KERNEL EXPANSION FOR ONLINE PREFERENCE TRACKING

Yvonne Moh **Joachim M. Buhmann**
Institute of Computational Science
Swiss Federal Institute of Technology (ETH) Zurich
{tmoh,jbuhmann}@inf.ethz.ch

ABSTRACT

User preferences of music genres can significantly changes over time depending on fashions and the personal situation of music consumers. We propose a model to learn user preferences and their changes in an adaptive way. Our approach refines a model for user preferences by explicitly considering two plausible constraints of computational costs and limited storage space. The model is required to adapt itself to changing data distributions, and yet be able to compress “historical” data. We exploit the success of kernel SVM, and we consider an online expansion of the induced space as a preprocessing step to a simple linear online learner that updates with maximal agreement to previously seen data.

1 INTRODUCTION

Adaptivity is an indispensable prerequisite for personalization of acoustic devices like hearing aids. Hearing instruments, for instance, are often expected to suppress distracting or even annoying acoustic background sounds, whereas enjoyable acoustic scenes should be enhanced.

Such acoustic scenes can often be differentiated in a fairly fine-grained way, e.g. users might like some specific pop artists, but dislikes other pop entertainers. We currently live in a constantly changing world, and the amount of personalization that can be predicted and implemented in advance is quite limited. This pre-training of device parameters cannot cover all possible acoustic scenes (databases) to provide user specific settings. Furthermore, new acoustic scenes might evolve (e.g. new music types), and the user might change his preference over time. Hence, we need to look at real-time online adaptive algorithms.

Many classification results are based on batch learning, where the training set with complete or partial labeling is static and given at the training time. State-of-the-art methods for classification are Support Vector Machines (SVM) [1] which have been successfully employed in genre/artist classification [2] as well as in active learning [3]. A SVM classifier is a kernel method which finds an optimal separating hyperplane of the data points x in a high-dimensional projection space $\Phi(x)$, without having to explicitly expand the data representation in this high-dimensional space. This computational advantage is achieved via the kernel trick, i.e., the scalar product in this high dimensional space assumes a functional form defined by the kernel. The large margin discriminatory hyperplane in this high dimensional space accounts for the success of SVM in batch classifiers.

In our scenario we deal with the user's preference. Here, batch algorithms are no longer applicable since data are not available in advance, nor can the system select data points from a known database for querying (e.g. in active learning). Furthermore, temporal preference changes can not be modeled via batch learning, since the time information is discarded.

Hence we need to consider a sequential online learning paradigm. In sequential online learning [4], the data are processed as a stream, and the algorithm updates the classifier parameters online. Here, we use two online-learning algorithms: an incremental SVM (LASVM) [5] and online passive-aggressive algorithms (PA) [6]. Both algorithms are discriminative kernel algorithms that learn sequentially. A drawback of kernelized online large margin methods is the tendency to invoke more and more support vectors which inevitably will violate the constraints on computational costs and space.

To maintain the advantages of kernelization without the space/computational violations, we introduce a infinite memory extension of kernel machines (specifically to PA) which integrates the advantages of kernelization with the memory space limitations.

2 BACKGROUND

In sequential learning, the data is accessible at times $t = 1, ..., T$, where the horizon T can be arbitrarily large. At each time step t, a new observation x_t is presented to the system, which predicts a label $f_t(x_t)$. After this prediction, the correct label y_t is revealed to the system which uses this information to update the classifier f_{t+1}. The new classifier f_{t+1} is used for the next time step $t + 1$.

This paper presents an extension of the passive-aggressive algorithm (PA) [6] for online preference learning and it compares it with an incremental SVM learning system (LASVM) [5]. Both algorithms learn discriminative classifiers in a sequential manner, i.e., LASVM optimizes a quadratic cost function, whereas PA implements gradient descent. Our variation of PA replaces the scalar product in the linear form of PA by a kernel expansion.

2.1 Incremental SVM

For comparison purposes, we consider an incremental version of SVM, the LASVM [5]. SVM optimization solves a quadratic programming problem for the dual objective function of large margin classifiers. One commonly used solver is the sequential minimal optimization (SMO) [7] which computes a τ-approximate solution. The LASVM implementation reorganizes the SMO sequential direction searches by considering pairs of examples (the new data point and existing support vectors (SVs)). In a second step, it repeats the search between pairs of SVs and eliminates redundant SVs.

We modified the code provided by the authors [1] to process the data in order of input. Simultaneously, we compute the cumulative error of the training in one epoch.

2.2 Online Passive-Aggressive Algorithms

Online passive-aggressive algorithms (PA) define a family of margin-based online learning algorithms [6] which are based on the hinge-loss for large margin classification

$$l(w;(x,y)) = \begin{cases} 0 & y(w \cdot x) \geq 1 \\ 1 - y(w \cdot x) & \text{otherwise.} \end{cases} \quad (1)$$

This constrained optimization is formulated as

$$w_{t+1} = \arg \min_{w \in \mathbb{R}^D} \frac{1}{2} ||w - w_t||^2 \text{ s.t. } l(w;(x_t, y_t)) = 0. \quad (2)$$

This procedure can be interpreted as follows: At each update step, if the current hyperplane misclassifies the current sample x (within a margin), then the hyperplane is adjusted to the next nearest hyperplane that resolves this problem. The nearest hyperplane is the one that deviates the least from the original hyperplane.

Algorithm 1 PA

```
Require: Input: aggressive parameter $C > 0$, initial classifier $w_1$
1: for t = 1, 2, ... do
2:     Predict: $\hat{y}_t = \text{sign}(w_t' x_t)$ for current data $x_t$
3:     Obtain $y_t$: loss $l_t = \max\{0, 1 - y_t(w_t' x_t)\}$
4:     Update:
           $\alpha_t = \min\left\{C, \frac{l_t}{||x_t||^2}\right\}$
           $w_{t+1} = w_t + \alpha_t y_t x_t$
5: end for
```

[1] http://leon.bottou.org/projects/lasvm/

The PA algorithm is outlined above. Its kernel counterpart is obtained by replacing the scalar product with a kernel function. The update step

$$w_{t+1} = w_t + \alpha_t y_t x_t \quad (3)$$

of the classifier function is replaced by

$$f_{t+1}(x) = \sum_{i=1}^{t} \alpha_i y_i K(x_i, x). \quad (4)$$

Comparing the Eq 3 and Eq 4, we see that the kernel formulation requires an explicit storage of the data points with $\alpha_i \neq 0$ ("SVs"). These SVs are accumulated during the learning phase and their number may grow arbitrarily large. Apart from violating storage constraints, a large number of SVs also leads to an increased computational cost for the prediction step, when the kernel function has to be evaluated for each of the many "SVs". Contrary to this growth of the computational complexity for kernel machines, the linear support vector machines (Eq 3) implicitly encode the SVs with the normal vector w_{t+1}, which leaves the storage requirements and the computational costs unchanged during update.

2.3 Limiting the Kernel PA

When the cache for the kernel storage is limited, two alternatives are available for discarding exemplars: removal of the oldest "SV", or removal of the worst performing "SV". By removing the oldest sample, we penalize the system where memory is needed. If the data shows cyclical behavior, then the "SVs" used to model a certain phase might have been discarded by the time this phase is revisited. In expulsion of the worst performing "SV", we discard "SVs", as advocated in [8]. If a loss is incurred, we find the "SV" for which the removal decreases the loss maximally:

$$\begin{aligned} &\arg\max_j y_t(f_t(x_t) - \alpha_j y_j K(x_j, x_t)) = \\ &\arg\min_j y_t \alpha_j y_j K(x_j, x_t) \end{aligned} \quad (5)$$

3 KERNEL EXPANSION FOR THE PASSIVE AGGRESSIVE ALGORITHM

Mercer's theorem states that a continuous, symmetric, positive semi-definite kernel function represents a scalar product in an appropriately high-dimensional space. Via the kernel trick [9], observations x_i of a non-linearly separable problem are mapped to a high-dimensional space $\Phi(x_i)$, such that these data becomes linearly separable. The scalar product in the linear function is replaced with the kernel function, such that it is not necessary to explicitly compute $\Phi(x)$:

$$K(x, z) = \Phi(x) \cdot \Phi(z). \quad (6)$$

Two commonly used kernels are the radial basis function (RBF) kernel

$$K(x,z) = e^{-\gamma||x-z||^2}, \tag{7}$$

where γ is the kernel width, and the polynomial kernel

$$K(x,z) = (\gamma x'z + c)^{\delta}, \tag{8}$$

for some scale factor γ, bias c and degree δ. For instance, the polynomial kernel emulates the scalar product of a finite induced feature space. Consider the polynomial kernel of degree 2, $K(x,z) = (x \cdot z')^2$. We obtain

$$\Phi(x) = (x_1^2, ..., x_D^2, \sqrt{2}x_1x_2, ..., \sqrt{2}x_dx_{d+j}, ..., \sqrt{2}x_{D-1}x_D) \tag{9}$$

where $d = 1, ..., D$, and $j = 1, ..., D-d$ indexes an exhaustive pairing of all features. This transformation yields a $D(D+1)/2$ dimensional space, i.e. $\Phi(x) \in \mathbb{R}^{D(D+1)/2}$.

We can now modify the algorithm PA to derive the algorithm passive-aggressive linear (expanded) PA-L-EX (Table3) which requires an additional input: function Φ which maps the input data to a different (higher) dimensional space induced by the kernel. Whilst this high dimensional representation is infeasible for batch-learning algorithms (dataset representation has to be stored for ALL data points which may be infeasible for large datasets), it is readily integrated in the online scenario. Since the data points are processed individually, this high dimensional expansion is calculated on the fly and, therefore, it is practical. In general, this extension holds for any kernel with a finite expansion for the induced feature space, or which admits some approximation of this induced feature space.

Algorithm 2 PA-L-EX

```
Require: Input: aggressive parameter $C > 0$, Expansion function $\Phi : \mathbb{R}^D \rightarrow \mathbb{R}^M, M \gg D$, initial classifier $w_1 \in \mathbb{R}^M$
1: for t = 1, 2, ... do
2:     Expand data $x_t \in \mathbb{R}^D : \Phi(x_t) \in \mathbb{R}^M$
3:     Predict: $\hat{y}_t = \text{sign}(w_t'\Phi(x_t))$ for current data $x_t$
4:     Obtain $y_t$: loss $l_t = \max\{0, 1 - y_t(w_t'\Phi(x_t))\}$
5:     Update:
           $\alpha_t = \min\left\{C, \frac{l_t}{||\Phi(x_t)||^2}\right\}$
           $w_{t+1} = w_t + \alpha_t y_t \Phi(x_t)$
6: end for
```

4 EXPERIMENTAL SETUP

Our experiments use a subset of the uspop2002[2] dataset. This subset of 18 pop artists 1 each with 5 or 6 albums is described in [3]. We use these authors' training and test set partition definition, which takes into account the producer's effect. The validation set has not been used during training. In one experiment, we combined the validation and training set to created a large 2-phase-training set to simulate user-preference change. Table 2 shows the statistics for the various database partitions.

[2] http://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html

Rolling Stones	Aerosmith	Beatles
Dave Matthews Band	Metallica	Queen
Fleetwood Mac	Garth Brooks	Madonna
Depeche Mode	Green Day	Roxette
Bryan Adams	Pink Floyd	Genesis
Creedence Clearwater Revival	Tina Turner	U2

Table 1. Artists in the 18-artist subset of uspop2002.

Set	Artist	Album	Song
`Train`	18	3×18	650
`Test`	18	2×18	448
`Valid`	9	1×9	102
`Phase1` $\subset$ Train	18	2×18	431
`Phase2` $=$ Valid $\cup$ Train$\setminus$Phase1	18	$2 \times 9 + 1 \times 9$	321

Table 2. Number of artists, albums and songs covered in each subset of the dataset. `Phase1` and `Phase2` data is a repartition of the combined `Train` and `Valid` sets.

4.1 Feature Extraction

We employed the 20 dimensional MFCC features distributed with the database (some songs were corrupt and discarded from the database), and we generated song level features. This grouping is achieved by vectorizing the sufficient statistics of the MFCC features per song, yielding a 230 dimensional input feature vector per song. We normalized[3] each feature dimension using the training data. High level features were not considered but can be readily integrated.

4.2 Verification and Simulation

To verify our dataset, we repeated the artist classification task outlined in [3] and we obtained the same accuracy rates.

For our experiments, we simulated random user profiles by splitting the 18 artists into two categories (like/dislike). The user likes a random subset of 3 to 15 artists, and dislikes the remaining artists. In this manner, we cover both balanced (same number of artists in each class) and unbalanced (one class having more artists) scenarios. For each

[3] For real-life scenario, approximate normalization factors can be estimated for a small dataset.

Kernel	$K(x,z)$	Test Acc
Linear	$x'z$	77.8 ± 6.6
Poly2	$(x'z)^2$	81.4 ± 5.2
RBF	$e^{-0.01*\|\|x-z\|\|^2}$	82.7 ± 4.7

Table 3. Results for batch SVM models. Hyper-parameters are optimized.

profile, we run 10 fold cross validation by using 10 permutations of the order in which the data is presented to the user. While the assumption of random selection might not correctly model the preferences of most users (preferences may occur in more structured groupings, e.g. mood or style), the random choice is certainly a difficult setting, since the classifier might be required to learn potential quirks.

4.3 Evaluation Measure

We consider two evaluation measure. For testing, we report accuracy rates $\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}(f(x_i), y_i) * 100\%$ where $\mathbf{1}$ denotes the indicator function. For online learning scenarios, we report the cumulative accuracy

$$CumAcc = \frac{1}{T}\sum_{t=1}^{T}\mathbf{1}(f_t(x_{t-1}), y_t) * 100\%, \qquad (10)$$

which is the accuracy during the learning phase. The incurred error $(100\% - CumAcc)$ indicates how often the user was required to correct the system by responding with the correct label.

5 EXPERIMENTAL RESULTS

We first consider static random user preferences by simulating 100 randomly sampled user profiles. The results for batch learning algorithms (Sec 5.1) are analyzed for comparison purposes, then we consider sequential learning with independent and identically-distributed (i.i.d.) drawing of the data (Sec 5.2). In Sec 5.3 we further analyze the situation when the data are no longer presented randomly, rather, they occur in cycles. Here, the simulations considered balanced preferences (liking 50% of the artists). Finally, in Sec 5.4 the users change their preferences after an initial training phase. During the second training phase some preferences are flipped, and the classifier should adapt for such opinion changes.

5.1 Batch Conditions

We first test the “best” case when all training data are known in advance. For this batch learning, we use a SVM classifier using libsvm[4]. Comparing the different kernel types,

[4] http://www.csie.ntu.edu.tw/cjlin/libsvm

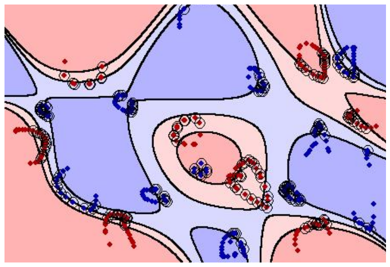
Figure 1. Illustration of SVM boundaries for RBF kernels. Circled data points are SVs. A high proportion of the observations serve as SVs.

Model	Kernel, Details	CumAcc	Test Acc
PA-L	Linear, D=230	75.0 ± 3.9	77.5 ± 5.7
PA-L-EX	Linear, Expand	80.8 ± 5.9	81.8 ± 5.7
PA-R200	RBF, 200 SV	77.5 ± 6.8	76.6 ± 7.6
PA-R	RBF, ∞ SV	81.6 ± 5.2	83.3 ± 5.1
LASVM	RBF, ∞ SV	80.8 ± 4.3	81.7 ± 4.6

Table 4. Descriptions of different incremental models and their performances during training (CumAcc) and for a hold out set (Test Acc). Sequential online learning on i.i.d. data. First three rows implement limited memory space. The last two rows are not truly online learners since no memory limitations is imposed on them.

the linear kernel performs poorly, whereas the RBF kernel shows the best performance, as can be seen in Table 3. The less frequently evaluated polynomial kernel, here with degree 2, almost matches the performance of RBF kernels.

Observing the models, we note that a large proportion of the training data was almost always retained within the model as support vectors, with almost all SVs located within the margin. For the RBF model, this $84.5 \pm 6.5\%$ of the training data were retained as SVs. This high proportion of SVs can be visualized as in Figure 1. New subcategories of data mark out new regions. As the amount of training data increases, it is possible that this growth in number of SV will level off. However, it is likely that new preferences (sub-categories) will require more SVs. Hence, for a fixed, batch training scenario, SVM may perform well, but for an equivalent online scenario, the algorithm may face resource problems such as infinitely large storage of SVs.

5.2 Sequential i.i.d. Conditions

To test the online learning under sequential i.i.d. conditions, we present the training data in a randomly permuted order. To ensure strict comparison, permutations were preserved across all systems, and 10 permutations were simulated for each different random preference.

Table 4 shows the descriptions and results for the various

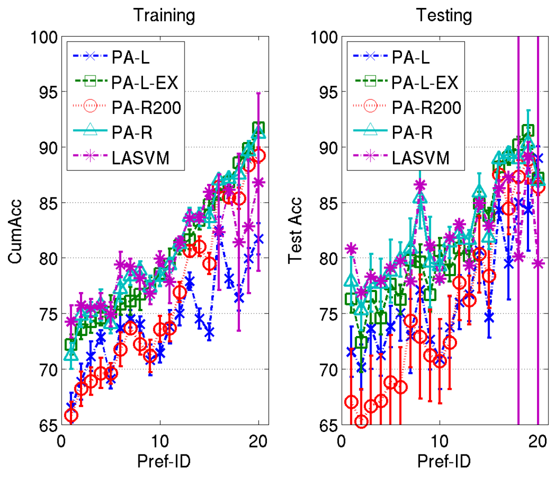

Figure 2. Detailed results on 20 randomly selected user profiles. Left: Train cumulative results. Right: Test accuracy on hold-out test set. Models are explained in Table 4.

models tested. The first three models are online sequential models which operate on a limited memory-storage, while the last two models (with ∞ SV) assume infinite storage space, and are not practical, and serve as a reference.

Comparing the performance sequential algorithms with that of the batch SVM classifiers (Table 3), we see that the performances are similar. LASVM attains a 81.7% performance which is 1.0% poorer than that of the SVM-RBF counterpart. PA-R even shows better performance at 83.3%. Similarly for the linear classifiers, PA-L performs 0.3% worse than that of the linear SVM.

While RBF-based algorithms are superior over the linear models for batch models, and also under impractical conditions of infinite memory, we see that this advantage no longer holds when memory is limited. Limiting the number of SV to 200SV (30% of the training set) as in PA-R200, we see a significant drop in performance, Instead, we observe that our approach of space expansion (PA-L-EX) combined with a linear classifier achieves similar performance levels to the kernel based classifiers with infinite storage.

Figure 2 demonstrates detailed results for 20 of the 100 random user profiles, both during training and on the hold-out test set. Note that while LASVM show slightly better results in the more difficult cases where accuracy is lower, it starts to show instable behavior when the sets are easy, resulting in large error bars. A detailed analysis shows that LASVM's stability depends on the sequence in which the data is presented. On the other hand, PA-algorithms are more stable for the same "unfavorable" sequences.

5.3 Sequential Non-i.i.d. Conditions

For common real world scenarios, data are no longer generated under i.i.d. conditions, rather, they show some patterns.

Model	Train CumAcc	Test Acc
PA-L	84.0 ± 1.6	65.7 ± 4.3
PA-L-EX	76.5 ± 1.5	73.1 ± 2.4
PA-R200	73.3 ± 2.6	59.1 ± 4.4
PA-R	79.7 ± 1.8	73.4 ± 4.2
LASVM	78.5 ± 2.0	78.5 ± 1.7
SVM (rbf)	batch	78.7 ± 1.7

Table 5. Performance when learning occur in an ordered manner (non-i.i.d.). Classes are balanced.

Artist ID	Phase 1	Phase 2	Test
a_1	likes	likes	likes
a_2	likes	likes	likes
a_3	likes	**dislikes**	**dislikes**
a_4	dislikes	**likes**	**likes**
a_5	dislikes	**likes**	**likes**
a_6	dislikes	dislikes	dislikes

Table 6. Simulation example for user preference change

For example, Christmas songs are usually popular only over Christmas, or some artist are more frequently played when they have released a new album. We have a learning phase that records the behavior of the user over a cycle, and hope that this preference is still represented during the evaluation cycle, where user feedback is no longer provided.

In this experiment, we have generated 10 random user profiles. Each user likes a random subset of 9 artists, resulting in a balanced class problem. The data is presented in phases, at each phase, only data from one "like" artist and one "dislike" artist is exhaustively presented, yielding a total of 9 phases during the training simulation.

Table 5 summarizes the results. Again, we show results for two online systems PA-L-EX and PA-R200. Results are also reported for PA-R and LASVM, but since they require infinite memory, they are impractical and serve as bounds.

Comparing the PA-L-EX and PA-R200 (limited storage), we see that the kernelized version does not perform well when data from previous phases are presented to the system during the evaluation phase, resulting in a low accuracy of 59.1%. PA-L-EX, on the other hand, still retains good classification accuracy, performing slightly worse at 73.1%. However, this almost equals the performance of a PA-RBF with infinite memory. We see that LASVM performs better, but is not feasible due to its requirement of large SV storage.

5.4 Adaptation to Preference Change over Time

In this experiment, we simulate a very adverse scenario. There are two phases during learning. During phase one, data from the set Phase1 are streamed i.i.d. to the system,

Model	Train CumAcc	Test Acc
PA-L-EX	75.8 ± 3.3	68.9 ± 3.6
LASVM	71.5 ± 1.6	61.1 ± 4.7

Table 7. Simulation for adverse (50%) preference change.

with the same 100 user profiles generated in the first experiment (i.i.d. data). Upon completion, data from set Phase2 are streamed with a 50% preference change. The 9 artists with 2 albums in the set Phase2 now have their labels (user preference) flipped. At the end of phase two, we test the final model on the test set, where the artist preference reflects that of phase two. Table 6 shows an example.

The performance of the two models LASVM and PA-L-EX are presented in Table 7. Recall that LASVM stores all the SVs that it accumulates. Its learning strength is greatly undermined by the change in user preference, which lead to conflicting SVs that greatly weakens the performance. As such, it breaks down and shows poor behavior (61.1% test accuracy). Even during the learning phase, it is already experiencing difficulties, degrading significantly from previous scenarios. PA-L-EX tries to accommodate for the new data by adjusting the hyperplane while trying to compromise for the previously learnt information. It shows weakened performance, but still retains information at 68.9%. This simulation models a severe 50% change in the classification classes, which takes its toll. Nevertheless, we notice that PA-L-EX does not break down which is in contrast to the complete failure of LASVM.

6 CONCLUSION

We have presented a user preference tracking system that exploits the advantages of using kernels for learning, without having to store potentially infinite numbers of support vectors. Our preference tracking system uses a simple algorithm PA-L-EX which can be efficiently implemented into small devices with limited storage and computational power. Despite these constraints, the algorithm has shown to be capable of recalling historically learnt events, and furthermore, adapt well to changes in concepts. These are necessary ingredients in adjusting and learning user preferences over time.

Our focus was primarily on song-level features, based on low level MFCC features. We experimented with 30 second music segments, which had slightly degraded performance (76.9% versus 81.8% test accuracy for i.i.d. case). Nevertheless, the results are encouraging given the simplicity of the features. There is potential for this algorithm to be used in combination with more sophisticated higher level features, and also online features.

We have addressed the case where the system learns and adjusts to the true label after each prediction. Future work could encompass the consideration of stream-based active learning ([10], [11]) where the data arrives in a stream, but labels must be procured. Other scenarios can include sparse feedback and inconsistent feedback.

Acknowledgements. This work is part of a project funded by KTI, Nr8539.2;2EPSS-ES.

7 REFERENCES

[1] N. Cristianini and J. Shawe-Taylor, Eds., *Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000.

[2] M. Mandel and D. Ellis, "Song-level features and support vector machines for music classification," *Proc. Int. Conf. on Music Information Retrieval*, pp. 594–599, Sept. 2005.

[3] M. Mandel, G. Poliner, and D. Ellis, "Support vector machine active learning for music retrieval," *Multimedia Systems, special issue on Machine Learning Approaches to Multimedia Information Retrieval*, vol. 12, no. 1, pp. 3–13, Aug. 2006.

[4] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning and games*, Cambridge University Press, 2006.

[5] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *Journal of Machine Learning Research*, vol. 6, pp. 1579–1619, 2005.

[6] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Schwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551–585, 2006.

[7] John C. Platt, *Advances in Kernel Methods: Support Vector Learning*, chapter Fast Training of Support Vector Machines using Sequential Minimal Optimization, pp. 185–208, MIT Press, 1999.

[8] Koby Crammer, Jaz S. Kandola, and Yoram Singer, "Online classification on a budget," in *NIPS*, 2003.

[9] M. Aizerman, E. Braverman, and L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning.," *Automation and Remote Control*, vol. 25, pp. 821 – 837, 1964.

[10] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Computational Learning Theory*, 1992, pp. 287–294.

[11] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," *Journal of Machine Learning*, vol. 28, pp. 133–168, 1997.