1 Introduction

We address the problem of learning the binary node labeling in a given undirected graph under the following assumption: **each node is labeled either** 0 **or** 1 **and the label of any node is equal to** 1 **if and only if at least two adjacent nodes are labeled with** 1. We study this problem within the *active learning setting* but we would like also to investigate it within the *online* setting and *batch* setting.

1.1 Classical node classification problem in machine learning

Formally, in this classification problem we are given a whole undirected, connected, and weighted graph $G(V, E, W)$ with positive edge weights $w_{i,j} > 0$ for each $(i, j) \in E$. We focus on the case in which the labels are binary and all edge weights are equal to 1 (unweighted graph). Hence, a *labeling* of G can be defined as an assignment $\boldsymbol{y} = (y_1, \ldots, y_n) \in \{0, 1\}^n$. The algorithm performance for this problem are usually measured simply with the total number of mistakes occurred. The *mistake bound* of an algorithm is an upper bound for the number of mistakes.

There are two main settings for this problem: *passive* and *active*.

In the passive setting the learner is a pure observer, since it obtains some of the node labels without being able to choose how that information is provided. Passive learning approaches can be subdivided into *batch* and *on-line*. In the batch paradigm, the temporal order in which the training data information is obtained does not matter for the learning task. The learner knows the labels y_i for all i belonging to some node subset $K \subset V$ (*training set*) and the goal is to predict the labels y_j of all j of the set $U \equiv V \setminus K$ (*test set*). The usual separation of data into training and test sets is difficult, since it may affect to a great extent the complexity of the learning task. The problem can be especially hard if, for example, the training set is formed by a subset of nodes of V connected to the rest of the graph through very few edges. In the on-line paradigm, vertices are presented in an arbitrary order. More precisely, at each time step, the algorithm is required to predict the label of a new arbitrarily chosen node. After each prediction, the true label is revealed. As real-world applications typically involve large graphs, on-line learners play an important role because of their good scaling properties. Moreover, on-line classification enables a clear characterization for optimal performance through the analysis of the interplay between the learner and an adversary. The study of this interaction provides theoretical

performance guarantees for the *worst case* input, which plays an important role in the choice of a suitable algorithm for a given problem. Furthermore, in many applications, methods having on-line good performances are often competitive even when used in batch settings.

In the *active setting* of the node classification problem the learner is allowed to choose the subset of training nodes. The choice of these vertices is driven by the goal of minimizing the number of mistakes made on the non-queried nodes. The active setting is motivated by several practical needs. In real-world web-based problems, for example, though the datasets can be very large, only a few labels may be obtained. More specifically the *non-adaptive* (offline) active protocol can be motivated considering that, in many situations, requesting a label is time consuming, since it may involve an expensive experiment. Hence, it may be significantly less costly to run a single batch of experiments in parallel as compared to running experiments in series.

Formally, in the *non-adaptive* (offline) active setting, the learner receives the input graph $G(V, E)$ in a preliminary phase and selects a nonempty node subset $L \subset V$. Thereafter all labels of the nodes in L are revealed at the same time and the learner is required to predict all labels in $V \setminus L$. In the *adaptive* (online) active setting the choice of the training set is driven by the incremental revelation of the labels selected so far, since after each query the learner obtains the label selected.

As pointed out in
(`http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf`) several methods operating in the passive batch setting, can be naturally cast in a regularization framework. These algorithms estimate a function f on set V in such a way that it satisfies two requirements: (i) f is close to the given labels y_i for all $i \in K$ (*fitting constraint*), and (ii) is smooth on the whole node set (*smoothness constraint*). This can be expressed by a linear combination of two terms: a loss function and a regularization term, relative to the first and the second constraints, respectively. Usually, the loss function depends on the difference, for all node $i \in K$, between $f(i)$ and y_i, and the regularization term is expressed as a function of difference, for all pair $(j, k) \in E$, between $f(j)$ and $f(k)$.

These graph methods differ only for the choice of these two terms.

Blum and Chawla
(`http://www.cs.cmu.edu/~avrim/Papers/mincut.ps`) reduce the node classification problem to the *mincut* problem, also known as *st-cut* (source-

target) problem. In the graph obtained via a suitable transformation of the initial dataset, the positive labels act as sources and must be separated from the negative labels, which acts as targets, minimizing the cutsize. The intuition underlying this reduction is simply that nodes that are strongly connected are unlikely to be separated by the mincut. Hence the mincut provides a solution also for the original classification problem, taking into account the homophilic principle. Clearly the sources and targets are fixed in the mincut problem. This implies that in the linear combination of loss and regularization term, the loss function can be seen as multiplied by an infinite coefficient, so that the minimization problem involves the regularization term solely.

This technique can be viewed as giving the most probable configuration to the labeling if each label y_i is seen as a random variable depending only on the labels of the nodes adjacent to i. The random function whose arguments are drawn from the set of the random labels $y_1, \ldots, y_{|V|}$ is a Markov Random Field. More in general, a Markov Random Field is random process defined on the node set of a graph G, in such a way that the random variable v_i associated to node i depends only on all v_j such that $(i, j) \in E$. Hence two random variables v_i are mutually dependent only through a combination of local interactions on the G's structure.

A shortcoming of the method used by Blum et al. is that the solution gives hard classification without confidence. This problem is addressed in
`http://repository.cmu.edu/cgi/viewcontent.cgi?article=1154&context=compsci`
extending the mincut approach by adding randomness to the graph structure.

The learning problem can also be formulated in terms of a Gaussian Random Fields on the given graph. In general, a Gaussian Random Field is defined as a probability measure f of a sample space on which a set X of random variables is mapped, satisfying the following property: for each variable subset $\{x_1, x_2, \ldots\}$, the vector with components $f(x_1), f(x_2), \ldots$ is distributed as a multivariate Gaussian. Zhu et al. in
`http://mlg.eng.cam.ac.uk/zoubin/papers/zgl.pdf` adopt Gaussian fields over a continuous state space rather than random fields over the discrete label set. Their approach is a continuous relaxation to the discrete Markov Random Field and it is intimately connected to random walks and spectral graph theory. The solution function $f : V \to \mathbb{R}$ is forced to take values $f(i) = y_i$ for all nodes $i \in L$, which is equivalent to having a infinite loss coefficient in the linear combination of loss and regularization term (like in the mincut approach of Blum and Chawla –
(`http://www.cs.cmu.edu/~avrim/Papers/mincut.ps`)). The smoothness

penalty term depends only on the differences between $f(i)$ and $f(j)$ for all $i, j \in V$ such that $(i, j) \in E$. The energy function f minimizing the regularization terms is harmonic. An interesting property of this function is that $f(i)$ is equal, for each node $i \in U$, to the weighted average of the values of f for all the i's neighbors. Moreover f is unique, it satisfies $0 \leq f(i) \leq 1$ for all $i \in V$. More precisely $f(i)$ is equal to the probability of hitting, with a random walk starting from node i, a node labeled with 1.

Bengio et al.
(`http://www.iro.umontreal.ca/~lisa/pointeurs/bengio_ssl.pdf`), besides showing how different node classification algorithms can be cast into a common framework where the objective is to minimize a quadratic cost criterion, propose a new method for "propagating" the label values through the edge connections from labeled examples to the whole dataset. This method is inspired by the Jacobi iterative method for linear systems. The approaches spreading the known labels on the other nodes using the graph structure (see also [Learning from labeled and unlabeled data with label propagation]) clearly reflect the homophilic bias, and are known as *label propagation* methods.

Szummer and Jaakkola
(`http://research.microsoft.com/pubs/65581/szummerjaakkola-nips01.pdf`) classify the vertices of the considered graph through the use of a t-step random walk.

In the active setting, one would like to devise a strategy for placing a certain budget of queries on the vertices of a given graph. This should be done so as to minimize the number of mistakes made on the non-queried nodes by some reasonable classifier like mincut. This question has been investigated from a theoretical viewpoint by Guillory and Bilmes
`http://papers.nips.cc/paper/3752-label-selection-on-graphs.pdf`,
and by Afshani et al.
`www.madalgo.au.dk/~peyman/pubdata/07Et.Al.COCOON.ps`.

2 Preliminaries

Let $\boldsymbol{y}$ be the node labeling, i.e. the binary vector in $\{0, 1\}^n$ associated with the node set of $G(V, E)$, where $n = |V|$. Given any node $u \in V$, let $N(u)$ be the set of nodes adjacent to u and define $y_{N(u)}$ as the sum of all labels of the nodes present in the neighborhood $N(u)$: $y_{N(u)} = \sum_{u' \in N(u)} y_{u'}$. Let $\overset{\triangle}{\longleftrightarrow}$

an equivalence relation over a set of triangles of G defined in the following way: given two triangles T and T' we write $T \overset{\triangle}{\longleftrightarrow} T'$ if T shares an edge with T', or if there exist a sequence of triangles $T_1, T_2, \ldots, T_k$ such that T shares an edge with T_1, which in turn shares an edge with T_2, ... , which in turn shares an edge with T_k, which in turn shares an edge with T'.

Now, let $\mathcal{T} \subseteq G$ be a subgraph of $G(V, E)$ such that (i) each node belongs to a triangle, (ii) for each pair of triangles T and T' in $\mathcal{T}$ we have $T \overset{\triangle}{\longleftrightarrow} T'$, and (iii) all triangles in G sharing an edge with any triangle in $\mathcal{T}$ are present in $\mathcal{T}$. We call any subgraph $\mathcal{T}$ satisfying these three properties an EST cluster (cluster of Edge Sharing Triangles). Observe that a single triangle may be an EST cluster.

Finally, we call 1-cycle any cycle entirely labeled with 1. We also call γ-node any node (i) belonging to an EST cluster or (ii) adjacent to two different γ-nodes. We call γ-cluster any *maximal* connected component of $G(V, E)$ where all nodes are γ-nodes. Observe that any EST cluster is a γ-cluster.

Our labeling assumption can be stated in the following way. For each node $u \in V$ we have[1]

$$y_u = 1 \iff y_{N(u)} \geq 2$$

This assumption implies that

1. If there exists at least one node labeled with 1, then there exists a 1-cycle in $G(V, E)$.

2. All the γ-clusters sharing an edge with any 1-cycle must be entirely labeled with 1. This implies that the same property holds for EST clusters.

3. According to our assumption, each EST cluster must be either entirely labeled with 1 or 0 (except the nodes shared by different EST clusters[2]). Furthermore, there are γ-clusters which, unlike EST-clusters, can be labeled using *both labels* in such a way that our assumption is never violated.

[1] It would be interesting to investigate the case $y_u = 1 \iff y_{N(u)} \geq k$ for various values of k (e.g. expressed as a function of n or, more in general, of some graph properties), and the case $y_u = 1 \iff y_{N(u)} \geq c|N(u)|$, where $c \in (0, 1)$.

[2] Observe that when two or more EST clusters share one node u, we have $y_u = 1$ if and only if at least one of these EST clusters is labeled with 1.

3 Active learning on graph nodes

3.1 Problem

Our problem is to find a "small" set of nodes $L \subset V$ such that, once we obtain all labels of the nodes in L, we are able predict with good accuracy the labels of the remaining nodes. More precisely we address a **selection problem** and a **prediction problem**.

According to the *active learning protocol*, in the selection problem we have to select a *small* query set L and we receive all labels of the nodes in L. An integer query budget b is often part of the problem input and in the selection phase the goal is to look for the best L such that $|L| \leq b$. In the prediction problem we need to predict all labels of the nodes in $V \setminus L$ exploiting the labels received at the end of the selection phase. We call L the *query set* and $V \setminus L$ the *test set*.

Consider for example the case in which $G(V, E)$ is formed by

A A clique.

B A set of different EST clusters such that in each pair of EST clusters they are connected by at most one single edge (or they share a single node).

C A set of different γ-clusters such that in each pair of γ-clusters they are connected by at most one single edge (or they share a single node).

D Two different cliques (having the same number of nodes) connected with three edges (u_1, u_2), (u_1, u_3) and (u_4, u_2).

E Two different cliques (having the same number of nodes) connected with four edges (u_1, u_2), (u_1, u_3) and (u_4, u_5) and (u_6, u_5), where u_1, u_4 and u_6 belong to the first clique and u_2, u_3 and u_5 belong to the other clique.

F A set of k line graphs sharing the same two terminal nodes where each line has more than one non-terminal node.

G A wheel graph.

H A wheel and a clique (having the same number of nodes) connected via three edges (u_1, u_2), (u_1, u_3) and (u_4, u_2).

I A "triangle chain", i.e. an EST cluster with maximum diameter.

J A "triangle grid" $n_h \times n_w$, i.e. the graph obtained adding an edge one edge in each square/cycle of four edges and four nodes to a grid $n_h \times n_w$, where n_h and n_w are the height and the width (in terms of number of egdes) of the grid. Observe that graph I is a triangle either n_h or n_w equal to 1.

K A "clique chain" of k cliques of equal size $C_1, C_2, \ldots, C_k$, where C_i and C_{i+1} are connected, for all $1 \leq i \leq k-1$, three additional edges (u_1, u_2), (u_1, u_3) and (u_4, u_2).

We have:

- Graph A, D, G, H, I, J and K: A single query is sufficient for predicting all the test nodes.
- Graph B, C, and E: We need to select one node in each different γ-cluster (in graph E we need only one query for the first clique and one query for second one).
- Graph F: In this case, we need to select one node label for each line which is part of the input graph.

We call **perfect labeling** a labeling where our assumption $y_u = 1 \iff y_{N(u)} \geq 2$ holds for *all* nodes $u \in V$. We call **node flipping** or, equivalently, **label flipping**, the operation of changing the value of a node label (from 0 to 1 or vice versa) and **node removal** the operation of removing a node with all adjacent edges.

Since for real social networks it is natural to assume that $y_u = 1 \iff y_{N(u)} \geq 2$ does not hold for *all* nodes $u \in V$, we consider two possible violations of this rule: *adversarial* and *stochastic*.

3.2 Adversarial setting

Within the adversarial setting the nodes that do not respect our assumption can be seen as a vertex set selected by an *adversary* whose goal is to maximize the number of prediction mistakes made by our algorithm. This number will be therefore expressed in terms of a *measure of irregularity of the labeling*. The game between learner and adversary expresses simply the interplay between the algorithm and *worst case* labeling. In other words we simply focus on the *worst case analysis* of the combination of our selection and prediction algorithm.

We basically found three natural and reasonable complexity measures (measures of irregularity) for this setting.

1. Starting from a perfect labeling, the adversary select a set of nodes $S \in V$ (it is worth to observe that we may have $S \cap L \not\equiv \emptyset$) and flip all labels of the nodes in S. We expect to have a mistake bound depending on the minimum number of *node removals* necessary for obtaining a graph in which our assumption is not violated (i.e. for obtaining a perfect labeling), as well as the graph topology and the choice of L.

2. Perhaps an interesting complexity measure might also capture the fact that, starting from a perfect labeling, a single label flipping can entail that our assumption is violated by several triplets[3]. It might be significant to consider that the violation of a set Z of triplets that (i) share few (compare with $|Z|$) edges and are "supported" by several other triplets that are not violated by the considered label flipping. An example can clarify this concept. Consider graph H (see Section 3.1). A measure with these characteristics would satisfy the following criteria: (i) the flipping of the central node of the wheel has a linear adversarial cost, (ii) the flipping of any other label has a constant adversarial cost, (iii) the cost is always included between 0 and $c|V|$ (for some small constant c, possibly close to 1), that is, the maximum cost cannot be superlinear in $|V|$ and it is "density-independent". With such a measure in graph H we could construct a set L containing the central node of the wheel and a random node in the clique, being able to propagate the label in the prediction phase in order to achieve a mistake bound linear in $|S|$.

It is worth to note that in this setting a suitable use of *randomization* techniques is necessary, since the adversary can, for example, select several (all) labels contained in L, forcing *any* algorithm to obtain a vacuous mistake bound paying no adversarial cost. This appears especially clear on graph I (triangle chain). Moreover, it is also very clear on the *grid* graph (one of the simplest triangle-free graphs), where the adversary can force vacuous mistake bounds even if $S \cap L \equiv \emptyset$, as it is easy to verify (all the labels can be set to 0 and the adversary selects a collection R of rectangles maximizing the total number of nodes contained in R such that the intersection of L

[3]In fact the the use of a biconditional logical connective in the assumption formulation implies the analysis of all triplet $\{u_1, (u_1, u_2), u_2, (u_2, u_3), u_3\}$ with $u_1, u_2, u_3 \in V$ and $(u_1, u_2), (u_2, u_3) \in E$.

and the R's node set is empty). In any case, as we pointed out before, the adversary could also select $S \equiv L$ still forcing a vacuous mistake bound, since the requirement $S \cap L \not\equiv \emptyset$ would appear artificial and it cannot be easily justified.

3.3 Stochastic setting

Another method for taking into account the violations of our basic assumption that may occur in real graphs is the stochastic setting, which can be studied according the following assumptions.

1. Starting from a perfect labeling, each node is flipped with a probability ϵ. Then propagate the flipped labels according to some stochastic method. This entails that a certain set of nodes, depending on ϵ, the graph topology and the stochastic flipped label propagation, violates our basic inductive principle.

2. Starting from a perfect labeling, each node belongs to a subset S with probability ϵ. Then in an adversarial way, each node in S can be flipped or not. Then propagate the flipped labels according to some stochastic method. Again, this entails the violations of our basic assumption for some nodes. We call this setting *semi-random model*.

3. Starting from a perfect labeling, each node $u \in V$ is flipped with a probability ϵ_u proportional to some feature of the node itself (e.g. its degree). Then propagate the flipped labels according to some stochastic method.

In all stochastic setting the complexity measure is given by the (expected) minimum number node removals necessary for obtaining a perfect labeling.

3.4 Input graph class restriction

As explained in the previous section, there are classes of graphs (e.g. the grid graphs) for which the problem appears to be "very worst-case", i.e. no algorithm is able to achieve significant results. Hence, we consider only the class of input graphs where each node belongs to a triangle (i.e. the graphs formed by EST clusters sharing nodes. This choice can be practically motivated focusing on social networks, where it is unlikely that a person/user u does not have, among the $\binom{degree(u)}{2}$ pairs of friends/contact, not even one

pair of people who know each other. Likewise, it is unlikely for a person to have only one contact in the network.

This input restriction makes the problem reasonable but not trivial and the relative assumption seems to be in some sense comparable to requiring a high clustering coefficient.

4 Objective functions for the selection problem

Let $\Lambda(\boldsymbol{y})$ be any (expected in the stochastic setting) minimal node set such that its removal is necessary for obtaining a perfect labeling from $\boldsymbol{y}$.

Following
`http://papers.nips.cc/paper/3752-label-selection-on-graphs.pdf` and `http://researchers.lille.inria.fr/vitale/active.pdf`, we believe that a meaningful objective function for selecting the query set in the adversarial setting could be

$$\mathbb{E}\frac{|V_L(\boldsymbol{y})|}{|\Lambda(\boldsymbol{y})|}$$

where the expectation is over all possible randomized choices of L made by the learner (and, in the stochastic setting, by the adversarial labeling method) and $V_L(\boldsymbol{y})$ is the set of all nodes belonging to the EST clusters that (i) remain after having removed all nodes of $\Lambda(\boldsymbol{y})$ (together with their incident edges) and (ii) do not contain any node in L. This function expresses to what extent is hard to exploit the labeling information of set L in order to learn the whole graph labeling $\boldsymbol{y}$ whose propagation can be interrupted by some nodes violating our basic labeling assumption. Hence, the goal is to find a set L minimizing this function for every adversarial choice of $\boldsymbol{y}$, taking into account that labeling irregularity cost paid by the adversary is $|\Lambda(\boldsymbol{y})|$.

For example, in graph I (triangle chain) a promising technique could be selecting the nodes of L uniformly at random. In graph K one could select a node uniformly at random for each clique in the input. Graph J with n_h and n_w equal to $c\sqrt{n}$, for some constant c, seems to suggest that the stochastic setting is less interesting compare to the adversarial one.

Futhermore in graph I (triangle chain), in the stochastic settings it is convenient to select the set L in such a way that the removal of all its nodes create EST clusters having the same size, as it is easy to verify. If the labels of L are selected in a suitable way during the prediction phase it is not difficult

obtain an expected mistake bound equal to $\mathcal{O}\Big(\frac{|V|}{|L|}\mathbb{E}\big[|\Lambda(\boldsymbol{y})|\big]\Big)$, which seems to be optimal up to constant factors.

5 The prediction problem

Once L is selected, a problem is predicting the labels of the remaining nodes propagating the ones that are revealed. Roughly speaking this propagation should operate in such way to minimize the number of violations that may be generated assigning the labels to the nodes in $V \setminus L$. This operation seems to be in some sense similar (at least in some cases) to minimize the number of edges connecting nodes assigned with different labels (but it is not clear at the moment to what extent the two tasks are comparable).

6 A non-adaptive selection algorithm

We describe a simple and fast selection algorithm that could be efficient (as well as it could be investigated for devising other selection algorithms) for the first two stochastic settings.

We start with $L \equiv \emptyset$. In an incremental fashion we add one node of $V \setminus L$ to L. At each time step we sample a node u from set $V \setminus L$ with a probability proportional to the number of nodes in $V \setminus L$ adjacent to u that are not *controlled*, where we call *controlled nodes* the vertices adjacent to at least one node in the current set L. We add u to L. We terminate when $|L| = b$ (we recall that we denote by b the query budget).

A selection algorithm for the adversarial case could be similar and described in the following way.

We start with $L \equiv \emptyset$. In an incremental fashion we add one node of $V \setminus L$ to L. At each time step we sample a node u_0 from set $V \setminus L$ with a probability proportional to the number of adjacent nodes in $V \setminus L$ adjacent to u_0 that are not *controlled*. We select uniformly at random a node u among all the non-controlled nodes adjacent to u_0. We add u to L. We terminate when $|L| = b$ (we recall that we denote by b the query budget).

Observe that in the adversarial case we estimate the label of u_0 with u. The wheel graph with $b = 1$ is a very simple example highlighting this need. In other words, at the end of the selection phase, we consider u_0 as part of the query set using the sampled label of its adjacent node u.

7 Desiderata

We would like to (i) find natural complexity measures (i.e. irregularity measures/adversarial costs) for our problem settings which are meaningful and, at the same time, allow us to analyze our techniques ($\Lambda(\boldsymbol{y})$ seems promising), (ii) validate the objective functions proposed, (iii) find a selection algorithm able (in an approximate way) to minimize the objective functions considered, (iv) find a prediction algorithm using the selected query set, (v) express the mistake bounds at least for some classes of input graphs, (vi) investigate the problem within the adaptive active setting, (vii) investigate the problem within the online and batch settings.