EPFL

Label propagation via diffusion on Graph

Sara Santos, Martin Josifoski, Antony Doukhan

January, 2021

Contents

1 Introduction

With the development of increasingly more powerful machine learning algorithms, obtaining the labeled examples required to train many traditional target functions is a demanding task, as it requires a considerable effort from human experts. In semi-supervised learning, we alleviate this issue by using unlabeled data, and leveraging it via considering the feature similarity between unlabeled datapoints when assigning them with labels. Rather than acquiring large amounts of labeled examples, it simply requires a small fraction of it and relies on the underlying topology of the data.

This study offers an in-depth mathematical analysis of a semi-supervised label propagation algorithm presented in [1] and its connection to graph kernels and the Representer Theorem. Later on, we evaluate the algorithm's performance on a sentiment analysis dataset comprised of movie reviews from IMDB.

2 Mathematical Framework

2.1 Label Diffusion algorithm

Consider a connected graph $G = (V, E)$ with n data points $\{(x_j, y_j)\}_{j=1}^{n}$. In semi-supervised classification, we aim to predict the labels $\{y_j\}_{j=l+1}^{l+u}$ from the features $\{x_j\}_{j=1}^{n}$ and a restricted number of given examples $\{y_j\}_{j=1}^{l}$, where typically $l \ll u$. We denote by L the set of labeled vertices and U the unlabelled ones. The data manifold structure is fully specified by a weight matrix W which encodes the similarity between features of data points. Note that W is symmetric and all the entries are positive.

Our objective is to compute a function $f : V \to \mathbb{R}$ and use it in assigning labels in accordance with the value it assigns (e.g. $\hat{y}_i = \mathbb{1}\{f(i) \geq 0.5\}$ for 0-1 labeling). This function must satisfy the *Dirichlet boundary conditions* on labeled nodes $f|_L(j) = y_j$. According to the main reference [1], a natural objective function that arises is given by:

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij}(f(i) - f(j))^2. \tag{2.1}$$

This loss function can be seen as a smoothness control over the predictor: similar inputs, as defined by the graph structure (large w_{ij}), should be assigned similar values. If we write the predictor $\boldsymbol{f} = (f(1), ..., f(n))$ as a vector, our tasks consists of minimizing $\frac{1}{2}\boldsymbol{f}^T \Delta \boldsymbol{f}$ subject to $f(j)|_L = y_j$, where $\Delta = D - W$ is the *combinatorial Laplacian matrix* defined on the graph. Differentiating provides the harmonic condition on our predictor, $\Delta \boldsymbol{f} = 0$. By the maximum principle of harmonic functions, this solution is unique and one can express [1] the function at unlabeled nodes $\boldsymbol{f}_u$ as a linear transformation of the already labeled data $\boldsymbol{f}_l$:

$$\boxed{\boldsymbol{f}_u = (D_{uu} - W_{uu})^{-1} W_{ul} \boldsymbol{f}_l = (I_u - P_{uu})^{-1} P_{ul} \boldsymbol{f}_l} \tag{2.2}$$

where the u, l subscript denote the sub-matrices used. In practice, we will have to compute the weight and degree matrices using a similarity metric, compute the predictor using the equation above and assign a label according to a decision rule.

2.2 Functional space and Green operator

In this subsection we derive equation (7) from [1], which is needed for linking our problem to the Representer Theorem formulation seen in class.

Consider the Hilbert space $\mathcal{H}$ of real-valued functions on G, with inner product $\langle f, g \rangle_{\mathcal{H}} = \sum_{x \in G} f(x)g(x)$, which is nothing but the dot product in $\mathbb{R}^n$. In this notation, using the Riesz Representation theorem, $f(j)$ can be seen as the projection of f living in the functional space onto $\mathbf{e}_j$ – a basis vector of the graph representation. Hence, the discrete Laplacian of a function on a graph can be expressed as $\Delta f(j) = \mathbf{e}_j^T(\Delta \boldsymbol{f})$ with Δ being an operator acting on $\mathcal{H}$. Note that Δ is

symmetric and positive semi-definite. From the spectral theorem, we know that the spectrum of Δ is real and positive (if we don't consider the class of constant functions in which we are not interested [2]). This allows us to state that Δ can be decomposed using the eigenfunctions $\{\phi_\alpha\}$ associated to the eigenvalues $\{\lambda_\alpha\}$. Furthermore, Δ is invertible, and one can defines the Green operator as $\Delta\mathcal{G} = \mathcal{G}\Delta = \mathbf{I}$. Thus, we have the following expressions for Δ and $\mathcal{G}$ acting on $\mathcal{H}$:

$$\Delta(\cdot\,,\cdot) = \sum_\alpha \lambda_\alpha \phi_\alpha(\cdot)\phi_\alpha^T(\cdot), \quad \mathcal{G}(\cdot\,,\cdot) = \sum_\alpha \frac{1}{\lambda_\alpha}\phi_\alpha(\cdot)\phi_\alpha^T(\cdot) \tag{2.3}$$

2.3 Laplace equation with Dirichlet boundary conditions

The derivation of what follows can be found in section A of the Appendix. Recall that the predictor f is the solution to the initial problem $\Delta f(j)|_U = 0$ with $f(i)|_L = y_i$. By writing f as a linear combination of the eigen-functions of Δ_U, one can express the predictor as a Kernel classifier, with the kernel being the Green operator:

$$\boxed{f = \sum_{k\in U} \beta_k K(k,\cdot) = \sum_{k\in U}\left(\sum_{i\in L} y_i w_{ik}\right)\mathcal{G}(k,\cdot)} \tag{2.4}$$

Hence, working in the space of real-valued functions acting on G allows us to retrieve equation (7) from the main reference. Note that this is equivalent to Eq.(2.2) where the predictor is computed using matrices, which are operator representations in the graph.

2.4 Smoothness and Representer Theorem

Note that for any function $f : G \to \mathbb{R}$, the cost function (2.1) can be rewritten as :

$$E(f) = \frac{1}{2}\sum_{i,j} w_{ij}(f(i) - f(j))^2 = \langle f, \Delta f\rangle_{\mathcal{H}} = \|f\|_\Delta^2 = \sum_\alpha c_\alpha^2 \lambda_\alpha, \tag{2.5}$$

where the last equality arises from the decomposition $f = \sum_\alpha c_\alpha \phi_\alpha$ in the eigen-functions $\{\phi_\alpha\}$ of Δ. Thus, $\|f\|_\Delta$ quantifies the smoothness of f on G. Consider now a regularizing kernel which should penalize functions that are not smooth over the graph [3], of the form $K = \sum_\alpha \mu_\alpha \phi_\alpha \phi_\alpha^T$. Analogous to what was seen in Exercise Session 10, K defines a reproducing kernel Hilbert space $\mathcal{H}_K$ on $\mathbb{R}^n$ with $\|f\|_{\mathcal{H}_K}^2 = \langle f, f\rangle_{\mathcal{H}_K} = \sum_\alpha c_\alpha^2/\mu_\alpha$. We see that setting $\mu_\alpha = 1/\lambda_\alpha$, i.e., setting K to be the Green operator (Eq.(2.3)), allows us to write the energy minimization as a regularization term $\|f\|_{\mathcal{H}_K}^2$. We can now apply the Representer Theorem to state that any local minimum is of the form (2.4). Section B of the appendix presents the generalization of this result with other regularizers involving the Laplacian operator.

3 Evaluation

3.1 Experimental setup

Dataset: IMDB reviews. The IMDB dataset is a sentiment analysis dataset, composed of 50k movie reviews. In particular, each datapoint corresponds to a review given in textual form associated with a positive or a negative label that reflects its sentiment. In order to analyse the sample efficiency of the algorithm and to make the experiments more computationally manageable, we assume access to disjoint datasets of 5k datapoints for training and 2k datapoints for testing, both randomly sampled from the full dataset. As the original dataset is balanced, for simplicity, we ensure that the subsampled datasets are also balanced.

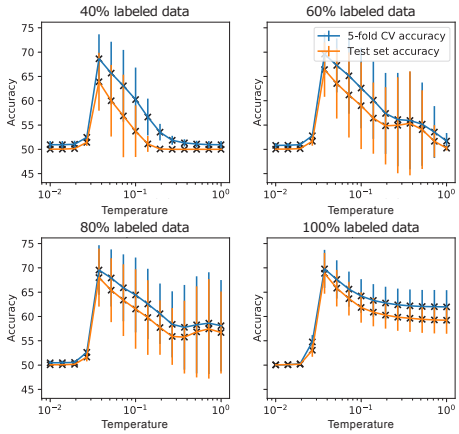
Standard Deviation. For more robust estimation of performance each experiment is repeated with 5 different seeds, that affect the training and testing datasets that are sampled.

Hyperparameters. The hyperparameter tuning is performed using 5-fold stratified cross-validation on the training data. Therefore, the testing dataset is not used for parameter selection and should consequently be more representative of the true test error.

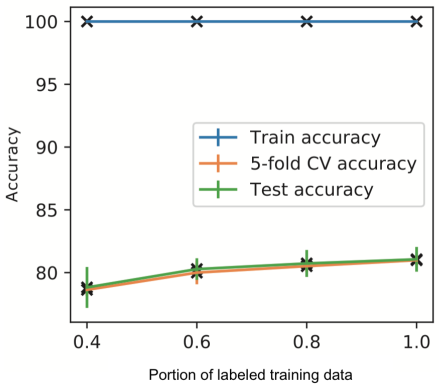
Portion of labeled data. As the label propagation algorithm can take advantage of unlabeled datapoints, we conduct experiments with varying portions of the training data labeled – 40%, 60%, 80% or 100%, in particular.

3.2 Network Construction

For both training and testing, we represent the input text as TF-IDF-weighted bag-of-word vectors, where the IDF weights are computed on the training set only (note that the unlabeled data is also used). Subsequently, we impose a network structure on the data, by considering each review as a node, and assigning the edge between any two nodes u and v with a weight given by $W_{uv} = \exp\left(-\frac{1}{T}\left(1 - \frac{u^T v}{|u||v|}\right)\right)$ where T is a temperature parameter.



(a) Semi-supervised classification



(b) Random forest classification

Figure 1: Standalone classification with semi-supervised label propagation algorithm (a) and supervised random forest classification (b). The error bars correspond to the standard deviation.

3.3 Semi-Supervised Classification

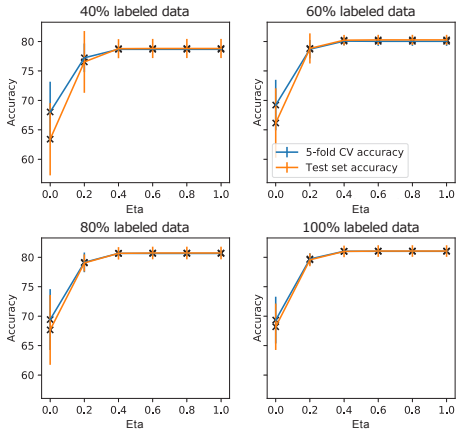
We start by performing semi-supervised classification using the label propagation algorithm. In particular, we employ the scores assigned by the algorithm as defined in Eq.(2.2), and classify reviews that are assigned a value higher than a threshold $\tau = 0.5$ as positive, while the ones below it as negative. For unbalanced classification problems, the imbalance can be accounted for by adjusting the score function, and subsequently the same threshold can be used.

In this setup, there is a single hyper-parameter, the temperature, that we select using cross-validation. Irrespective of the portion of labeled data, we get the best results in terms of CV accuracy – which were confirmed by the test accuracy – when setting the temperature to $T = 0.04$. The results are given in Fig.(1a). As expect, increasing the portion of labeled data increases the test accuracy from 64% when training with 40% labeled data points, to 70% when training with 100% labeled data points. Additionally, increasing the portion of labeled data points decreases the variance in the accuracy across the 5 different seeds.

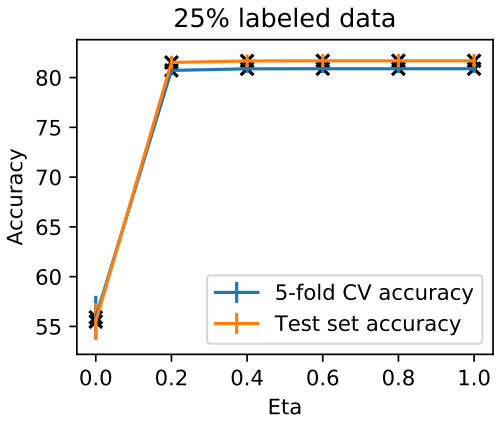
3.4 Supervised Classification

For comparison, we now train a standard supervised classifier, as a baseline. In particular, we opt for using a Random Forest classifier with the default parameters given by the implementation in the scikit-learn library. The results given in Fig.(1b) showcase that even with only 40% of the training data labeled, the Random Forest classifier robustly (with low-standard deviation across different runs/seeds) outperforms the semi-supervised label classification algorithm. As expected,

the random forest classifier overfits the training data, but generalizes well, with an accuracy that could be improved by further tuning the hyper-parameters. However, as this is not the topic of this study, we do not pursue it further.



(a) Joint classification with 5k training samples



(b) Joint classification with 25k training samples

Figure 2: Joint classification using the label propagation algorithm combined with an external classifier using 5k training samples (a) and 25k training samples (b). The error bars correspond to the standard deviation.

3.5 Joint Classification

We now incorporate the predictions of the supervised classifier in the label propagation algorithm's scoring. In particular, to each unlabeled node, we add an edge to a dummy node associated with the prediction of the supervised classifier and weight the edge with η, which is a hyper-parameter. The scores from the label propagation on this "enriched" graph, are given by:

$$\boldsymbol{f_u} = (I - (1-\eta)P_{uu})^{-1}\left((1-\eta)\,P_{ul}\boldsymbol{f_l} + \eta\hat{\boldsymbol{f}}_{\boldsymbol{u}}\right)$$

where $\hat{\boldsymbol{f}}_{\boldsymbol{u}}$ are the supervised classifier's predictions. To construct the network, we set the temperature parameter to the optimal value of $T = 0.04$, according to the cross-validation performed in the semi-supervised classification setting. Thus, we only cross-validate η. Note that $\eta = 0$ corresponds to classification by the label propagation algorithm, while $\eta = 1$ is equivalent to the supervised classification scenario, and anything in-between is an interpolation thereof.

As showcased in Fig.(2a), the results get significantly better for $\eta >= 0.2$, suggesting that incorporating the supervised classifier is very beneficial. We also observe that $\eta = 0.4$ yields the best results independently of the amount of labeled data, however the performance difference with the "vanilla" – untuned – supervised classifier (the model with $\eta = 1$) is only marginal.

We provide further evidence of this behavior in our last experiment, where we increase the number of training datapoints to 25k, but keep only 25% of it labeled. Thus, we have an equal number of labeled training datapoints as in the previous experiments with 100% of the training data being labeled. The results in Fig.(2b) imply that despite the 5 fold increase in the number of datapoints, most of the performance benefits still come from the supervised classifier.

4 Conclusion

This paper expands on the mathematical connection between the label propagation algorithm based on harmonic functions and graph kernels. Experiments on the task of sentiment analysis on movie reviews show that, even with a small number of labeled datapoints, the supervised classifier is still superior to the label propagation algorithm. This suggests that unless the network structure captures significant aspects of the data manifold structure, the label propagation algorithm will not lead to performance gains. Finally, we hypothesize that capturing the data manifold for many problems arising in practice will in general be hard endeavour.

References

[1] X. Zhu, Z. Gharamani, J. Laferty (2003). Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. *International Conference on Machine Learning.*

[2] F. Chung (1997). Lectures on Spectral Graph Theory. *University of Pennsylvania*

[3] O. Chapelle, B. Shölkopf, A. Zien (2005). Semi-Supervised Learning. *Massachusetts Institute of Technology.*

[4] A.J. Smola, R. Kondor (2003). Kernels and Regularization on Graphs.

Appendix

A Kernel Classifier

Consider our initial harmonic problem with constrain on the labeled data :

$$\begin{cases} \Delta f(j) = 0 & \forall j \in U \\ f(i) = y_i & \forall i \in L \end{cases} \tag{A.1}$$

By writing $f = \sum_\alpha c_\alpha \phi_\alpha$ as a linear combination of eigen-functions of Δ_U, we derive the Kernel classifier solution using some results from [2]. First define f_0 as follows :

$$\begin{cases} f_0(k) = 0 & \forall k \in U \\ f_0(i) = y_i & \forall i \in L \end{cases} \tag{A.2}$$

By noting $\{\lambda_\alpha\}$ the spectrum of Δ_U we then have :

$$\begin{aligned} \lambda_\alpha c_\alpha &= \lambda_\alpha \langle \phi_\alpha, f_U \rangle_{\mathcal{H}} \\ &= \langle \Delta_U \phi_\alpha, (f - f_0)_U \rangle_{\mathcal{H}} \\ &= \langle \phi_\alpha, \Delta_U (f - f_0)_U \rangle_{\mathcal{H}} \\ &= \langle \phi_\alpha, (-\Delta f_0)_U \rangle_{\mathcal{H}} \\ &= -\sum_{k \in U} \phi_\alpha(k) \Delta f_0(k) \\ &= -\sum_{k \in U} \phi_\alpha(k) \sum_{i \neq k} (f_0(k) - f_0(i)) w_{ki}, \quad \text{using the definition of } \Delta \text{ and } f_0 \text{ properties} \\ &= \sum_{i \in L} \sum_{k \in U} \phi_\alpha(k) y_i w_{ki}, \qquad \text{since } f_0 \text{ is non-zero on } L \text{ only} \end{aligned} \tag{A.3}$$

Thus we have the final expression for the evaluation $f(j)$ when $j \in U$:

$$\begin{aligned} f(j) &= \sum_\alpha c_\alpha \phi_\alpha(j) \\ &= \sum_\alpha \left(\frac{1}{\lambda_\alpha} \sum_{k \in U} \sum_{i \in L} \phi_\alpha(k) y_i w_{ki} \right) \phi_\alpha(j) \\ &= \sum_{i \in L} \sum_{k \in U} y_i w_{ki} \left(\sum_\alpha \frac{1}{\lambda_\alpha} \phi_\alpha(k) \phi_\alpha(j) \right) \\ &= \boxed{\sum_{i \in L} \sum_{k \in U} y_i w_{ki} \mathcal{G}(k, j)} \end{aligned} \tag{A.4}$$

Where we have used the definition of Green's operator from Eq.(2.3), evaluated at k, j.

B Other regularization kernels

We generalize the theoretical results to other types of regularizers that inherit from the Laplacian operator. We can define a class of regularization functionals on graphs as:

$$\langle f, f \rangle_P = \boldsymbol{f}^T P \boldsymbol{f} = \boldsymbol{f}^T \tau(\Delta) \boldsymbol{f} \tag{B.1}$$

where $\tau(\Delta) = \sum_\alpha \tau(\lambda_\alpha) \phi_\alpha \phi_\alpha^T$ can be interpreted as the application of the function τ to the eigenvalues of Δ. To ensure the smoothness properties of the function, τ should be monotonically increasing in λ and positive. We then have from [4] the following theorem:

Theorem. *For any regularization operator P, denote by $\mathcal{H}_K$ the image of $\mathbb{R}^n$ under P. Then $\mathcal{H}_K$ with dot product $\langle f, f\rangle_{\mathcal{H}_K} = \boldsymbol{f}^T P \boldsymbol{f}$ is a Reproducing Kernel Hilbert Space and its kernel is given by $K = P^{-1}$.*

Indeed the reproducing property $f(j) = \langle f, K(j, \cdot)\rangle_{\mathcal{H}_K}$ is true if and only if $K = P^{-1}$, and in eigenvector notation:

$$K = \sum_{\alpha} \tau^{-1}(\lambda_\alpha)\phi_\alpha \phi_\alpha^T. \tag{B.2}$$

We thus have established a way of constructing a RKHS with respect to a regularizer $\tau(\Delta)$ that depends on the Laplacian operator acting on the graph. Moreover, $\|f\|^2_{\mathcal{H}_K} = \langle f, f\rangle_{\mathcal{H}_K} = \boldsymbol{f}^T K^{-1} \boldsymbol{f}$ is the regularizer term that appears in the Representer theorem and its minimization is given by a kernel expansion.

Examples We give some commonly used regularizers presented in [3]:

- Regularized Laplacian : $\tau(\lambda) = \sigma^2\lambda, \quad \lambda \neq 0$
- Diffusion process : $\tau(\lambda) = \exp(\sigma^2\lambda/2)$
- p-step random walk : $\tau(\lambda) = (\alpha - \lambda)^{-p}$

We see that our problem of energy minimization (Eq.(2.5)) corresponds to the Regularized Laplacian case where the regularizer is just the identity applied to the Laplacian (with $\sigma = 1$) and the corresponding kernel is the Green operator.