with respect to Perceptron, under the uniform assumption.

Moreover, we show the efficacy of these online active learning algorithms for classification, in an OCR application. While there has been recent progress on theory and algorithms for online active learning, our work is a first step towards applying these new techniques in practice. We hope that these techniques will find further use in computer vision, and other applications that involve vast amounts of data, or constraints on computational resources, and for which labeled data is expensive or difficult to obtain.

Acknowledgments

The first author would like to thank Tommi Jaakkola and Sanjoy Dasgupta for helpful discussions, as well as Luis Perez-Breva and Jason Rennie for technical advice.

References

[1] D. Angluin. Queries revisited. *In Proc. 12th Int. Conference on Algorithmic Learning Theory*, LNAI,2225:12–31, 2001. 1

[2] M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *International Conference on Machine Learning*, 2006. 2

[3] M.-F. Balcan, A. Broder, and T. Zhang. Margin based active learning. In *Proc. 20th Annual Conference on Learning Theory*, To Appear, 2007. 2, 4

[4] E. B. Baum. The perceptron algorithm is fast for nonmalicious distributions. *Neural Computation*, 2:248–260, 1997. 3

[5] N. Cesa-Bianchi, A. Conconi, and C. Gentile. Learning probabilistic linear-threshold classifiers via selective sampling. In *The Sixteenth Annual Conference on Learning Theory*, 2003. 2

[6] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Worst-case analysis of selective sampling for linear-threshold algorithms. In *Advances in Neural Information Processing Systems 17*, 2004. 2, 3, 4

[7] D. A. Cohn, L. Atlas, and R. E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994. 1

[8] S. Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in Neural Information Processing Systems 18*, 2005. 2

[9] S. Dasgupta, A. T. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. In *Proc. 18th Annual Conference on Learning Theory*, 2005. 2, 3, 4, 5, 7

[10] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997. 2

[11] R. Gilad-Bachrach, A. Navot, and N. Tishby. Query by committee made real. In *Advances in Neural Information Processing Systems 18*, 2005. 2

[12] T. Joachims. Making large-scale support vector machine learning practical. In A. S. B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998. 4

[13] Y. Lecun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/. 1998. 4

[14] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proc. of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, 1994. 1, 2

[15] C. E. Monteleoni. Learning with online constraints: Shifting concepts and active learning. PhD Thesis. *MIT Computer Science and Artificial Intelligence Lab Technical Report 2006-057*, 2006. 3, 5

[16] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proc. Fifth Annual ACM Conference on Computational Learning Theory*, 1992. 2

[17] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001. 2, 4