Automatic Detection of Learnability under Unreliable and Sparse User Feedback

Y. Moh, W. Einhäuser, and J. M. Buhmann

Institute of Computational Science
Swiss Federal Institute of Technology (ETH) Zurich
{tmoh,einhaeuw,jbuhmann}@inf.ethz.ch

Abstract. Personalization for real-world machine-learning applications usually has to incorporate user feedback. Unfortunately, user feedback often suffers from sparsity and possible inconsistencies. Here we present an algorithm that exploits feedback for learning only when it is consistent. The user provides feedback on a small subset of the data. Based on the data representation alone, our algorithm employs a statistical criterion to trigger learning when user feedback is significantly different from random. We evaluate our algorithm in a challenging audio classification task with relevance to hearing aid applications. By restricting learning to an informative subset, our algorithm substantially improves the performance of a recently introduced classification algorithm.

1 Introduction

Sound classification according to user preferences poses a challenging machine learning problem under real world scenarios. In the realm of hearing instrument personalization, additional challenges arise since hardware considerations constrain processing and storage capacity of the device, while the acoustic environment may undergo substantial changes. Furthermore, user feedback is typically sparse, and individual preferences may shift. Finally, the device shall be operative without initial individual training (default settings), learning shall only smoothly affect settings, and adaptation may only occur as consequence of user feedback. Consequently, a "smart" hearing instrument, which is envisioned to adapt its amplification settings according to user preferences and acoustic environment, needs to employ a sophisticated online adaptation algorithm that is passive, efficient and copes with sparse labels.

The key to enable adaptivity is to decide when learning shall take place. At the one extreme, one may assume that the hearing instruments are exclusively factory-trained with no adaptation to idiosyncratic preferences [1]. At the other extreme, continuous user feedback would be required, which is infeasible in practice. Using the sub-problem of music categorization into like/dislike, we here develop an algorithm that fulfills the aforementioned constraints. It decides based on a statistical criterion when learning shall take place by evaluating the consistency of user feedback. The criterion endows the learning algorithm with a confidence measure in the newly provided labels. This test enables efficient online learning of user preferences under real-world conditions.

A recent study [2] proposed to extend additive expert ensembles [3] to semi-supervised learning via learning with local and global consistency (LLGC) [4]. These results provided the proof-of-concept that this approach works the current problem. In particular the system of [2] learns only when feedback is made available *and* its current ensemble of classifiers does not suffice to model the preference indicated by the feedback. However, this earlier model suffered from several shortcomings. Most importantly, the system continues to learn even when the novel information is inconsistent and, therefore, unusable and when classifier and/or features are ill-suited for the proposed separation. Here we propose an algorithm that decides whether new labels should be ignored or learning should take place. A confidence score over the user feedback is computed based on semi-supervised graphs with randomized labels. This score determines whether the labels are consistent and thus learnable. Learnability, i.e. self-consistency of the labels, is established by determining the its deviation from the random case..

2 Background

In semi-supervised learning, a set of data points $X = X_L \cup X_U$ is available, where labels Y_L for X_L are provided, and X_U is the unlabeled dataset that we wish to label or exploit for learning. Graphs provide an excellent means to code manifold structures which can be exploited for the purpose of semi-supervised learning [5].

Formally, we are presented with a set of n data points $X = X_L \cup X_U = \{x_1, ..., x_l, x_{l+1}, ..., x_n\}$, and $x_i \in \mathbb{R}^m$ is a m-dimensional feature vector representing the i-th data point. For the first l data points, we have the corresponding labels $Y_L = \{y_1, ..., y_l\}$, where $y_i \in \{-1, 1\}$. Let $F(0) = \{y_1, ..., y_l, 0, ..., 0\}$ be the vector that represents the labels for the data points, where $y_i = 0$ for $l < i \leq n$. Furthermore, let $g(x_i, x_j)$ be any similarity measure between two points x_i, x_j. In our experiments, we use $g(x_i, x_j) = \exp(-||x_i - x_j||^2/2\sigma^2)$.

Graph Notations: We code the data as a graph $G = (X, W)$. The nodes X represent individual data points, and the edges are coded in an affinity matrix W. W stores the similarity between data points, i.e. $W_{ij} = g(x_i, x_j)\ \forall i \neq j$, and $W_{ii} = 0$. The normalized graph Laplacian $\mathcal{L}$ is defined as

$$\mathcal{L} = D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \text{ with } D_{ii} = \sum_j W_{ij}. \tag{1}$$

Learning with Local and Global Consistency (LLGC)[6]: LLGC is a graph based semi-supervised learning technique that spreads the label information from X_L to the X_U based on the affinity of the data points. It does this via the normalized graph Laplacian, and mathematically, the iterative information spreading is:

$$F(t+1) = \alpha \mathcal{L} F(t) + (1-\alpha) F(0), \tag{2}$$

where α is a suitably chosen learning rate. This iteration converges to the solution

$$F^* = (1 - \alpha)(I - \alpha\mathcal{L})^{-1} F(0), \tag{3}$$

as can easily been seen from the condition $F(t + 1) = F(t)$.

The LLGC solution F^* can be interpreted as node weight after Y_L has been propagated across the graph. Due to the smoothness constraints, reliable labels should reinforce each other, resulting in higher node weights, whereas labels showing inconsistencies tend to cancel out, resulting in lower node weights.

Here, we endeavor to find a measure of quality for the solution obtained by LLGC. This criterion will allow us to decide if the label information provided is useful for our classification task in an unsupervised manner. We propose to measure the *reliability* of the LLGC solution, and the *learnability* of the provided labels.

3 Algorithm

Our algorithm is presented with a set of data X, for which a fraction q is labeled by the user ($Y_{L,usr}$). The aim is to decide, whether or not this scenario is learnable. To achieve this goal, the algorithm measures the quality of the LLGC solution on $(X, Y_{L,usr})$ by comparing it with solutions on K sets of randomly drawn labels on the same data $(X, Y_{L,k}), 1 \le k \le K$.

The interplay of two major factors affect the solution of LLGC:

1. The topology of the graph (coded by the affinity matrix W).
2. The labels provided in $Y_{L,k}, k \in \{usr, 1, ..., K\}$.

Each factor on its own influences the final configuration of the graph: The topology appears in the graph Laplacian $\mathcal{L}$ term, and the labels are $F(0)$.

$$F^* \propto \underbrace{(I - \alpha\mathcal{L})^{-1}}_{\text{Topology}} \underbrace{F(0)}_{\text{Labels}} \ . \tag{4}$$

We denote each configuration by $H_k = (G, Y_{L,k}, F_k^*)$, where $G = (X, W)$ represents the data points, $Y_{L,k}$, is the set of labels provided for a subset (X_L) of the data points, and F_k^* is the solution of LLGC. The set of source (emitting) nodes (X_L) has to be kept constant to ensure comparability of hypotheses, since the graph is weighted and the topology plays an essential role in the label-spreading.

To measure learnability, we need to quantify the quality of H_{usr} as compared to the average over H_k. The weight distributions corresponding to H_k are $F_k^* = [f_{k,1}, ..., f_{k,n}]$, for $k \in \{usr, 1, ..., K\}$, where n is the total number of nodes. We compare F_{usr}^* to all other F_k^* by computing the absolute node weight difference for each node X_i in H_{usr} to its corresponding counterpart in H_k, given as

$$d_{usr,k}(i) = |f_{usr,i}| - |f_{k,i}|. \tag{5}$$

Similarly, we compute d_{k_u,k_v} for all pairs, $1 \le k_u < k_v \le K$. Comparing the statistics of $d_{usr,k}$ to d_{k_u,k_v} allows us to quantify the significance of $d_{usr,k}$. Hence we have reduced the test of learnability of H_{usr} to a test of statistical significance on d.

We outline the statistics that we generate over d. We compute the mean μ_s and standard deviations σ_s of the s-th percentile values. For robustness considerations (sensitivity to outliers), we base our statistical test on the properties of a fixed quantile of the distribution of D. In this study, we focus on the 90-th percentile ($t = 0.9$), but also examine other percentiles (median, $t = 0.5$) and other statistics (mean) (Sec. 5). For the H_{usr}, we compute $\mu_{usr,t}$ t-th percentile:

$$\mu_{usr,s} = \frac{1}{K}\sum_{k=1}^{K} Q_s(d_{usr,k}), \tag{6}$$

where $Q_s(d)$ computes the s-th percentile value of vector d. For the random drawings, $\mu_{rnd,s}$ is

$$\mu_{rnd,s} = \frac{2}{K(K-1)}\sum_{k=1}^{K}\sum_{j=k+1}^{K} Q_s(d_{k,j}). \tag{7}$$

$\sigma_{usr,s}$ and $\sigma_{rnd,s}$ are correspondingly computed. If K is sufficiently large, we can approximate $\mu_{rnd,s}, \sigma_{rnd,s}$, by an appropriate subset, e.g.

$$\mu_{rnd,s} = \frac{1}{K}\sum_{k=1}^{K} Q_s(d_{k,k\oplus 1}), \tag{8}$$

where $\oplus$ denotes addition modulo K. H_{usr} is defined to be learnable iff $(\mu_{usr,s}, \sigma_{usr,s})$ is significantly different from $(\mu_{rnd,s}, \sigma_{rnd,s})$.

Algorithm: Below, we formally present our algorithm which applies LLGC as a subroutine. As input, the algorithm is presented with $G = (X, W), Y_{usr,L}$. The percentile level used here is $s = 0.9$, and $K = 100$.

1. Generate the K random shufflings of $Y_{usr,L}$ (same label distributions and node subset X_L) to obtain $Y_{k,L}$ for $1 \leq k \leq K$.
2. Compute $F^*_\cdot$ using LLGC to obtain $H_{usr} = (G, Y_{usr,L}, F^*_{usr})$ and $H_k = (G, Y_{k,L}, F^*_k)$ respectively[1].
3. Calculate $d_{\cdot,\cdot}$, which are used to compute $\mu_{usr,s}$,$\sigma_{usr,s}$, $\mu_{rnd,s}$ and $\sigma_{rnd,s}$ for a desired percentile t.
4. Accept H_{usr} for learning if $\mu_{usr,s} > \mu_{rnd,s} + \sigma_{rnd,s}$. Otherwise reject.

[1] The cost of computing the random probes creates only negligible overhead compared to computing the original LLGC algorithm. Hence this step has only minimal impact on overall runtime.

4 Experimental Setting

In this section we evaluate the performance of our algorithm to test for learnability. We base our experiments on two standard music datasets. We simulate different user profiles, i.e., their individual preferences and feedback consistency. Preference defines the decision boundaries, whilst the consistency refers to the noise level in the feedback (labels $Y_{L,usr}$).

Data: We used two distinct music audio datasets, one comprising predominantly classical music ("`classic`"), and another dataset of pop music ("`artist20`"). The former is used by the algorithm to determine learnability ("validation"). The latter is a "hold-out" set, used exclusively for performance evaluation.

`classic`: The `classic` dataset consists of 2000 song files divided in 14 categories. The bulk of the dataset are classical music: opera (4 categories: Händel, Mozart, Verdi and Wagner), orchestral music (6 categories: Beethoven, Haydn, Mahler, Mozart, Shostakovitch, Piano concertos) and chamber music (3 categories: piano, violin sonatas, and string quartets). The remaining category is comprised of a small set of pop music, intended as "dissimilar" music. Figure 1 shows a LDA projection plot of this dataset for visualization.

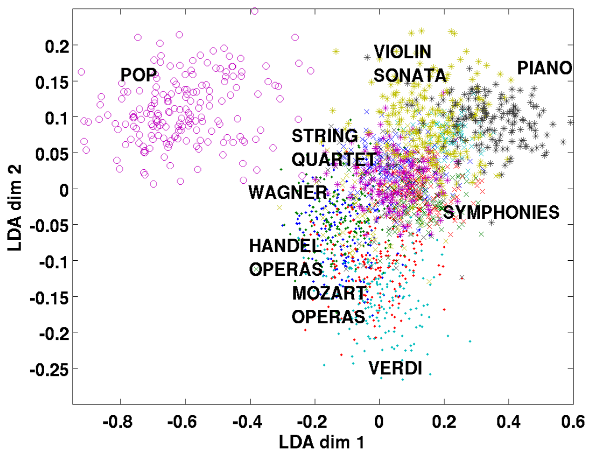

Fig. 1. Visualization plot of classical music dataset projected onto the first two LDA dimensions computed over the 14 music categories. General regions of music subclusters are identified.

`artist20`**:** We use a publicly available set `artist20` [7] to evaluate performance. `artist20` is a dataset of 20 pop artists (Table 1), each represented by six albums to yield a total of 1413 songs. Here each artist represents a category, the songs are the exemplars.

Aerosmith	Beatles	Creedence Clearwater Revival	Cure	Suzanne Vega
Tori Amos	Green Day	Dave Matthews Band	U2	Steely Dan
Garth Brooks	Madonna	Fleetwood Mac	Queen	Metallica
Radiohead	Roxette	Led Zeppelin	Prince	Depeche Mode

Table 1. Artists found in the `artist20` set. Each artist is covered by six albums.

Features: Data are represented by 20 mel frequency cepstral coefficients (MFCC) features [8], which are commonly used in speech/music classification. For each dataset, these features are computed from mono channel raw sources. For each song, we compute means and variances of the MFCC components averaged over time, to obtain a static 40-dimensional song-specific feature vector .

5 Results

For validation of our approach, we first examine the usefulness of the statistics $\mu_{usr,s}, \sigma_{usr,s}, \mu_{rnd,s}$, and $\sigma_{rnd,s}$ to decide on learnability. This test does not include the actual learning, which is later evaluated on the hold-out set `artist20` (sec 5.4).

For a given dataset, we simulate different user profiles. Each user likes a random subset of music categories, and dislikes the remaining categories. The user then labels a fraction q of songs in the dataset, drawn at random over all categories, into "like" or "dislike". Hereafter we will refer to this like/dislike judgment as "preference" to avoid confusion with the category "labels". Besides modeling user preferences, we also model their consistency: a given user has identical preference judgments only for a fraction p of songs of the same preference class. A p close to 0.5 thus indicates a high noise level, while a p of 1 indicates no noise ($p < 0.5$ would correspond to less noise with flipped labels). Since we are in particular interested in sparse feedback, we step q logarithmically from 0.5% and 10% for each user. Similarly, we simulate p from 50% (random) to 100% (no noise), in steps of 5%.

5.1 Easy, Noise-free User Preferences

Before using the 50 simulated users, we consider a simple example user for a first illustration ("`LikeVocal`"): `LikeVocal` has an easy preference scenario, liking vocal music (pop, opera; total 5 categories) and disliking non-vocal music (orchestral, chamber; total: 9 categories). From the LDA visualization in Figure 1, we see that the preference classes of `LikeVocal` are almost linearly separable, rendering this case particularly easy.

In the first test, we additionally assume a noise-less situation with `LikeVocal`'s preference being fully consistent ($p = 1$). For each value of q, we then generate 50 sets of randomly selected $Y_{L,usr}$, and run the algorithm for each set.

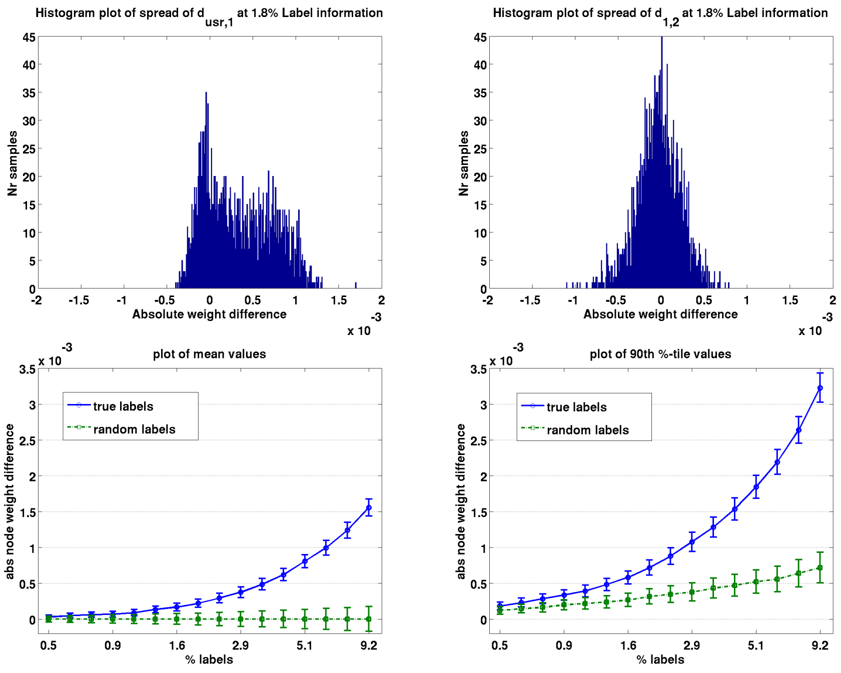

Fig. 2. For user `LikeVocal`: *Top:* Distribution of values for $d_{usr,1}$ (left) and $d_{1,2}$ (right). We see that the distribution of $d_{usr,1}$ is not normally distributed, hence computing statistics on the median or other percentiles is more robust than mean and variance. *Bottom:* Absolute weight value differences w.r.t. random labeling with conservation of label distribution. Left: Mean value, Right: 90-th percentile. Note that the x-axis is presented on the log-scale.

First we inspect the distributions of the $d_{\cdot,\cdot}$ (Figure 2, top). Whilst the random case (e.g., $d_{1,2}$) is likely to be Gaussian (Figure 2, upper right), this does not hold for the user set (e.g., $d_{usr,1}$; Figure 2, upper left). We observe that the accumulated weights are substantially higher for the true labels than the weights for the randomly seeded graphs. This is clearly reflected in Figure 2, bottom, where the 90-th percentile reflects better discriminatory strength. This observation motivates the use of the 90-th percentile statistic for the remainder of the paper.

5.2 Noisy User Preferences

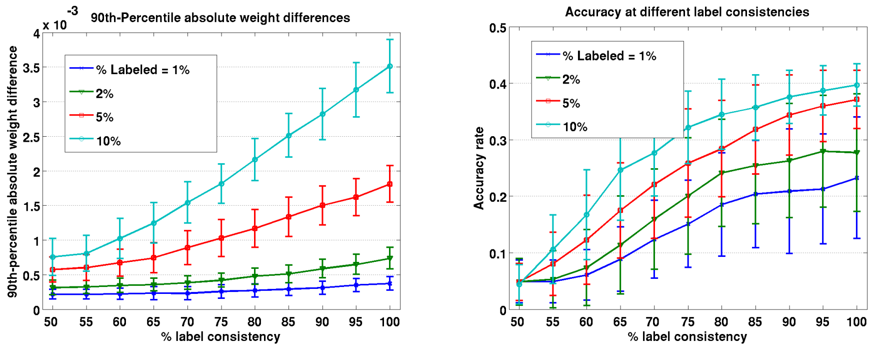

Fig. 3. Left: Distribution of 90-th percentile absolute weights for different levels of noise on `classicaleasy` Right: Corresponding error rates of LLGC.

So far we have considered a noise-free case ($p = 1$). While preserving the general preference, our user `LikeVocal` becomes inconsistent ($p < 1$) . The absolute weight difference d increases with decreasing noise (Figure 3). This trend is mirrored in the classification performance on the entire data X (including the unlabeled subset $X = X_U \cup X_L)^2$.

5.3 Random User Preferences

We now consider a harder problem, where the sub-clusters are randomly assigned to the two classes of user preferences. Here, the overlap zones are higher, and some assignments may be less intuitive, e.g. the user likes Mozart operas but dislikes Verdi operas. In `classicalhard`,we randomly model 50 users, with randomly sampled preference assignments. The results of our algorithm are shown in Figure 4. Compared to the easy user `LikeVocal`, the error rates are considerably higher. This effect is expected, as the increased class overlap renders the classification problem more difficult. Nevertheless, we observe the same pattern as in the easy case. Again, weight differences increase with p and q. This trend shows that the weight differences are a good measure for learniblity and thus a useful basis for our algorithm.

5.4 Evaluation on hold-out set

So far we have demonstrated that our statistical measures can provide a useful threshold for learning, but it remains to be evaluated in the full setting. For each

[2] The error rate is corrected for symmetry, i.e.consistent mislabeling would be judged as correct: the error rate is maximally 50%

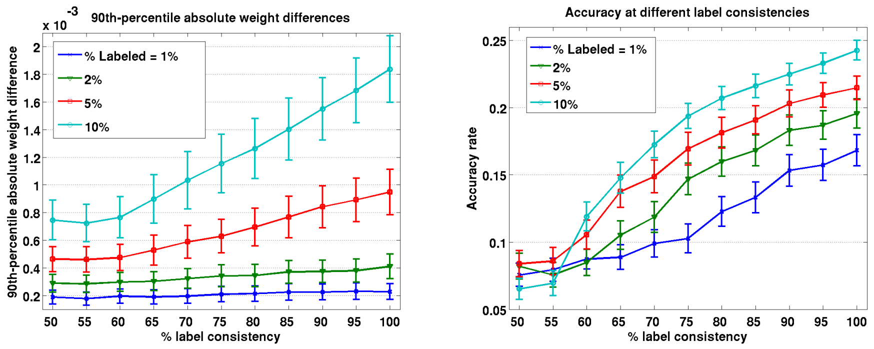

Fig. 4. Left: 90-th percentile of absolute weights differences for different levels of noise on set `classicalhard`. Right: Corresponding error rates of LLGC. Note that error bars show standard error of mean (SEM).

simulated user, our algorithm is presented with different proportion of labels (q) at different consistency levels (p). The task is to decide whether or not to reject the labels. On the sets that are accepted, the error rates are evaluated. We here use a dataset held-out when designing the algorithm (`artist20`). We again simulate 50 users, each by splitting the 20 artists (categories) randomly into 2 classes (like/dislike). For each user, we vary the proportion of user feedback (the number of songs labeled, q), and consistency of the preference (p). For all values of p and q average categorization accuracy on all songs (labeled+unlabeled) is improved when our algorithm is (Figure 5) compared to a continuously learning baseline system. The improvement gained through our algorithm is largest, when the labels are very inconsistent (small p) or only a small fraction is labeled (small q). In these cases rejection rate is high, and classification can benefit most. In turn, if the user provides a large and consistent set of preference judgments (large p and q), rejection is mild and performance without rejection is similar. This result shows that our algorithm reliably rejects uninformative data, while preserving consistent and informative feedback nearly in full.

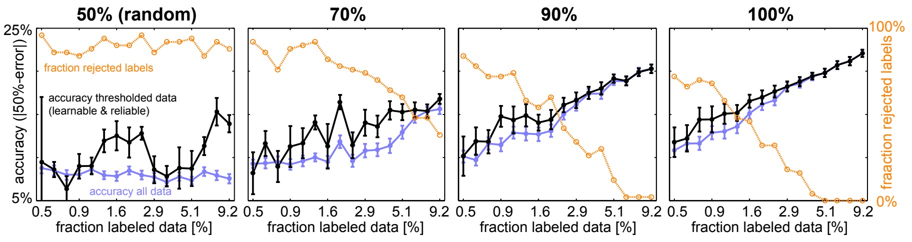

Fig. 5. Estimated accuracy (left axis) with decision to learn and rejection rates (right axis). Panels left to right show values $p = 50\%, 70\%, 90\%$ and 100%. and noise levels. Error bars report SEM.

6 Conclusions

We present an algorithm that decides whether or not user feedback in a real-world application is consistent and thus learnable given the input representation. By using real-world data and realistic labeling, the learning threshold induced by our algorithm increases performance for the envisioned hearing instrument application. Hence we demonstrate the feasibility of a statistically motivated selection stage as preprocessing step that can be integrated into online learning on noisy data with sparse labels.

Although the present study focuses on a specific application in the auditory domain, we are confident that the presented algorithm is applicable to more general settings. Whenever labeling can be corrupted by noise or incompatible with the features used, our criterion prevents a system from learning from malicious data. In fact, any online learning system with limited memory and unreliable labeling can benefit from our algorithm. Beyond the realm of hearing instruments, this might be of importance for any portable device that needs to adapt to a changing environment, such that factory-training is not sufficient. In the dawning age of ”wearable” computing, such devices are likely to become more and more ubiquitous. While the relevance of our algorithm for such applications still needs to be established in further research, our results present a crucial step towards ”smart” hearing instruments that adapt to changes in environment and in idiosyncratic preferences.

Acknowledgement: This work is funded by KTI, Nr8539.2;2EPSS-ES.

References

1. Büchler, M., Allegro, S., Launer, S., Dillier, N.: Sound classification in hearing aids inspired by auditory scene anlysis. Journal of Applied Signal Processing **18** (2005) 2991–3002
2. Moh, Y., Orbanz, P., Buhmann, J.: Music preference learning with partial information. In: ICASSP. (2008)
3. Zolter, J.Z., Maloof, M.A.: Using additive expert ensembles to cope with concept drift. In: Proceedings of the 22nd Intl Conference on Machine Learning. (2005)
4. Chapelle, O., Schölkopf, B., Zien, A., eds.: Semi-Supervised Learning. MIT Press (2006)
5. Belkin, M., Niyogi, P.: Using manifold structure for partially labelled classification. In: Advances in NIPS. Volume 15. (2003)
6. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Advances in Neural Infomation Processing Systems. Volume 16., MIT Press (2004) 321–328
7. Ellis, D.: Classifying music audio with timbral and chroma features. In: Proc. Int. Conf. on Music Information Retrieval, Vienna, Austria (Sep. 2007)
8. Ellis, D.: PLP and RASTA (and MFCC, and inversion) in Matlab (2005) online web resource.