Budget-Driven Big Data Classification

No Author Given

No Institute Given

Abstract. A practical large-scale data classification approach is presented in this paper. By exploiting online learning framework, our approach learns a set of competing one-class Support Vector Machine models, one for each data class. The presented approach enjoys two budget-driven features: 1) it is capable of handling classification when data cannot fit in memory; 2) both training and labeling process is user controllable. Compared with the most popular big data classification tool, LibLinear, our approach is shown to be competent at processing extreme large data, while consuming a fractional of memory and time.

Keywords: Big Data, Budget-Driven Classification, Support Vector Machine (SVM), One-Class SVM (1SVM)

1 Introduction

We are in the era of big data - data that is both large and complex. As one of the main big data analytic tools, big data classification is used to support many applications in different scientific disciplines, e.g. economics, social networks, climate prediction, etc. There are three main computational challenges [25] in big data classification: (1) *volume*, which corresponds to the ever increasing amount of data. For example, Facebook reports about 6 billion new photos every month and 72 hours of video are uploaded to YouTube every minute [19]. It leads to the problem that the excessive data volume cannot fit in computer memory, especially for single commodity machines, whereas most of existing methods assume data can be stored in memory. (2) *velocity*, which means data is streaming in at unprecedented speed. To cope with that, online learning model is proposed to provide immediate response to the newly generated data, e.g. [23,6,16,20]. (3) *variety*, which refers to data diversity. Real data is often heterogeneous, coming in different types of formats and from different sources.

Motivated by the above challenges, we propose a practical One-Class Support Vector Machine (1SVM) approach, which is specifically designed for big data classification under a limited-budget environment. Grown out of the kernel classifier, i.e., SVM, the proposed approach inherits the flexibility of using kernels, which helps to address the issue of *variety*. The proposed approach incorporates online learning framework into 1SVM training to address the issue of *volume* and *velocity*. Training examples are read into memory in an online sequence, allowing reading and training happens at the same time. Hence, the

proposed approach only needs to allocate buffer for the current training example and the support vectors, rather than storing all data volume. Moreover, only dominant support vectors which influence decision boundaries are used, further saving memory. The training and labeling process are both user controllable. When a new example is observed during training, only a simple score function computation is required to update the 1SVM models, allowing our approach to react quickly to deal with data velocity. Users may suspend the training process anytime by stop feeding in the data, and the partial trained model can be used to perform classification while achieving reasonable accuracy.

Our experiments are performed on a real dataset used in the well-known Yahoo! Large-scale Flickr-tag Image Classification Grand Challenge [1]. The dataset exhibits the three challenges of big data classification. By iteratively increasing problem size, the proposed approach achieves superior classification performance on extremely large data, compared with two other sate-of-the-art methods.

The rest of the paper is organized as follows. Related works are reviewed in Section 2. Section 3 presents the details of the proposed approach. Experimental results and comparisons to existing methods are provided in Section 4. Finally, Section 5 concludes the paper.

2 Related Work

There is a vast body of existing work in the areas of big data classification. Here the most related ones are discussed in the subsections below.

2.1 Budget-Driven Classification

Linear classification models have been shown to handle big data classification well. Existing work on training linear models under a limited-budget environment can be categorized into memory-driven [26,5,3,28] and time-driven [15,23,19] approaches. Memory-driven approaches focus on solving the problem when data cannot fit in memory. LibLinear [10] is the most popular solver for linear SVM. Yu *et al.* [26] proposed a block optimization framework to handle the memory limitations of LibLinear. They splits data into blocks and stores them in compressed files. By training on a block of samples at a time, they save required training memory. Following [26], various block-optimization based variants are also proposed [5,3,28].

Time-driven approaches try to solve the problem in linear time. Joachims [15] proposed SVMPerf by reformulate the original SVM function to a structural SVM, achieving linear time complexity for sparse training features. Unlike SVMPerf and many other methods, which formulate SVM as a constrained optimization problem, Shalev-Shwartz *et al.* [23] relies on an unconstrained optimization formulation of linear SVM and achieves liner training time. Very recently, Nie *et al.* [19] proposed a new primal SVM solver with liner computational cost for big data classification.

Another way to scale up big data classification is through parallelism, such as [22,14,7,12]. However, the development of parallel classification is limited by not only parallel program implementation difficulties but also by data synchronization/communication overheads on distributed systems. More recent advances of big data classification is surveyed in [25,27]. Different from most existing approaches, which focus on solving linear SVM, our approach relies on a competing 1SVMs model, and is designed on a single commodity machine. In addition, our approach is not constraint to linear kernel, making it easily customizable for handling a variety of data.

2.2 Online vs. Batch Learning

Batch learning has been the standard methods for data classification [9,8,21,11]. When employing batch learning for large-scale datasets, one often has to fight with a number of bottlenecks such as memory issue and computational costs. One notable exception is [4], where a scalable batch learning methods is proposed. Nevertheless, it requires complex speeding-up techniques such as disk swapping and chunking, which unfortunately also introduce quite a few tuning parameters. This is in sharp contrast to online learning methods, such as [17,23,16,20], that are usually very simple and efficient. Moreover, 1SVM [6,13] is shown to work well with large-scale and dynamic data under online setting.

3 Proposed Approach

The budget-driven approach is presented in this section. Grown out of 1SVM training model, the proposed approach possesses two distinct features for limited-budget classification: controllable training time and low memory requirement.

3.1 Competing 1SVM

Given a set of examples that can either belong to or not belong to a given class, 1SVM uses a hypersphere to surround some of the examples. Examples inside the hypersphere are considered as inliers and otherwise outliers. Unlike the existing 1SVM training that is designed for handling one class classification problem, our approach builds upon the Competing 1SVM (C-1SVM) model proposed in [13], where two 1SVMs are trained to solve a binary classification problem. We here extend the C-1SVM for multiclass classification and apply it to big data. The key idea is to train and maintain multiple 1SVMs, each models training data distributions from a given class. Each 1SVM may label a testing example as inlier or outlier independently, and hence *compete* with either other. A testing example is *jointly* labeled by those 1SVMs. Compared with the SVM model using hyperplanes to split data, modeling different classes separately using multiple 1SVMs produces hypersphere boundaries as shown in Figure 1.

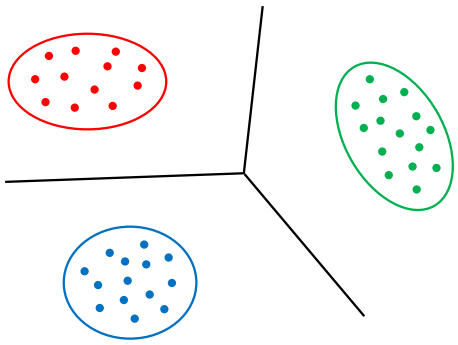


Fig. 1. Comparison between multiclass SVMs and C-1SVMs. Red, green and blue dots represent training instances from three different classes, respectively. The straight line indicates the decision boundary of the multiclass SVMs, whereas the ellipsoids show the boundaries of the three C-1SVMs. Different from multiclass SVMs which use hyperplanes, C-1SVMs uses hyperspheres to capture the structures of different classes.

3.2 Budget-Driven Online Learning Model

Given a large set of examples, the conventional 1SVM training minimizes the volume of the hypersphere while at the same time includes as many "in-class" examples as possible, which is a classical batch learning problem. When dealing with big data problems, optimization in batch 1SVM learning often has to address memory issue and expensive computational costs. While C-1SVMs used in this paper has to train multiple models for different classes, it requires *much more* memory. To alleviate these bottlenecks on big data, we propose to employ online learning in C-1SVMs training. Our approach inherits the following distinct advantages of online learning for budget-driven problems. Different from batch learning that considers examples in a batch mode, examples are observed by the learner one by one in a time sequence under online learning. The online learner is gradually refined based on its current partial model and the newly observed example. The training can terminate anytime when the user stops presenting examples, and the partially trained model can be used to perform classification. As a result, training time is user controllable under online learning, which is promising when trying to classify big data under limited training time while still aiming for good performance. Furthermore, our approach can easily handle dynamic data such as video stream and online user generated webdata, etc. When a new training example (e.g. a new image uploaded to social media) is observed, the minimization function of the batch 1SVM is changed, and thus it should be solved again to obtain the new solution. Hence, batch learning needs to start over, which is time-consuming. In comparison, by incorporating online learning into C-1SVM model, we need to refine the current model using the newly observed example, which only involves a simple computation of score function, as shown in Algorithm 1.

Grown out of the online learner in [13], our approach differs from it in several ways to fit our demand: budget-driven big data classification. First, all examples are shown repetitively to the online learner in [13] and it requires a large number of iterations to converge. In contrast, we believe redundancies exist in big data, and competitive classification accuracy can be obtained by considering only a portion of training examples. In each iteration, a randomly selected example is read, based on which the C-1SVM is updated. Reading data and training happen at the same time in our approach, allowing to only store one training example and sparse support vectors of each class during training. Notice that the algorithm in [13] reads all data into memory since examples are inputted in multiple rounds. Hence, our approach is particularly useful on big data classification when data cannot fit in memory. Experimental results in Figure 3 also show that the presented approach can converge in only one-round data reading when the problem size is large enough, which further confirms our assumption of redundancies in big data. Second, as the training set becomes larger, the number of support vectors increase significantly, especially when we train multiple C-1SVMs for a big data problem. Storing all these support vectors would require vast memory. To further reduce required memory for training, our approach assumes the hyperspheres of C-1SVMs are primarily determined by the *dominant* support vectors, i.e. those with high supporting weights. In practice, we set the number of support vectors of each class as a fixed value and store only the most dominant support vectors in memory. Compared with the conventional 1SVM algorithm that keeps all support vectors, our approach achieves competitive performance as reported in Figure 2. Moreover, controlling the number of support vectors helps reduce training time in each iteration. By only keeping dominant support vectors, the model refinement in each iteration would be more efficient as reflected from the summation in Equation 1.

3.3 The Training Algorithm

The training algorithm of our approach is presented in Algorithm 1, which is very easy to implement. When a new example and its label $\{x_t, y_t\}$ is observed/read at time t, our approach first evaluates the score of x_t based on the existing support vectors from the same class and computes the weight of x_t. Following the decision function of SVM, the score function is defined as:

$$f_t(x_t) = \sum_{j=1}^{n} w_j \chi\left(S_{y_t}(j) \neq x_t\right) K(S_{y_t}(j), x_t), \tag{1}$$

and its supporting weight:

$$w_t = \max\left(0, \min\left(\frac{\gamma - f_t(x_t)}{K(x_t, x_t)}, C\right)\right), \tag{2}$$

where S_{y_t} is the support vector set of the class y_t, w_j is the weight of the j_{th} support vector in S_{y_t}. $\gamma := 1$ is the margin, and $\chi(\cdot)$ is an indicator function

Algorithm 1 Budget-driven C-1SVMs

Input: training examples with corresponding labels $\{x_t, y_t\}$, kernel function $K(\cdot,\cdot)$, cut-off value C, support vector buffer size n
Output: support vector set S

Initialize each S_i as an empty set for each i_{th} class
repeat
  randomly read an example (x_t, y_t) at time t
  compute score $f_t(x_t) \leftarrow \sum_{j=1}^{n} w_j \chi\left(S_{y_t}(j) \neq x_t\right) K(S_{y_t}(j), x_t)$
  $w_t \leftarrow \max\left(0, \min\left(\frac{\gamma - f_t(x_t)}{K(x_t,x_t)}, C\right)\right)$
  if x_t already exists in S_{y_t} **then**
    update the weight of x_t with w_t
  else if $w_t >$ the minimal weight in S_{y_t} **then**
    replace the SV with minimal weight in S_{y_t} with $\{x_t, w_t\}$
  end if
until User Termination or S keep unchanged for T times

with $\chi(true) = 1$ and $\chi(false) = 0$. $K(\cdot,\cdot)$ is the kernel function, and different kernels can be used for different applications. Duplication is common in big data. Users may upload the same image to social media. In the conventional online learning, all duplicates are added into support vector sets as long as their supporting weights are large enough. These duplicate support vectors come from the same example but have different weights. In contrast, when observing a duplicate example x_t that is already in the corresponding support vector set, the proposed approach computes the score with the duplicate x_t excluded and the original weight of x_t is substituted by w_t. By summing the supports of the remaining support vectors, $f_t(x_t)$ can better shows how well the current model can predict x_t. The supporting weight w_t is computed by comparing the margin and the score. Training examples can be corrupted by different label noises. Assigning a large weight on the corrupted examples would increase the chance of adding them into support vector sets, thus distorting the decision boundary. To address this, a cut-off value C is used to bound w_t, thus limiting the effects of label noise. $C = 0.5$ is used in our implementation. Finally, if the weight w_t of the newly observed x_t turns out larger than the minimal weight in S_{y_t}, the support vector with the minimal weight is replaced by $\{x_t, w_t\}$.

With the benefits of online learning, the training algorithm can be terminated by users anytime when they stop feeding in training examples. Users can also selectively evaluate the partially trained model using a validation dataset to check its effectiveness. Besides user controllable termination, our algorithm itself can converge fast but still achieve competitive accuracy. Following the convergence condition used in the conventional 1SVM training, the proposed approach thinks the training algorithm converges until support vector sets keep unchanged for at least $T = 100$ times while different new examples are observed. Labeling process is straightforward in our approach, where an example is labeled as the class with the highest score based on Equation 1.

4 Experiments

In this section, we evaluate performance of the proposed approach for data classification with limited budget on a large-scale social media dataset, and compare it with previous approaches in the literature. Besides, detailed analysis on the effect of the proposed budget-driven features are presented. We have implemented the proposed approach in C++ and experiments were run on a Intel Core i5 (3.20GHz) machine with 4GB RAM.

4.1 Datasets

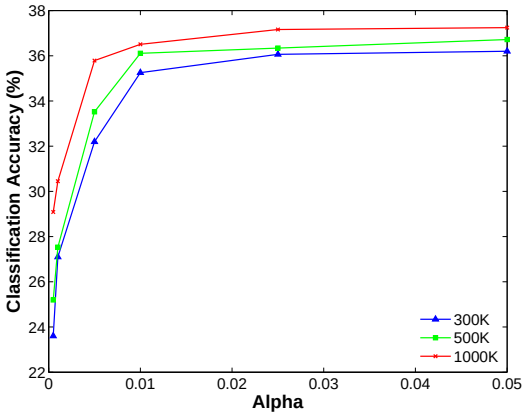
Table 1. Summary of the Flickr-tag dataset

# Train Instances	# Test Instances	# Classes	# Features
1,350,000	450,000	9	400

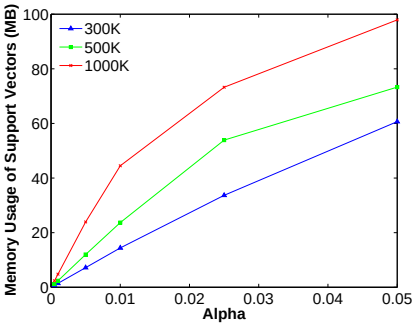
The Flickr-tag image dataset summarized in Table 1 is extracted from the one provided by Yahoo! Large-scale Flickr-tag Image Classification Grand Challenge [1,2] in ACM Multimedia 2013, It is also the data we found that is challenging enough to evaluate a classification algorithm towards the three aforementioned properties of big data. The main challenge of this dataset is that it consists of 10 classes (e.g. food, people) with $200K$ images for each class, and large intra-class visual diversity exists so that some images from the same class might not share any visual similarities. Furthermore, all tags are annotated by Flickr users, which are quite noisy. Following [24] and [18], instances tagged with "2012" are removed in the dataset used in this paper since the tag "2012" is given based on the taken time of images and is unrelated to visual information. Unlike previous works on the Flickr-tag dataset which concentrate on exploring multiple image features, we are interested in classification under limited budget and the provided bag-of-words feature representation is used in our implementation so that we can evaluate different classification techniques on the same ground.

4.2 Evaluation of Support Vector Number

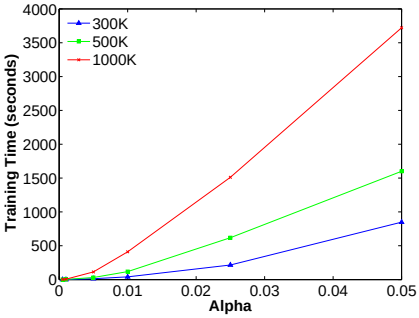
We first evaluate the influence of parameter n, the support vector buffer size. Without explicitly setting a common n for different classes, we observe that the optimal buffer size is related to the number of examples in each class. Denote the ratio of examples selected as support vectors as α. For the i_{th} class, we set the support vector buffer $n_i = \alpha N_i$ in our experiments, where N_i is the number of the observed examples belonging to the i_{th} class. Figure 2 plots the classification accuracy of our approach under different α values on the Flickr-tag dataset. Keeping only a small number of support vector in the buffer produces lower accuracy, since the limited number of support vectors cannot capture complex



(a) Classification Accuracy



(b) Memory Usage of SVs



(c) Training Time

Fig. 2. Evaluation of support vector number on three sets of $300K$, $500K$ and $1000K$ training examples from the Flickr-tag data. Figure (a)-(c) show the classification accuracy, memory usage of support vectors and training time used under different α values, respectively.

example distributions. As the α value increases, the classification accuracy improves rapidly at first and then levels off. It is worth noting that when using a large α value, e.g. 0.05, our approach becomes the conventional 1SVM training, because the support vector buffers are large enough to store all support vectors. Our approach with a smaller α value can achieve close performance to the conventional 1SVM training, while only uses a fraction of support vectors and saves time and memory, as shown in Figure 2. It confirms our assumption that the decision boundaries of C-1SVMs are mainly influenced by dominant support vectors, while competitive performance can still be achieved no matter how large the training size is. While users can set any α value based on their memory limitation, we observe that setting $\alpha = 0.01$ provides a good trade-off regardless the training size. Hence, we use $\alpha = 0.01$ for evaluation in the following experiments.

4.3 Comparisons with Other Approaches

In this subsection, we compare the performance of our approach with LibLinear and the KNN classifier. LibLinear [10] is the most popular solver for large-scale data classification, and its extension [26] can handle data that cannot fit in memory using a block optimization method. The KNN classifier is used in the recent work [24] and is shown to be efficient on the Flick-tag prediction task.[1].

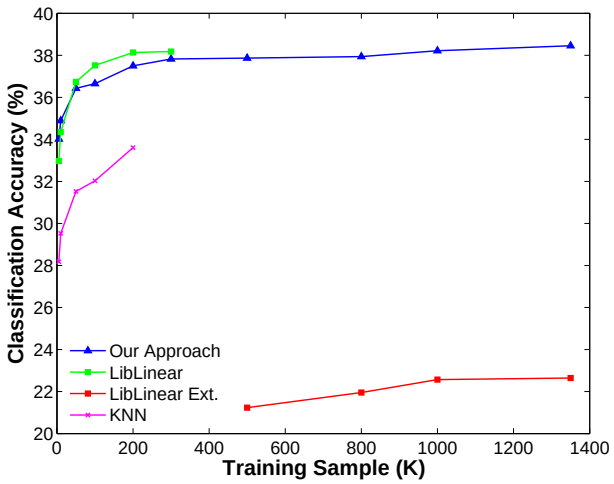


Fig. 3. Comparisons between the proposed approach and LibLinear, KNN classifier. As the training set is increasing, KNN classifier and LibLinear can handle $200K$ and $300K$ examples at most because of memory limitation, respectively. In contrast, our approach can achieves superior performance than the extension of LibLinear in large sets.

To evaluate budget-driven features of our approach, performance on different problem sizes are compared. By increasing the training set size, we compares our approach with LibLinear and KNN classifier, as shown in Figure 3. We use the original LibLinear for $< 500K$ dataset. When data cannot fit in memory, the extension of LibLinear is used for $\geq 500K$ dataset. Our approach achieves similar accuracy to LibLinear on small datasets, and outperforms LibLinear's extension on large datasets. Figure 3 displays the following budget-driven features of our approach: First, the training process is user controllable. Users can terminate the training process anytime because of their time/memory limitation, and the partially trained model can still obtain reasonable accuracy. Second, the accuracy of our approach reaches stable with only $300K$ examples observed. Reading more examples would not obviously improve performance, which confirms our

[1] Note that our results of KNN method are different from the one reported in [24], since they are using multi-features

assumption of redundancy of big data, and a fraction of instances can produce encouraging results for our online C-1SVM approach when training set is large enough.

Besides achieving promising classification accuracy on large datasets, our approach can converge fast and save vast memory. Figure 4 compares memory usage and training time of our approach with that of LibLinear extension, where only $\geq 500K$ datasets are used to illustrate the effectiveness of our budget-driven approach on extreme large datasets. It is worth noting that LibLinear extension needs to split training set into blocks, whereas data splitting is time-consuming. Default parameters are used for LibLinear extension, where dataset is split into 8 blocks. By observing training data in different sequences, results in Figure 3 and 4 are averaged on 5 runs.

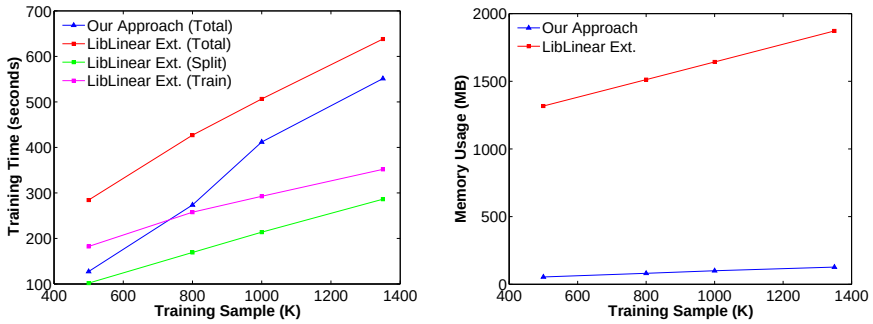


Fig. 4. Processing seconds memory usage on large subsets. Left and right side show comparisons on training time and memory usage, respectively. The processing time of LibLinear extension consists of data splitting and training.

5 Conclusions

A practical big data classification approach is proposed in this paper that is able to work under limited-budget environment. The proposed approach trains multiple C-1SVM models for different classes, one for each class. By employing online learning, the proposed approach is user controllable, capable of handling dynamic data, and can react quickly towards the newly observed examples. In terms of computational resources needed, our approach only requires keeping a fractional of examples in memory as support vectors, and the training can converge in as few as just one round of data-reading. In addition, only the most dominant support vectors are used in our approach, further saving memory. Experiments on a challenging social media dataset demonstrate that our approach possesses superior performance comparing to the state-of-art approaches LibLinear, especially on large-scale data.

When a new training example arrives, the score function (Equation 1) compares it with all support vectors stored. This process is parallel friendly. In the future, we would like to explore the possibility of implementing our technique on GPUs to further reduce the training time needed.

References

1. Yahoo! large-scale flickr-tag image classification grand challenge. http://acmmm13.org/submissions/call-for-multimedia-grand-challenge-solutions/yahoo-large-scale-flickr-tag-image-classification-challenge/
2. Yahoo! webscope dataset ydata-flickr-ten-tag-images-v1_0. http://webscope.sandbox.yahoo.com/catalog.php?datatype=i
3. Blondel, M., Seki, K., Uehara, K.: Block coordinate descent algorithms for large-scale sparse multiclass classification. Machine learning 93(1), 31–52 (2013)
4. Boullé, M.: A parameter-free classification method for large scale learning. The Journal of Machine Learning Research 10, 1367–1385 (2009)
5. Chang, K.W., Roth, D.: Selective block minimization for faster convergence of limited memory large-scale linear models. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 699–707. ACM (2011)
6. Cheng, L., Gong, M., Schuurmans, D., Caelli, T.: Real-time discriminative background subtraction. Image Processing, IEEE Transactions on 20(5), 1401–1414 (2011)
7. Chu, C., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G., Ng, A.Y., Olukotun, K.: Map-reduce for machine learning on multicore. Advances in neural information processing systems 19, 281 (2007)
8. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. The Journal of Machine Learning Research 2, 265–292 (2002)
9. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. arXiv preprint cs/9501101 (1995)
10. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. The Journal of Machine Learning Research 9, 1871–1874 (2008)
11. Gao, T., Koller, D.: Multiclass boosting with hinge loss based on output coding. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11). pp. 569–576 (2011)
12. Ghoting, A., Krishnamurthy, R., Pednault, E., Reinwald, B., Sindhwani, V., Tatikonda, S., Tian, Y., Vaithyanathan, S.: Systemml: Declarative machine learning on mapreduce. In: Data Engineering (ICDE), 2011 IEEE 27th International Conference on. pp. 231–242. IEEE (2011)
13. Gong, M., Cheng, L.: Foreground segmentation of live videos using locally competing 1svms. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 2105–2112. IEEE (2011)
14. Graf, H.P., Cosatto, E., Bottou, L., Dourdanovic, I., Vapnik, V.: Parallel support vector machines: The cascade svm. In: Advances in neural information processing systems. pp. 521–528 (2004)
15. Joachims, T.: Training linear svms in linear time. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 217–226. ACM (2006)

16. Langford, J., Li, L., Zhang, T.: Sparse online learning via truncated gradient. In: Advances in neural information processing systems. pp. 905–912 (2009)
17. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. Information and computation 108(2), 212–261 (1994)
18. Mantziou, E., Papadopoulos, S., Kompatsiaris, Y.: Scalable training with approximate incremental laplacian eigenmaps and pca. In: Proceedings of the 21st ACM international conference on Multimedia. pp. 381–384. ACM (2013)
19. Nie, F., Huang, Y., Wang, X., Huang, H.: New primal svm solver with linear computational cost for big data classifications. In: Proceedings of the 31st International Conference on Machine Learning (ICML) (2014)
20. Rai, P., Daumé III, H., Venkatasubramanian, S.: Streamed learning: One-pass svms. In: IJCAI. vol. 9, pp. 1211–1216 (2009)
21. Rifkin, R., Klautau, A.: In defense of one-vs-all classification. The Journal of Machine Learning Research 5, 101–141 (2004)
22. Shafer, J.C., Agrawal, R., Mehta, M.: Sprint: A scalable parallel classifier for data mining. In: Proceedings of the 22th International Conference on Very Large Data Bases. pp. 544–555. VLDB '96, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1996)
23. Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A.: Pegasos: Primal estimated sub-gradient solver for svm. Mathematical programming 127(1), 3–30 (2011)
24. Su, Y.C., Chiu, T.H., Wu, G.L., Yeh, C.Y., Wu, F., Hsu, W.: Flickr-tag prediction using multi-modal fusion and meta information. In: Proceedings of the 21st ACM international conference on Multimedia. pp. 353–356. ACM (2013)
25. Tong, H.: Big data classification. Data Classification: Algorithms and Applications p. 275 (2014)
26. Yu, H.F., Hsieh, C.J., Chang, K.W., Lin, C.J.: Large linear classification when data cannot fit in memory. ACM Transactions on Knowledge Discovery from Data (TKDD) 5(4), 23 (2012)
27. Yuan, G.X., Ho, C.H., Lin, C.J.: Recent advances of large-scale linear classification. Proceedings of the IEEE 100(9), 2584–2603 (2012)
28. Zhang, K., Lan, L., Wang, Z., Moerchen, F.: Scaling up kernel svm on limited resources: A low-rank linearization approach. In: International Conference on Artificial Intelligence and Statistics. pp. 1425–1434 (2012)