Active-Transductive Learning with Label-Adapted Kernels

Dan Kushnir*
Alcatel-Lucent Bell Laboratories
Murray Hill, NJ 07974
dan.kushnir@alcatel-lucent.com

ABSTRACT

This paper presents an efficient active-transductive approach for classification. A common approach of active learning algorithms is to focus on querying points near the class boundary in order to refine it. However, for certain data distributions, this approach has been shown to lead to uninformative samples. More recent approaches consider combining data exploration with traditional refinement techniques. These techniques typically require tuning sampling of unexplored regions with refinement of detected class boundaries. They also involve significant computational costs for the exploration of informative query candidates. We present a novel iterative active learning algorithm designed to overcome these shortcomings by using a linear running-time active-transductive learning approach that *naturally* switches from exploration to refinement. The passive classifier employed in our algorithm builds a random-walk on the data graph based on a modified graph geometry that combines the data distribution with current label hypothesis; while the query component uses the uncertainty of the evolving hypothesis. Our supporting theory draws the link between the spectral properties of our iteration matrix and a solution to the minimal-cut problem for a fused hypothesis-data graph. Experiments demonstrate computational complexity that is orders of magnitude lower than state-of-the-art, and competitive results on benchmark data and real churn prediction data.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*

Keywords

classification; transductive learning; graph; active learning; uncertainty sampling

*Dan Kushnir is a member of technical staff at Bell Laboratories research arm

KDD'14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623673.

1. INTRODUCTION

Active learning is concerned with the design of algorithms that efficiently select a set of labeled training samples. The goal is to query those labels from a teacher which potentially optimize the prediction accuracy for the remaining unlabeled set. An active learner chooses at each trial a point (or possibly a batch) to be labeled by the teacher, and then learns a classification model. This process repeats itself until a label budget is used up. It has been shown that in some cases actively querying training samples expedites the learning process over traditional semi-supervised learning (e.g. [3, 15]), and is valuable especially when labels are scarce, expensive to obtain, or when unlabeled samples are abundant.

Past work on active learning can be roughly divided into two main approaches: First, an approach that focuses on refinement of the version space by actively probing the class decision boundary (e.g. [28]). This approach, typically an inductive one, often fails as it misses class boundaries that were not sampled, simply by focusing on refinement of boundaries already discovered and avoiding 'outlier' portions of the data. A second approach is a combined one, which employs both refinement of the version space and general exploration of the feature space (e.g. [3, 4, 6, 23]). These methods typically use graph-based techniques, where the exploration step samples pockets of data that the learner misclassifies and refines their class boundaries. However, the detection of such pockets is computationally challenging.

In this paper we focus on transductive learning and in particular on the second approach which combines exploration with boundary refinement since it captures both common types of classification failures. Our contribution is an active-transductive approach that yields competitive results while its implementation and computational effort are significantly simpler and lower than state-of-the-art algorithms. These advantages render it useful for real large data sets. In addition, unlike the combined approach that alternates between boundary refinement and exploration by use of pre-calculated parameters (e.g. [3, 23]), our method demonstrates early preference for exploration and naturally switches to refinement without the need for such parameters.

Our query strategy adapts the 'uncertainty sampling' criterion [20] to the graph setting, and shows that it can be used to *both* probe unexplored data pockets and refine decision boundaries in linear running time. It builds upon a probabilistic framework that combines the density and geometry of data with the current label hypothesis that emerges in the active learning process. Our framework uses

label-adapted kernels which also improve the accuracy of the querying process when class distribution is *not smooth* over the data. This advantage is emphasized since many other (passive and) active-transductive approaches assume such limiting smoothness for their success (e.g. [8, 15, 32, 35]). In this paper we introduce theoretical justification for the use of label-adapted kernels as well as a convergence analysis, demonstrating its advantages in active-transductive learning. Our supporting theory relies on a careful spectral analysis of the iterative technique of Jacobi [13] for a particular system of equations and its connection with spectral graph theory [10].

Aside from the observed improved accuracy, the presented classifier and the query components both have asymptotic running time that is linear in the size of the data set. In particular, and unlike other active classifiers (e.g. [3,23,35]), our algorithm does not require a separate run of several classifiers for each query, and does not employ computationally demanding clustering algorithms, such as [4, 22]. It exhibits orders of magnitude speedup in empirical experiments, confirming its asymptotic complexity, while achieving superior accuracy.

The structure of the paper is as follows: in section 2 we review related work, and in section 3 we briefly introduce the problem of active learning. In section 4 we describe our approach and give a detailed description of our algorithm components and their running time. Section 5 consists of running time and accuracy experiments with benchmark and real data. We conclude in section 6.

2. RELATED WORK

The literature on semi-supervised learning is extensive (see [8] for an overview). Semi-supervised learning algorithms use unlabeled data during training to improve learning performance. In the transductive setting the unlabeled set is transduced with label information from other points. Graph-based algorithms have been developed to accurately transduce labels from a few training points to the unlabeled set of points by relying on graph structure (e.g. [5, 8, 11, 19, 32–34]). A basic assumption behind these methods is that class structure is smooth over the graph, or in other words, that classes separate well with respect to the data density patterns. This assumption allows to construct Markov processes whose transition probabilities rely on proximity in the data feature space (e.g. [8, 11, 27, 34]) in order to propagate label information. Alternatively, others use the smooth eigenvectors of the graph Laplacian (similar to the eigenvectors of the Markov matrix) as a basis to extend the labeling function from training samples to the unlabeled set (e.g. [26], [5]).

Much of the research in active learning focuses on the tradeoff between refinement and exploration (e.g. [3, 6, 23]), which is central in motivating our approach. As an example for algorithms that conduct merely refinement, we consider 'SIMPLE' [28]. 'SIMPLE' queries points closest to the center of the largest hypersphere that fits in the version space. Yet, 'SIMPLE' assumes a fairly symmetric version space and centrality of the hypersphere, which in many cases is not true. The geometry of the version space may be complex and requires exploration first, as naturally incorporated in our approach. Moreover, 'SIMPLE' has a high running time, which scales between $O(N^2) - O(N^3)$, as the number of support vectors is multiplied by the time to compute the kernel for each query. Other algorithms such as 'COMB' [3] incorporate exploration to overcome the shortcomings of 'refiner'-type algorithms. 'COMB' employs three active learners: the inductive 'SIMPLE' [28], a risk minimization active learner [24], and KFF (Kernel Farthest First) [3] which is an 'explorer'-type active learner. At each stage of 'COMB' a learner is chosen based on the reward utility assigned by each learner. While this approach, in fact, introduces a tradeoff between three different querying approaches, its implementation and choice of algorithm selection parameter are complex. Its computational complexity is also far from being negligible, as it uses 'SIMPLE'. A second example is the exploration method suggested in [23]. This algorithm combines 'SIMPLE' with KFF. The authors use an adaptable probabilistic parameter to choose the learner. However, the implementation is still demanding in terms of running time. As a third example we consider the algorithm +EXPLORE of [4]. The authors use spectral clustering [21] to expose uncovered clusters that the learner missed while querying samples. Otherwise they use uncertainty sampling [20]. +EXPLORE is augmented to graph-based active-transductive algorithms such as [5, 19, 32, 35] and introduces a remarkable improvement to these algorithms. Yet, +EXPLORE involves a significant computational effort in order to discover data clusters at each learning step. +EXPLORE is very close to our approach as its graph construction also relies on the currently available hypothesis to change the graph similarities. Even so, it also deviates from our approach in two fundamental ways: first, it uses the computationally consuming spectral clustering [21] (which has to be used after each query), whereas we use a iterative linear time diffusion kernel approach. And second, +EXPLORE uses a 'hard' decision criterion to connect same label nodes by an edge, while we use a 'soft' local one.

The algorithms [34] and [32] are two iterative graph transductive learning algorithms that inspired active learning extensions such as [34], [29], and [17]. These approaches rely on the limiting smoothness assumption, whereas we introduce a new label-adapted iteration to overcome this limitation. We note [26] and [11] that use variations of function-adapted kernels in the context of passive learning. However, these ideas were not used before or extended into the active learning setting.

In a recent work by [31], which focuses on active learning for unsupervised spectral clustering, the authors use the eigenvalue decomposition of the data graph Laplacian to probe query points that preserve matrix structure. Our supporting theory also relies on the spectral properties of the data graph. Yet, for our carefully designed iteration matrix the query component uses a linear combination of the dominant first and second eigenfunctions which are shown to minimize a combined cost function that captures *simultaneously* the labeling-function graph-cut and the data graph-cut. Finally, we consider recent work on batch-mode learning (e.g. [9, 18, 30]). Batch-mode active learning reduces the number of classifier calls per query, yet, it involves optimization procedures that are computationally demanding (polynomial or exponential in number of points). The tradeoff in running time should be carefully investigated with big data sets. For small data sets such as UCI's SEGMENTATION (as reported in [9]) the accuracy of our linear running time algorithm is favorable (see Sec. 5). Also, [9] reports running time results on data sets of a few hundred points and

the reported ones (e.g. WAVEFORM) have higher running time than those reported for our algorithm in single-query mode. [18,30] do not report running times. Our experiments on big data sets include a simplistic uncertainty-based batch selection approach, and we mark more advanced batch-mode learning as future work.

3. PROBLEM SETTING

We address the problem of actively learning a classifier. Without loss of generality, consider the binary case with a set $S_{l+u} = \{x_i, y_i\}_{i=1}^{l+u} \subseteq \mathbb{R}^D \times \{+1, -1\}$ of $l+u$ points in $\mathbb{R}^D$ and their binary labels $h \in \{+1, -1\}$. In transductive learning the learner receives $X_{l+u} = \{x_i\}_{i=1}^{l+u}$ and a random training set $S_l = \{x_i, y_i\}_{i=1}^{l}$. The learner generates a soft prediction $\hat{\chi} = (\hat{\chi}_1, .., \hat{\chi}_{l+u})$, where $\hat{\chi}_i \in [+1, -1]$, and outputs $\hat{h} = \texttt{sign}(\hat{\chi}_i)$. In our active-transductive setting the l training samples are *actively* chosen from X_u by the learner with the goal to minimize some loss-function on the remaining unlabeled points in X_u. We also consider in our experiments an active learning problem in which $X_u = \{X_{u1}, X_{u2}\}$ such that the learner is allowed to query points only from X_{u1}, and predicts for X_{u2}.

4. THE ALGORITHM AND ITS IMPLEMENTATION

We describe below our approach, supporting theory, and the detailed implementation of our algorithm, including its extension to the multi-class problem and its running-time complexity analysis.

4.1 Notation and Preliminary Setting.

We start by defining a finite weighted graph $G = (V, E)$ consisting of a set of *vertices* V, a set of *edges* $E \subset V \times V$, and a nonnegative *weighting* function $W : E \to \mathbb{R}^+$. We interpret the weight $W(v, \tilde{v})$ as a measure of similarity between the vertices v and $\tilde{v}$, when $(v, \tilde{v}) \in E$. The graph kernel is defined by

$$K = D^{-1}W, \tag{1}$$

where D is diagonal with $D_{ii} = d_i = \sum_j W(v_i, v_j)$. Thus Kf is a weighted averaging operator of some arbitrary function f to be studied: $f_i = (d_i)^{-1} \sum_{i\ j} w_{ij} f_j$, with the weights measured by the similarities W_{ij}; it has the 'averaging effect' of *smoothing* the function f over the graph.

In the data context, a graph G can be constructed in which the vertices of G correspond to the data points in X. W represents similarity between data points:

$$W(x_i, x_j) = m_1\left(\frac{\rho(x_i, x_j)^2}{\sigma_1(i,j)}\right), \tag{2}$$

where $\sigma_1(i, j)$ is a local scaling parameter, and typically $m_1(a) = exp(-a)$ with ρ as the Euclidean distance. The actual similarities used are the local ones in order to preserve local geometry and reduce computation time by using sparse matrices. These local similarities are realized by computing the k-nearest neighbors for each point x, denoted by $N_k(x)$. Throughout our experiments we select $\sigma_1(i, j)$ to be the median distance among the k nearest neighbors of x_i and x_j:

$$\sigma_1(i,j) = \text{median}\{\rho(x_i, x_l)_{l \in N_k(x_i)}, \rho(x_j, x_q)_{q \in N_k(x_j)}\}. \tag{3}$$

To facilitate notation $\sigma_1(i, j)$ will be typically denoted by σ_1. $K(x_i, x_j)$ follows immediately from (1):

$$K(x_i, x_j) = d_i^{-1} W(x_i, x_j). \tag{4}$$

4.2 Classification via Diffusion Kernels.

At the center of our algorithmic ideas lies a Markov process that is used to propagate labels from S_l to X_u. We start with the row-stochastic kernel defined by (1), which is viewed as the transition probabilities of a Markov random walk on the data neighborhood graph. The weights for each row i correspond exactly to the transition probabilities of a random walk starting at x_i. Specifically, the one step transition probability between states x_i and x_k is given by

$$p_{ik} = K(x_i, x_k) = \frac{W_{ik}}{\sum_j W_{ij}}, \tag{5}$$

where we denote $W(x_i, x_j)$ by W_{ij} to facilitate notation. We consider a random walk as means to assign a label to $x_i \in X_u$. The predicted label of x_i is associated with the probability of arriving to a labeled point of class 1 after performing a random walk starting at x_i [34] (we consider the binary case w.l.g.). Marking this probability as $p(y_{end} = 1|i)$, we consider the relation

$$p(y_{end} = 1|i) = \sum_j p(y_{end} = 1|j) p_{ij}, \tag{6}$$

and associate $p(y_{end} = 1|i)$ with the probability $p(y_i = 1|x_i)$. For labeled points $p(y_{end} = 1|i) = 1$, or 0 otherwise. Denoting $2p(y_{end} = 1|i) - 1$ by χ we see that $\chi \in [-1, 1]$ and its sign can be used to generate $\hat{h}$. χ can be partitioned as $\chi = [\chi_l, \chi_u]$. Similarly, D and W are partitioned into blocks

$$D = \begin{pmatrix} D_{ll} & 0 \\ 0 & D_{uu} \end{pmatrix} \text{ and } W = \begin{pmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{pmatrix}.$$

Eq. (6) can be transformed and re-written for the unlabeled points X_u and χ in matrix form as

$$\chi_u = (D_{uu}^{-1} W_{ul} \;\; D_{uu}^{-1} W_{uu}) \begin{pmatrix} \chi_l \\ \chi_u \end{pmatrix}, \tag{7}$$

resulting in the system

$$L_{uu}\chi_u = W_{ul}\chi_l \tag{8}$$

for the unlabeled samples, where $L = D - W$ is the graph Laplacian, and the sign of each χ_i gives the label of x_i. A similar system, motivated by quadratic energy minimization, is obtained by minimizing

$$C(\chi) = \frac{1}{2} \sum_{i,j=1}^{l+u} W_{ij}(\chi_i - \chi_j)^2 = \chi^T L \chi \tag{9}$$

while forcing equality on the labeled set $\chi_l = Y_l$ [8]. Specifically, minimizing (9) with respect to χ_u leads to

$$L_{ul}Y_l + L_{uu}\chi_u = 0 \Leftrightarrow L_{uu}\chi_u = -L_{ul}Y_l, \tag{10}$$

which is the same as (8), since $L_{ul} = -W_{ul}$.

The system (10) can be solved via the well-known Jacobi method [8, 25]. The iterative Jacobi method solves the system $Mx = b$ by approximating the solution at the step $(t+1)$ by

$$x_i^{(t+1)} = \frac{1}{M_{ii}}\left(b_i - \sum_{j \neq i} M_{ij} x_j^{(t)}\right). \tag{11}$$

The Jacobi iteration matrix is defined to be $B_J = D^{-1}(R+L)$, where D is the diagonal matrix with M_{ii} on its i-th diagonal element, and R and L are the upper and lower triangular matrices of M. Hence, in matrix notation the iterative scheme is

$$x^{(t+1)} = D^{-1}(b - (R+L)x^{(t)}) \Leftrightarrow x^{(t+1)} = B_J x^{(t)} + D^{-1}b. \quad (12)$$

For faster convergence one may consider the damped iteration involving a parameter $\omega \in [0,1]$, where the iteration matrix is defined as $B_\omega = (1-\omega)I + \omega B_J$, so that

$$x^{(t+1)} = B_\omega x^{(t)} + \omega D^{-1}b. \quad (13)$$

For the system (10) we have $M = L_{uu}$, $x = \chi_u$, and $b = L_{ul}Y_l$, which then yields the iteration

$$\chi_i^{(t+1)} = \frac{1}{L_{uu}(x_i,x_i)} \left(-(L_{ul}Y_l)_i - \sum_{j \neq i} L_{uu}(x_i,x_j)\chi_j^{(t)} \right) \quad (14)$$

It is clear that (14) is a label diffusion process: transducing a label to x_i as a weighted average of the labels of its neighbors. In a slightly different view, it is equivalent to the iterative application of the kernel (1) on the characteristic vector χ, while restarting χ_l to Y_l after each iteration. Its probabilistic justification is in the random walk framework (6), which leads to classification. The summary of the label diffusion algorithm *DiffuseLabels* is given in Fig. 1.

We note that this formulation is related to various label propagation algorithms suggested in [11, 32, 33] except for the normalization steps there. Also, when $t \to \infty$ *DiffuseLabels* is related to the harmonic classifier of [34]. Yet, the equivalence we have just drawn between our label propagation and the Jacobi iteration plays an important role in deriving convergence and solution properties for the following iterative process that we propose for classification and active learning. We prove bellow that our label diffusion process converges, and provide a *multi-class extension* to our approach.

COROLLARY 1. ***The iteration in DiffuseLabels (14)***

$$\chi_i^{(t+1)} = \frac{1}{L_{uu}(x_i,x_i)} \left(-(L_{ul}Y_l)_i - \sum_{j \neq i} L_{uu}(x_i,x_j)\chi_j^{(t)} \right)$$

converges as $t \to \infty$.

PROOF. *The proof relies on the convergence proof of the Jacobi method [25]: if M (11) is strictly diagonally dominant then the iteration converges. Since for L_{uu} we have $L_{uu}(x_i,x_i) = \sum_{j=1}^{l+u} L(x_i,x_j)$, we have that $\forall i$ $L_{uu}(x_i,x_i) > \sum_j L_{uu}(x_i,x_j)$, therefore $M = L_{uu}$ is strictly diagonally dominant. This implies that the row sums $\{s_1,...,s_n\}$ of the iteration matrix B_J are smaller than 1. Since $\|L_{uu}\|_\infty = \max\{s_1,...,s_n\} < 1$, we obtain that the spectral radius of B_J is bounded by 1: $\max_i |\lambda_i| \leq \|L_{uu}\|_\infty < 1$, therefore B_J is a convergent matrix and the Jacobi iteration (14) converges.* □

The multi-class case. In the multi-class setting we consider $C = \{c_1,...,c_m\}$ classes, and $\Xi = \{\chi(c_1),...,\chi(c_m)\}$ characteristic assignment vectors instead of a single one as in the binary case. Each $\chi(c_j), 1 \leq j \leq m$, is a 'one vs. all' vector: $\chi_i(c_j) = 1$ if $x_i \in X_u$ belongs to c_j, or $\chi_i(c_j) = -1$, if $x_i \in X_u$ belongs to c_k, $k \neq j$, otherwise $\chi_i(c_j) = 0$. The label diffusion is applied to each vector and the classification is computed for a data point x_i as $\arg\max_j\{\bar{\chi}_i(c_j)\}$.

DiffuseLabels(X_{l+u},S_l, t, K, χ)
Input: X_{l+u}: data, S_l: training set, K: kernel,
χ: current labeling approx. t: number of iterations.
Output: $\bar{\chi}$
if K and χ are empty:
  $K = ConGeometKernel(X_{l+u})$
  Initialize χ with labeling of S_l and 0 elsewhere.
for $i = 1$ to t, do:
  $\bar{\chi} = K\chi$; $\bar{\chi}_l = Y_l$; $\chi = \bar{\chi}$;
end for
Return $\bar{\chi}$.

Figure 1: The algorithm *DiffuseLabels*.

4.3 Classification via Label-Adapted Kernels.

The diffusion of labels (*DiffuseLabels*) depends only on information from data feature space encoded into the weights W_{ij}. In particular, the minimization of the quadratic energy (9) relies on the assumption that data points that belong to different classes will have a low similarity weight. It is this assumption on the smoothness and separability of the class label function which guarantees that the diffusion process will capture the class structure. This smoothness assumption, which is used in semi-supervised classification tasks, is often both local and global, i.e. h does not change much between nearest neighbors and on clusters of data points. However, in many real data sets this might not be the situation, as h may not be smooth with respect to W. To this end, we propose to modify the geometry captured by W so that the geometry of $\bar{\chi}$ - a smoothed version of χ that captures the current labeling hypothesis, is taken into account. To achieve this goal we consider the following label-adapted weight matrix in a second diffusion process:

Definition 1. **The label-adapted weight matrix is defined as**

$$W^{\bar{\chi}}(x_i,x_j) = m_1\left(\frac{\rho_1(x_i,x_j)^2}{\sigma_1}\right) m_2\left(\frac{\rho_2(\bar{\chi}(x_i),\bar{\chi}(x_j))^2}{\sigma_2}\right), \quad (15)$$

and its associated kernel is $K^{\bar{\chi}}$. m_1 and m_2 are decaying exponentials, ρ_1 and ρ_2 are distance metrics in $\mathbb{R}^D$ and $\mathbb{R}^k$ (typically $k=1$), σ_1 is a data scaling parameter that is chosen as in (3), σ_2 is a scaling parameter in the label feature space (its selection is discussed below), and $\bar{\chi}$ is a smoothed soft label estimate for h obtained via *DiffuseLabels* with the kernel K.

The construction in (15) assigns stronger affinity between points that are believed to belong to the same class, and weaker affinities between points of different classes. In particular, when $\sigma_2 << \sigma_1$, the associated averaging kernel K will average locally, but much more along the (estimated) contours of h than across them. Thus the selection of σ_2 depends on the importance of the estimation of h by $\bar{\chi}$. A median approach (3) can be used, which captures the level of uncertainty in $\bar{\chi}$. Or $\sigma_2 = 1$, which captures the average distance between the two labels {-1,1}.

We therefore consider a second label diffusion process involving the kernel (15), which follows a diffusion process with the kernel (4). This second diffusion process yields a

new labeling estimation denoted by $\hat{\chi}$, as described in the pseudo-code *LAClassifier* in Fig. 2. The label diffusion process involving $K^{\bar{\chi}}$ will manifest fast label diffusion along the (suspected) same-class samples and slow diffusion along class boundaries. This property is advantageous to accelerate and improve classification, in particular when classes do not separate over the data, or in other words, the smoothness assumption does not hold. In this case our resulting iterant $\hat{\chi}$ will provide an approximation to h and not only to the data density distribution. In order to establish these claims, we first introduce Theorem 1 which shows that the Jacobi iteration with the label-adapted kernel (15) converges to an approximation of the class decision boundary *independently* of the smoothness assumption. Second, we show in Theorem 2 that in the modified graph geometry the energy function (9) is further minimized by $\hat{\chi}$.

THEOREM 1. ***Let $\hat{x}$ be the resulting iterant from the Jacobi iteration with the kernel (15) in LAClassifier. Then $\hat{x}$ approximates a minimal graph-cut in a graph constructed from the unknown true binary label function h and the data. In particular, $\hat{x}$ approximates h.***

PROOF. See appendix. □

THEOREM 2. ***Let $\bar{\chi}$ and $\hat{\chi}$ be the corresponding minima of the quadratic energy function (9) with W and $W^{\bar{\chi}}$. Then the label-adapted weight matrix and $\hat{\chi}$ further minimize (9). In particular,***

$$\sum_{i\in u, j\in l+u} W_{ij}^{\bar{\chi}}(\hat{\chi}_i - \hat{\chi}_j)^2 \leq \sum_{i\in u, j\in l+u} W_{ij}(\bar{\chi}_i - \bar{\chi}_j)^2$$

PROOF. See appendix. □

Theorem 1 and its proof reveal the structure of $\hat{\chi}$: it is dominated by the first eigenvectors of the kernel $W^{\bar{\chi}}$. The combination of the 1st and 2nd eigenvectors provides a real smooth approximation of a graph-cut of the points of the binary set h. As such it is simply an approximation of h. Most important, we show below how $\hat{\chi}$ is used within active learning: its transition from negative to positive values indicates the class decision boundary, and precise zero values indicate unexplored data portions. Another strength of Theorem 1 is in its independence from the smoothness assumption and class separation in data space: Popular data-constructed kernels (4) as, for example, in [5,11,32–34] result in a good estimate for h *only* if the smoothness assumption holds.

Theorem 2 establishes that the new solution $\hat{\chi}$ is smoother in the modified geometry as it minimizes the classification error (9) with $W^{\bar{\chi}}$. Using $W^{\bar{\chi}}$ in the quadratic energy function (9) is more meaningful than using W, because it contains a penalty that does not depend only on the data but also depends on the labels approximation $\bar{\chi}$. Connecting this with Theorem 1 and its proof, the resulting iterant $\bar{\chi}$ that minimizes the classification error contains the smooth components which better approximate h. As a result, the query component which relies on the absolute values of $\bar{\chi}$ attains better accuracy, as shown in the following section.

To this end the label-adapted diffusion suggests two key advantages: First, it computes the soft label approximation $\hat{\chi}$ in linear running time, while other methods, relying on black-box clustering algorithms, run with higher complexity (e.g. [4, 22, 23]) (see Sec. 4.5). Second, it suggests a classifier that is highly accurate and can be applied to classification problems where there is no class separation over the data density. Our supporting theory links $\hat{\chi}$ directly with h, which is key in constructing an efficient active learning algorithm, referred to as **Label Adapted Active Learning (LAAL)**.

LAClassifier$(X_{l+u}, S_l,\ t)$
Input: X_{l+u}: all data measurements,
S_l: the training set, t: number of iterations.
Output: $\hat{\chi}$
$K = ConGeometKernel(X_{l+u})$
Initialize χ with labeling of S_l and 0 elsewhere.
$\bar{\chi} = DiffuseLabels(t, K, \chi)$
$K^{\bar{\chi}} = ConstructLAKernel(W, \bar{\chi})$ (Eq. (15))
$\hat{\chi} = DiffuseLabels(t, K^{\bar{\chi}}, \chi)$
Return $\hat{\chi}$

Figure 2: Label propagation with label-adapted kernels *LAClassifier*.

4.4 Active Learning with Label-Adapted Kernels.

Theorem 1 and its proof establish that the output of *LAClassifier* - $\hat{\chi}$, is an approximation to the true binary labeling function h. As such, low $\hat{\chi}$ absolute values indicate either a class decision boundary or a yet unexplored region where a decision boundary could exist. With this observation we choose to adapt an uncertainty sampling criterion to perform active learning in our transductive setting. In general, representative points whose approximated soft labels $\{\hat{\chi}_i\}_{i=l+1}^{(l+u)}$ have a low absolute value: $q = \arg\min_i(|\hat{\chi}_i|)$ are queried. Such points indicate that the labeling function has not been sampled sufficiently or that a decision boundary exists. We distinguish between the two cases:

1. **Exploration**: Our fundamental observation is that after the iterative label-adapted diffusion process completes, regions of X_u that have not been explored will be identified as X_u points with label weights that are *essentially zero*: $\hat{\chi}_i = 0$. Zero-label points indicate clusters or even disconnected graph components that are unreachable by the labeling propagation process. Representative points of such data portions are potential candidates to reduce the hypotheses space, and should be explored. These representative points are identified by a high value of d_i which approximates the probability density function $p(x_i)$. This exploration is fast since the propagation algorithm has linear running time (see section 4.5), and the choice of the next query is based on merely sorting $\hat{\chi}$.

2. **Refinement:** Our second observation is that once exploration is mature, the label-adapted diffusion from S_l to X_u will essentially cover the whole graph. At that stage $\hat{\chi}$ will be non-zero everywhere. This situation naturally redirects LAAL's query selection process to query points along decision boundaries, since their soft labels are close to zero but numerically are *not* zero. The decision boundary points are captured by low $\hat{\chi}$ weights, as suggested by Theorem 1, which shows that $\hat{\chi}$ is a soft approximation of h.

Our query strategy is justified by the following probabilistic argument in Lemma 1:

LAAL(X_{l+u}, l)
Input: X_{l+u}, l: number of points to query,
Output: $h = \texttt{sign}(\hat{\chi})$
$S_0 = \emptyset$.
for $i = 1$ to l
  $\hat{\chi} = LAClassifier(X_{l+u}, S_{i-1})$.
  $(x_q, y_q) = Query(\hat{\chi}, X_u)$;
  $S_i = S_{i-1} \cup (x_q, y_q)$;
  $X_u = X_u \setminus x_q$;
end for
$\hat{\chi} = LAClassifier(X_{l+u}, S_l)$.
Return $\texttt{sign}(\hat{\chi})$

Figure 3: Pseudo-code for *LAAL*.

LEMMA 1. ***Choosing*** $q = \arg\min_i\{|\hat{\chi}_i|\}$ ***minimizes the expected error criterion*** $E[(\hat{y}_i - y_i)^2|x_i]p(x_i)$, ***where*** $p(x)$ ***is the density of*** X.

PROOF. *We expand the error expectation term:*

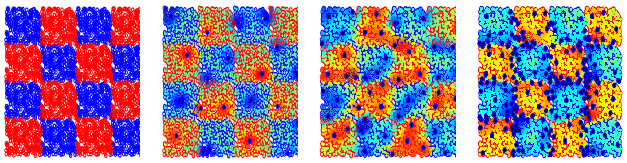
$$E[(\hat{y}_i - y_i)^2|x_i]$$
$$= p(y_i = 1|x_i)(\hat{y}_i - 1)^2 + p(y_i = -1|x_i)(\hat{y}_i + 1)^2$$
$$= (\hat{y}_i^2 + 1) - 2\hat{y}_i(p(y_i = 1|x_i) - p(y_i = -1|x_i)).$$

Following Theorem 1, the difference $p(y_i = 1|x_i) - p(y_i = -1|x_i)$ *can be well approximated by* $|\hat{\chi}_i| : E[(\hat{y}_i - y_i)^2|x_i] \approx (\hat{y}_i^2 + 1) - 2|\hat{\chi}_i|$. *Therefore, when* $\hat{\chi}_i$ *is minimal, and* $p(x_i)$ *(approximated by* d_i*) is maximal the expected error is maximal, which suggests querying such points.* □

REMARK 1. ***If*** $|\hat{\chi}_q| \neq 0$ ***we do not use the representative criterion*** $\arg\max_i\{d_i : i = \arg\min_i |\hat{\chi}_i|\}$ ***since class decision boundary may not correspond to data density, and therefore choosing according to density may cause contamination.***

Description of the algorithm. We are now ready to describe the label-adapted active learning algorithm (LAAL) in Fig. 3. LAAL is a meta-algorithm that employs the label-adapted diffusion classifier *LAClassifier* (Fig. 2) to produce the current labeling hypothesis, and a query selection component *Query*. LAAL iterates the two procedures and outputs the final hypothesis once the required number of labels l had been queried. The query selection algorithm - *Query* uses the current soft hypothesis to select the point x_i^q that satisfies the criterion of a minimal absolute value: $q = \arg\min_i\{|\hat{\chi}_i|\}$, in accordance with Lemma 1. If $\hat{\chi}_i = 0$ *Query* selects x_i^q to be a representative of an unexplored data portion or cluster by choosing the candidate with the maximal sum of weights - d_i. Alternatively, when all $\bar{\chi}_i$'s are nonzero (i.e. exploration is saturated), *Query* will enforce the minimum criterion only.

A toy example. Fig. 4 illustrates the active learning process of LAAL on a simple XOR-type binary classification problem (A), where class separation does not correspond to the data density. In this example, points are queried using LAAL, a hypothesis is generated for the rest of the points X_u, and accuracy is reported. In sub-figures B-D black circles correspond to queried points, and the data points circle-filling colors correspond to the soft hypothesis generated by LAAL. Sub-figure B demonstrates the exploration stage: data portions for which the soft hypothesis is essentially zero are queried. In C all X_u points found to be assigned with a non-zero soft label - indicative of a saturated exploration phase. As seen, a rough sketch of the correct classes is well captured by the current hypothesis. Finally, in sub-figure D the query process is naturally redirected to the refinement of class boundaries. The advantages of using the label-adapted process are further discussed in our experimental results in Fig. 5-right for the same XOR problem, where we show that active learning with label adaptation attains significantly better error rates (10%) than without label-adaptation.



A B: 71% C: 84% D: 98%

Figure 4: Demonstration of LAAL with accuracy rates. A: the XOR data set. B: Exploration stage (20 queries). C: Maturity of exploration: most of X_u are assigned with soft labels (50 queries). D: Refinement stage: additional 190 queries concentrate mostly on class boundary refinement.

4.5 Running time of the algorithm.

In the following we show that the running time of LAAL is linear in the size of the data set and its dimension. We underline two key observations: First, the diffusion of labels in $DiffuseLabels$ is an iteration of a product of a sparse matrix of size $N \times N$ ($N = l + u$) with a vector. Given that the kernel is $k-$sparse, the total complexity of $DiffuseLabels$ is $O(t \cdot N \cdot k)$. In practice, t is small and was found to give optimal results in our experiments when $t = 2, 3$. Careful analysis shows that optimally $t \approx \frac{\log(N/l)}{\log(k)} \approx \log(N)$. Secondly, the first query selection process ($Query$) performs $O(|X_l|k^t \cdot log(|X_l| \cdot k^t))$ operations to sort and find the lowest values of $\hat{\chi}$. This running time can be further reduced on subsequent iterations by using simple book keeping on the last iteration sorting results and a heap: there are at most k^t elements in $|X_u|$ that are influenced by the last query which need to be updated. Therefore the query running time is $O(k^t log(|X_l| \cdot k^t))$. In practice of LAAL's running time is mostly dominated by the graph construction procedure, which involves finding nearest neighbors for all data points in X_{l+u}. Naive exact nearest neighbor algorithms have running time that is $O(N^2 D)$, whereas space partitioning algorithms exhibit exponential complexity in high dimensions. We therefore resort to an approximate search (e.g. [1]) that suggests a tradeoff of an almost linear time complexity with a prescribed error constant ε.

For comparison, we discuss the running time of a few relevant active learners: Both [4] and [22] employ full clustering schemes for each query. '+ EXPLORE' [4] involves a spectral algorithm whose computational effort in general is cubic in N. As the authors report they had to half the size of standard benchmark UCI data sets in order to reduce a running time of over a month with 20 CPUs. [22] uses K-medoids clustering with complexity that scales as $O(i \cdot K \cdot (N-k)^2)$ with i as the number of iterations. Algorithms such as [35], also involve a computationally demanding procedure for es-

Table 1: Data sets used for experiments.

Data set / Info	size	D	type	m	source
XOR	6400	2	Real	2	[3, 23]
XOR-REDUCED	1000	2	Real	2	[3, 23]
XOR-HUGE	560000	2	Real	2	[3, 23]
MUSH	8124	139	Cat,Int	2	UCI
MUSK	476	166	Int	2	UCI
MAGIC	19020	11	Real	2	UCI
MAGIC-REDUCED	7762	10	Real	2	UCI
WAVEFORM	5000	21	Real	3	UCI
IONOSPHERE	351	34	Real,Int	2	UCI
CHURN	50000	171	Real,Int, Cat	2	[14]
COVERTYPE	581012	54	Real,Int	7	UCI
SUSY	1027000	18	Real	2	UCI
SEGMENTATION	2310	19	Real	7	UCI

timating the risk for each query candidate. In [23] and [3], support vector machines are used, and therefore for each query the running time is $O(|X_u| \cdot |V_h| \cdot T_k)$, where V_h is the number of support vectors and T_k is the time to compute the kernel. To this end LAAL's linear time complexity and observed accuracy suggest a significant improvement over current active algorithms, rendering it as practical for large data sets.

5. EXPERIMENTS

Active learning has already gained a significant body of algorithmic approaches. In our experiments we compare LAAL's performance with several carefully chosen representative baseline methods. First, the spectral-clustering based +EXPLORE of [4] with representative sampling which is closely related to our iterative approach mostly since it uses label adaptation via a 'hard' criterion, namely, by deleting graph edges of differently labeled points and establishing edges between similar label nodes. As shown in our accuracy experiments +EXPLORE indeed shows the closest performance to LAAL's; however, in our running time experiments it is significantly slower.

In addition, we compare LAAL with baseline uncertainty sampling approaches [20] where a variety of recent approaches have been proposed [12], [24]: We use a parametric version with a Bayes classifier [24], a non-parametric version referred to as 'Non-Adaptation' which amounts to using $\bar{\chi}$ for uncertainty sampling. We also compare with 'Large Margin' inductive approaches with Support Vector Machines [7], and with the ensemble approach of [23] and the algorithms of [3] and [28]. As control experiments we demonstrate LAAL's performance against a random query sampling approach and with and without label-adaptation. We use the 'XOR' data as a specific experiment to compare our performance on non-smooth data sets. We note that both [23] and [4] provide performance comparison with additional algorithms [3,5,19,28,32], therefore our comparison with [4] provides an even broader view of LAAL's performance. Other approaches focusing on batch learning [9,18,18,30], are fundamentally different and discussed in Sec. 2.

We start with an accuracy comparison that demonstrates LAAL's competitive accuracy, and then continue to a running time comparison on data sets of size up to 1 million which demonstrates LAAL's outstanding efficiency. The data sets used are summarized in table 1.

Parameters choice. $t = \log(N)$, σ_1 according to (3), and $\sigma_2 = 1$ (see 4.3).

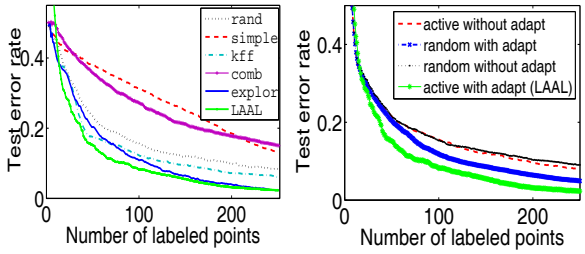


Figure 5: XOR-REDUCED data set. Left: results of [28] (SIMPLE), [3] (KFF and COMB), [23] (explor), random queries (random), and LAAL. Right: Comparison of LAAL with/without label-adapted kernels, and with random queries.

5.1 Accuracy experiments.

Non-smooth class function. We start with the binary 'XOR-REDUCED' classification problem which involves a synthetic data set from [3, 23], similar to Fig. 4. In 'XOR' the labeling function is not separable on the data, and therefore serves here as an example to compare LAAL's performance with various other methods some of which assume separability. The data is a generalization of the exclusive OR: a d-dimensional, $n \times n \times ... \times n$ "checkerboard" in which every square contains 250 points collected to a data set X ($\|X\| = 250n^d$). We repeated the following experiment 50 times: 100 points were randomly selected from X to serve as the test set X_{u2} and the remaining points were placed in X_{u1}. Classification error rates of LAAL as a function of the size of the training set X_l are shown in Fig. 5 [1] for $d = 2$, $n = 4$. Each curve is the average of 50 runs. The left sub-figure demonstrates a fast learning curve that LAAL exhibits over the other methods [3,23,28]. The result after 250 queried points is comparable to the best algorithm suggested in [23], with a faster convergence rate for LAAL. The right sub-figure depicts a performance comparison of LAAL with kernel methods that do not use the label-adapted kernel, but only the kernel that entails the geometric proximity of the data points, as in [32, 33]. This figure also provides a comparison in which active learning is disabled, so that the data points are selected randomly. In this case LAAL performs with significant accuracy improvement when compared with the other diffusion-based methods: the label-adapted kernel improves classification, and, most important, it enhances accuracy when our active query process is used.

The following accuracy experiments involved UCI data sets MUSK, MUSH, MAGIC-REDUCED, IONOSPHERE, and IMAGE SEGMENTATION [2]. We used our codes for LAAL, '+EXPLORE' [4], uncertainty sampling [20] in 3 differen versions: 1. with the parametric Bayes classifier [24] (coded 'US'), 2. with non-parametric *DiffuseLabels* (without label-adapted kernel - using only $\bar{\chi}$) (coded 'NAdapt'), and 3. inductive SVM-based large margin approach [7, 28] (coded 'ISVM'). As a control experiment we also used a random query selection (coded 'Rand') with LAAL. In these experiments X_l was minimal and the test set includes the rest of the data X_u. We run each algorithm to query 50 data points and checked the accuracy at each step. To fairly com-

[1] left figure contains results of [23], courtesy of Kun Deng

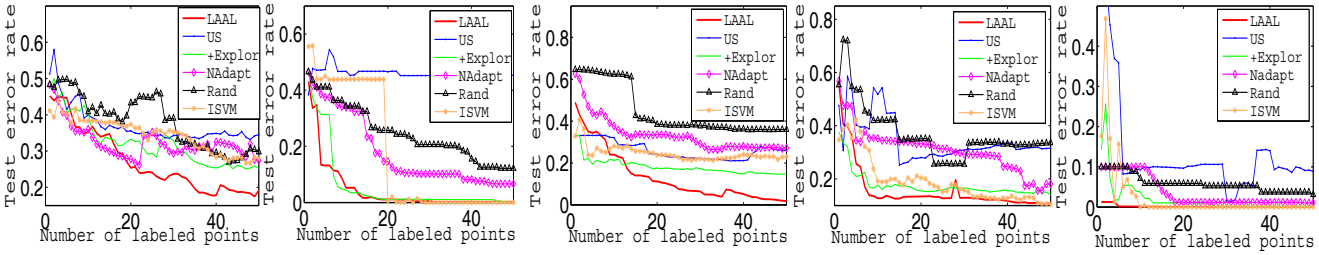


Figure 6: Error rate vs. number of labeled samples. Left to right: MUSK, MUSH, MAGIC, IONOSPHERE, and SEGMENTATION data sets.

pare LAAL and NAdapt with other methods we decided that points that have zero labels (typically because of being far away from labeled ones) will have a random label sampled which depends on the current proportion of positively classified labels. Results are reported in Fig. 6. LAAL demonstrates accuracy that is significantly better than the other algorithms over the full active learning period. '+EXPLORE' has a closer performance to LAAL and in early learning has short segments of better error performance; however, in all cases its overall behavior is less accurate than LAAL's, and we note again its higher running time. The Bayes classifier with uncertainty sampling shows to be less accurate, and diffusion without label-adapted kernels (using only $\bar{\chi}$) performs worse than LAAL, as it relies on the smoothness assumption. The inductive ISVM sometimes shows high accuracy at latent stage of learning - when enough points have been labeled. Lastly, we demonstrate the application of LAAL on a noisy large churn prediction tournament [14] data set, obtained from the telecommunication industry. The binary classification problem for predicting churn involves a noisy labeling function of a human decision process, and typically a rare class. We chose 10 random initial training sets and corresponding pool sets X_{u1} of size 5K. To simulate a realistic distribution of the churn class we selected a test set that contains 3% churn samples, and a training set with 50% churn class. We started with a preliminary experiment involving the Stochastic Gradient Boosting (SGB) classifier with decision trees [16]. We then introduced feature selection based on the output of SGB for the top 25 features. The selected features were used in a following experiment with LAAL and SGB. Results are reported in table 2 for the lift value of the top decile, showing an average improvement of 15% over SGB.

Table 2: Lift values for LAAL and SGB over 10 churn prediction data sets.

Alg./dataset	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	mean
LAAL	2.2	2.6	1.8	2.3	2.3	1.4	2.3	2.2	2.4	1.9	2.1
SGB	1.9	1.8	1.9	1.9	1.8	1.9	2.0	1.9	1.9	1.9	1.9

5.2 Running time experiments.

To demonstrate the efficiency of LAAL we performed running time comparisons with three representative algorithms: First, a baseline linear-running time parametric uncertainty sampling algorithm [20] with a Bayes classifier which in terms of speed should be the fastest algorithm to be examined (marked as 'US'). Second, the closely related clustering-based '+EXPLORE' [4] which uses the passive transductive classifier of [32] and spectral clustering [21] (marked as '+EXPLOR'). And third, an inductive active learning with SVM [7] (marked as 'ISVM'). We set an accuracy goal of 0.8 and run each algorithm until this accuracy is obtained, or until the query pool is exhausted. The experiment shows the tradeoff between accuracy and running time. CPU running time (log scale) vs. data size is reported in Fig. 7. The superiority of LAAL is clear: demonstrating 2-4 orders of magnitude speedup relative to 'US' and '+EXPLOR'. When data has a clear class-cluster structure (e.g. MUSH data set) '+EXPLORE' attains better running time than 'US'. However, LAAL, which also detects clusters, does so more efficiently. The results involving 'ISVM' demonstrate that once a single clear boundary exists, 'ISVM' is very accurate in refining it. However, its computational complexity is shown to be prohibitive. In many cases ISVM simply exhausted the query pool without reaching the desired accuracy: for example, in the 'XOR' data set where ISVM concentrated only on a single decision boundary.

Big data sets. To further demonstrate LAAL's efficiency we experimented with data sets that are 1-2 orders of magnitude bigger, including over 1,000,000 data points. There (as well as on previous experiments shown in Fig. 7) running time could not be measured for all algorithms as they mostly caused the system to crash, or simply run indefinitely beyond 10^5 seconds. In particular, in '+EXPLORE' the spectral clustering is a major computational bottleneck. We took a different approach with the other algorithms in order to provide the reader with an informative comparison: the algorithms were transformed to use batch learning with an uncertainty sampling criterion. Specifically, the query batch of size $b = 20$ was chosen as the sorted set $Q = \{q_{i_1}, ..., q_{i_b}\}$ of minimal $|\hat{\chi}_i|$ values. Batch learning has indeed accelerated the convergence of the algorithms to the desired accuracy. However, at $N \approx 10^4$ both US and ISVM exhausted the pool, run indefinitely, or crashed. We provide three experiments with big data sets: UCI COVERTYPE, UCI SUSY, and XOR-HUGE. Because of space limitations we plot running times for two of them in the bottom of Fig. 7. When applied to XOR-HUGE, ISVM exhausts the pool and crashes at $N \approx 10^4$. LAAL was running at $\sim 10^2$ seconds. For the COVERTYPE data set ISVM crashed at $N \approx 3 \times 10^4$, and shows a very steep slope. LAAL continues from $N \approx 10^4$ to $N = 5.8 \times 10^5$ with $10^2 - 10^3$ seconds. The experiments with the SUSY data set starting at $N = 4.8 \times 10^5$ show a similar trend towards and beyond 1M points.

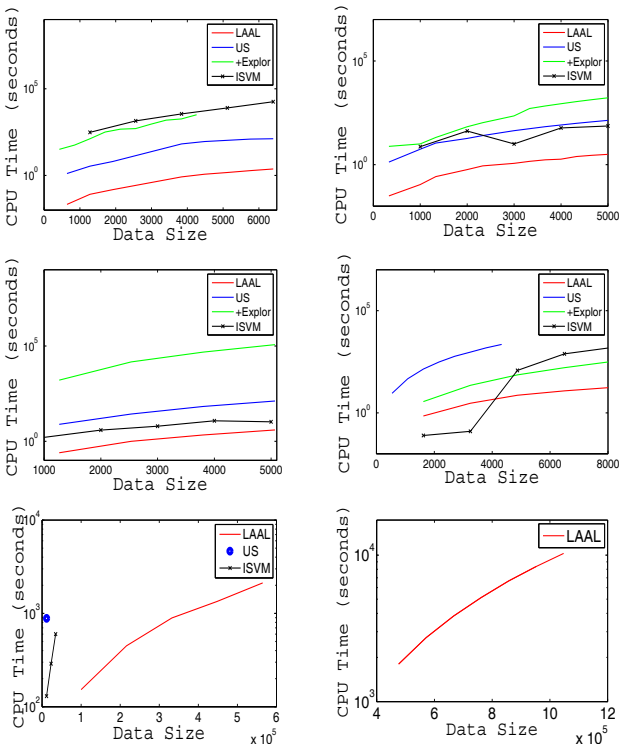


Figure 7: Running time (log-scale) vs. data size. Left to right: XOR and UCI data sets: WAVEFORM, MAGIC, MUSH, COVERTYPE, and SUSY.

6. CONCLUSIONS AND FUTURE WORK

We presented an efficient label diffusion method for active-transductive classification. LAAL demonstrates improved classification results over state-of-the-art active learning algorithms, and a significant reduction of running time. Our approach relies on the combination of classical geometric kernels with label-adapted kernels and on uncertainty sampling for query selection. These properties allow LAAL to efficiently overcome certain challenging situations in active learning such as non-smoothness of the class labeling function over the data distribution. Future work aims at accelerating the iteration process via low rank matrix approximation, an adaptive selection of the σ parameters to adapt to the distance from the nearest class decision boundary, imbalanced class distribution, and batch-mode learning.

7. ACKNOWLEDGMENTS

We thank Dr. Ron Begleiter and Prof. Ran El-Yaniv for the code of '+EXPLORE', and Kun Deng for his contribution of results from [23]. We also thank our anonymous reviewers for their useful comments.

8. REFERENCES

[1] A. Andoni, M. Datar, N. Immorlica, and V. Mirrokni. *Locality-sensitive hashing using stable distributions.* MIT Press, Cambridge, MA, 2006.

[2] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[3] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *JMLR*, 5:255–291, December 2004.

[4] R. Begleiter, R. El-Yaniv, and D. Pechyony. Repairing self-confident active-transductive learners using systematic exploration. *Pattern Recogn. Lett.*, 29(9):1245–1251, July 2008.

[5] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *COLT*, pages 624–638, 2004.

[6] A. Bondu, V. Lemaire, and B. M. Exploration vs. exploitation in active learning : A bayesian approach. In *IJCNN*, pages 1–7, 2010.

[7] C. Campbell, N. Cristianini, and A. J. Smola. Query learning with large margin classifiers. In *ICML*, pages 111–118, 2000.

[8] O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 2006.

[9] R. Chattopadhyay, Z. Wang, W. Fan, I. Davidson, S. Panchanathan, and J. Ye. Batch mode active sampling based on marginal probability distribution matching. In *ACM SIGKDD*, pages 741–749, 2012.

[10] F. R. K. Chung. *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics 92, 1996.

[11] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *PNAS*, 102(21):7426–7431, June 2005.

[12] I. Dagan and S. P. Engelson. Committee-based sampling for training probabilistic classifiers. In *ICML*, pages 150–157, 1995.

[13] G. Dahlquist and A. Björck. *Numerical Methods*. Courier Dover, 2003.

[14] *Churn prediction tournament*, 2003. http://www.fuqua.duke.edu/centers/ccrm/index.html.

[15] E. J. Friedman. Active learning for smooth problems. In *COLT*, 2009.

[16] J. H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, February 2002.

[17] Q. Gu, C. Aggarwal, J. Liu, and J. Han. Selective sampling on graphs for classification. In *ACM SIGKDD*, pages 131–139, 2013.

[18] Y. Hu, D. Zhang, Z. Jin, D. Cai, and X. He. Active learning via neighborhood reconstruction. In *IJCAI*, pages 1415–1421, 2013.

[19] T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, pages 290–297, 2003.

[20] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR*, pages 3–12, 1994.

[21] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[22] H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *PICML*, pages 623–630, 2004.

[23] T. Osugi, D. Kun, and S. Scott. Balancing exploration and exploitation: A new algorithm for active machine learning. In *ICDM*, pages 330–337, 2005.

[24] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *ICML*, pages 441–448, 2001.
[25] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Addison-Wesley Publishing, Boston, MA, 1996.
[26] A. D. Szlam, M. Maggioni, and R. R. Coifman. Regularization on graphs with function-adapted diffusion processes. *JMLR*, 9:1711–1739, June 2008.
[27] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *NIPS*, pages 945–952, 2002.
[28] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2:999–1006, November 2001.
[29] X. Wang, R. Garnett, and J. Schneider. Active search on graphs. In *ACM SIGKDD*, pages 731–738, 2013.
[30] Z. Wang and J. Ye. Querying discriminative and representative samples for batch mode active learning. In *ACM SIGKDD*, pages 158–166, 2013.
[31] F. L. Wauthier, N. Jojic, and M. I. Jordan. Active spectral clustering via iterative uncertainty reduction. In *ACM SIGKDD*, pages 1339–1347, 2012.
[32] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, pages 321–328, 2003.
[33] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Melon, 2002.
[34] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.
[35] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop*, pages 58–65, 2003.

APPENDIX

Proof of Theorem 1

PROOF. We start with showing that a Jacobi iteration using an iteration matrix that is built from the *true* bi-class labels (using m_2 in (15)) results in an approximation to the class decision function:

Definition 2. ***The class weight matrix is defined as*** $W^h(x_i,x_j)=\exp\left(\frac{\rho_2(h(x_i),h(x_j))^2}{\sigma_2}\right)$, $K^h=(D^h)^{-1}W^h$ ***and*** B^h_J ***are the corresponding kernel and iteration matrix. Finally, the class graph Laplacian is defined as*** $L^h=D^h-W^h$.

L^h (and K^h) has similar eigenvectors to those of B^h_J. Let $v_1,...,v_n$ and $\lambda_1,...,\lambda_n$ be the set of eigenvectors of B^h_J ordered in the decreasing order of its eigenvalues. We have that a general solution is given by

$$\chi^{(t+1)}=(B^h_J)^t\chi^{(0)}=c_1\lambda_1^t v_1+...+c_n\lambda_n^t v_n, \tag{16}$$

where $c_1,...,c_n$ are coefficients that are prescribed by the initial condition $\chi^{(0)}$: $\chi^{(0)}=c_1v_1+...+c_nv_n$. For a large enough t, and since $\forall i\ |\lambda_i|<1$, we have that the dominant components in $\chi^{(t+1)}$ are the highest eigenvectors v_1 and v_2. v_1 has a constant sign, and the structure of h: it is piecewise constant on the indices belonging to the same class in h. A second component that may be dominant (for sufficiently small t and since c_2 is large because of negative and positive training samples), is v_2. v_2 obeys $v_2\perp v_1$, and thus approximates a minimal cut in h [10]:

$$MinCut=\min_{h'}\sum_{i,j}W^h_{ij}|h'_i-h'_j|^2. \tag{17}$$

Therefore v_2 is an approximation for the binary function h, and its transition from negative to positive entries is corresponds to the decision boundary. λ_2 corresponds to how well h can be partitioned (the algebraic connectivity of the graph) [10], in particular, if h has a clear partition then λ_2 is close to λ_1. The combination of v_1 and v_2 in (16) forms an approximation to h, since $h'=h$ itself provides the minimum of (17). Clearly, h is partially known and we use instead $\bar{\chi}$ to build up a kernel and the Jacobi iteration process. Since $\bar{\chi}$ is approximates h from training samples and the propagated labels, the same spectral analysis applies for an iteration using $\bar{\chi}$ instead of h.

Now, consider the label-adapted kernel (15) in its matrix representation

$$K^{\hat{\chi}}=(D^{\bar{\chi}})^{-1}(W\cdot W^{\bar{\chi}}), \tag{18}$$

where $W^{\bar{\chi}}$ corresponds to m_2 in (15). The associated energy function is then

$$C(\chi)=\sum_{i,j}W_{ij}\cdot W^{\bar{\chi}}_{ij}|\chi_i-\chi_j|^2. \tag{19}$$

In this case the solution obtained by iterating with $B^{\hat{\chi}}_J$ will have components corresponding to the eigenvectors of $K^{\hat{\chi}}$: a constant sign component $v_1^{\hat{\chi}}$, and a second component $v_2^{\hat{\chi}}$ that reflects a cut in a fused graph constructed from a weight-by-weight multiplication of $W^{\bar{\chi}}_{ij}$ with W_{ij} - the weight of the graph constructed from X. In particular, it will capture a cut that minimizes (19). The cut reflects the data density and geometry as in (4), and, as shown above for W^h the currently approximated hypothesis $h\approx\bar{\chi}$. □

Proof of Theorem 2

PROOF. *Note that i does not include labeled points indices because those are always restarted in our algorithm to* Y_l. *We need to show that*

$$\exp\left(-\frac{\|x_i-x_j\|^2}{\sigma_1}\right)\exp\left(-\frac{|\hat{\chi}_i-\hat{\chi}_j|^2}{\sigma_2}\right)|\hat{\chi}_i-\hat{\chi}_j|^2$$
$$\leq\exp\left(-\frac{\|x_i-x_j\|^2}{\sigma_1}\right)|\bar{\chi}_i-\bar{\chi}_j|^2.$$

We distinguish between two cases 1. $|\bar{\chi}_i-\bar{\chi}_j|\leq|\hat{\chi}_i-\hat{\chi}_j|$, and 2. $|\hat{\chi}_i-\hat{\chi}_j|\leq|\bar{\chi}_i-\bar{\chi}_j|$.

1. *Since* $0<|\bar{\chi}_i-\bar{\chi}_j|\leq|\hat{\chi}_i-\hat{\chi}_j|\leq 2$ *we can choose* σ_2 *so that*

$$\exp\left(-\frac{|\hat{\chi}_i-\hat{\chi}_j|^2}{\sigma_2}\right)|\hat{\chi}_i-\hat{\chi}_j|^2\leq|\bar{\chi}_i-\bar{\chi}_j|^2.$$

Or alternatively we can choose a less tight σ_2 *s.t.*

$$\sum_{j\in l+u}\exp\left(-\frac{|\hat{\chi}_i-\hat{\chi}_j|^2}{\sigma_2}\right)|\hat{\chi}_i-\hat{\chi}_j|^2\leq\sum_{j\in l+u}|\bar{\chi}_i-\bar{\chi}_j|^2.$$

2. *Since* $\exp\left(-\frac{|\bar{\chi}_i-\bar{\chi}_j|^2}{\sigma_2}\right)\leq 1$ *the result is immediate.*

□